Cyclone V Device Handbook

Volume 3: Hard Processor System Technical Reference Manual

Subscribe

Send Feedback

cv_5v4
2014.06.30

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

Scan Manager....6-1

System Interconnect....7-1

HPS-FPGA Bridges....8-1

Cortex-A9 Microprocessor Unit Subsystem..........9-1

Introduction to Cyclone V Hard Processor System

1

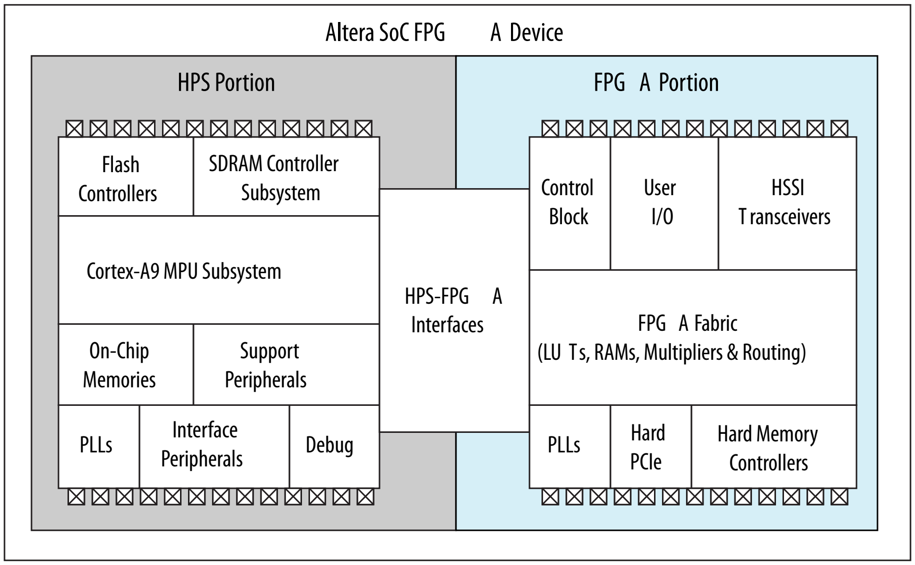
2014.07.31

cv_54001 Subscribe Send Feedback

The Cyclone V device is a single-die system on a chip (SoC) that consists of two distinct parts—a hard processor system (HPS) portion and an FPGA portion.

The following figure shows a high-level block diagram of the Altera SoC device.

Figure 1-1: Altera SoC FPGA Device Block Diagram



The HPS contains a microprocessor unit (MPU) subsystem with single or dual ARM® Cortex®-A9 MPCore processors, flash memory controllers, SDRAM L3 Interconnect, on-chip memories, support peripherals, interface peripherals, debug capabilities, and phase-locked loop (PLLs). The dual-processor HPS supports symmetric (SMP) and asymmetric (AMP) multiprocessing.

The FPGA portion of the device contains the FPGA fabric, a control block (CB), phase-locked loops (PLLs), and depending on the device variant, high-speed serial interface (HSSI) transceivers, hard PCI Express® (PCIe®) controllers, and hard memory controllers.

The HPS and FPGA portions of the device are distinctly different. The HPS can boot from multiple sources, including the FPGA fabric and external flash. In contrast, the FPGA must be configured through either the HPS or an externally supported device.

The HPS and FPGA portions of the device each have their own pins. Pins are not freely shared between the HPS and the FPGA fabric. The HPS I/O pins are configured by software executing in the HPS. Software executing on the HPS accesses control registers in the system manager to assign HPS I/O pins to the available HPS modules. The FPGA I/O pins are configured by an FPGA configuration image through the HPS or any external source supported by the device.

The MPU subsystem can boot from flash devices connected to the HPS pins. Or, when the FPGA portion is configured by an external source, the MPU subsystem can boot from memory available to the FPGA portion of the device.

The HPS and FPGA portions of the device have separate external power supplies and independently power on. You can power on the HPS without powering on the FPGA portion of the device. However, to power on the FPGA portion, the HPS must already be on or powered on at the same time as the FPGA portion. You can also turn off the FPGA portion of the device while leaving the HPS power on.

Related Information

- **Cyclone V Device Overview**
  For information about the FPGA portion of the device, refer to *Cyclone V Device Handbook*.
- **Booting and Configuration Introduction** on page 30-1

Features of the HPS

The following list contains the main modules of the HPS:

- MPU subsystem featuring dual ARM Cortex-A9 MPCore processors
- General-purpose Direct Memory Access (DMA) controller
- Two Ethernet media access controllers (EMACs)
- Two USB 2.0 On-The-Go (OTG) controllers
- NAND flash controller
- Quad SPI flash controller
- Secure Digital (SD) / MultiMediaCard (MMC) controller
- Two serial peripheral interface (SPI) master controllers
- Two SPI slave controllers
- Four inter-integrated circuit (I^2C) controllers
- 64 KB on-chip RAM
- 64 KB on-chip boot ROM
- Two UARTs
- Four timers
- Two watchdog timers
- Three general-purpose I/O (GPIO) interfaces
- Two controller area network (CAN) controllers
- ARM CoreSight™ debug components
  - Debug Access Port (DAP)
  - Trace Port Interface Unit (TPIU)
  - System Trace Macrocell (STM)
  - Program Trace Macrocell (PTM)
  - Embedded Trace Router (ETR)
  - Embedded Cross Trigger (ECT)
- System Manager

- Clock Manager
- Reset Manager
- Scan manager
- FPGA manager

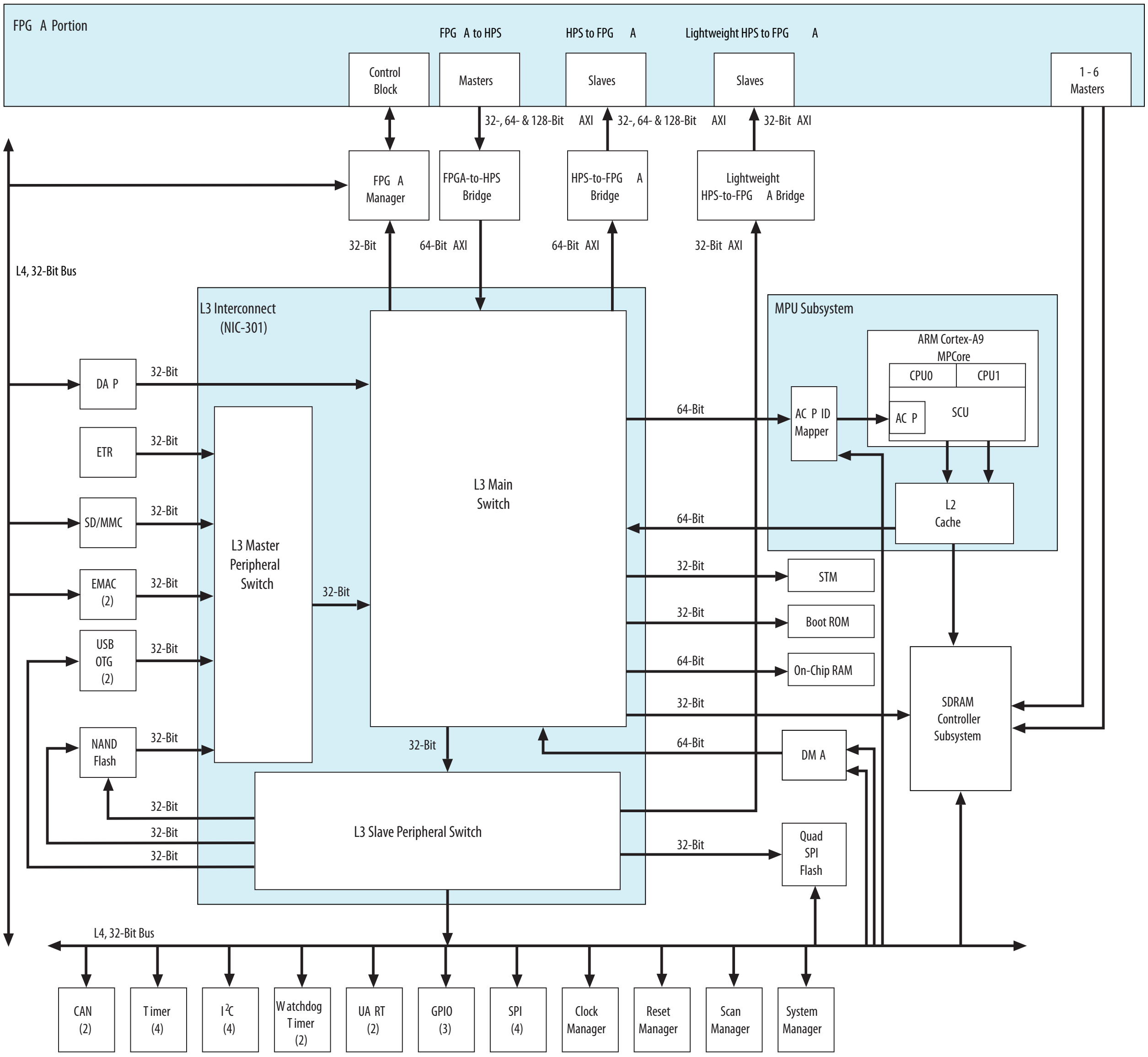
HPS Block Diagram and System Integration

The following figure shows a block diagram of the HPS, excluding the debug system module.

Note: The HPS incorporates third-party intellectual property (IP) from several vendors.

HPS Block Diagram

Figure 1-2: HPS Block Diagram



MPU Subsystem

The MPU subsystem provides the following functionality:

- ARM Cortex-A9 MPCore
  - One or two ARM Cortex-A9 processors in a cluster
  - NEON™ Single Instruction, Multiple Data (SIMD) coprocessor and Vector Floating-Point v3 (VFPv3) per processor
  - Snoop Control Unit (SCU) to ensure coherency within the cluster
  - Accelerator coherency port (ACP) that accepts coherency memory access requests
  - Interrupt controller
  - One general-purpose timer and one watchdog timer per processor
  - Debug and trace features
  - 32 KB instruction and 32 KB data level 1 (L1) caches per processor
  - Memory management unit (MMU) per processor
- ARM L2-310 level 2 (L2) cache
  - Shared 512 KB L2 cache
- ACP ID mapper
- Maps the 12-bit ID from the level 3 (L3) interconnect to the 3-bit ID supported by the ACP

As shown in the "HPS Block Diagram", the L2 cache has one 64-bit master port that is connected to the L3 interconnect, one 64-bit master port connected to the SDRAM L3 Interconnect, and three ports that connect the FPGA to the SDRAM L3 Interconnect. A programmable address filter in the L2 cache controls which portions of the 32-bit physical address space use each master.

Related Information

- **HPS Block Diagram** on page 1-3
- **Cortex-A9 Microprocessor Unit Subsystem** on page 9-1

Interconnect

The interconnect consists of the L3 interconnect and level 4 (L4) buses. The L3 interconnect is an ARM NIC-301 module composed of the following switches:

- L3 main switch
  - Connects the master, slaves, and other subswitches
  - Provides 64-bit switching capabilities
- L3 master peripheral switch
  - Connects master ports of peripherals with integrated DMA controllers to the L3 main switch
- L3 slave peripheral switch
  - Connects slave ports of peripherals to the L3 main switch

The L4 buses are each connected to a master in the L3 slave peripheral switch.

- Each L4 bus is 32 bits wide and is connected to multiple slaves.
- Each L4 bus operates on a separate clock source.

Related Information

System Interconnect on page 7-1

Memory Controllers

SDRAM Controller Subsystem

The SDRAM controller subsystem is mastered by HPS masters and FPGA fabric masters.

The SDRAM controller subsystem implements the following high-level features:

- Support for double data rate 2 (DDR2), DDR3, and low-power double data rate 2 (LPDDR2) devices
- Software-configurable priority scheduling on individual SDRAM bursts
- Error correction code (ECC) support, including calculation, single-bit error correction and write-back, and error counters
- Fully-programmable timing parameter support for all JEDEC-specified timing parameters
- All ports support memory protection and mutual accesses
- Support for ARM Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) quality of service (QoS) for the fabric interfaces

The SDRAM controller subsystem is composed of the SDRAM controller and the DDR PHY.

Related Information

Functional Description—HPS Memory Controller on page 11-1

SDRAM Controller

The SDRAM controller contains a multiport front end (MPFE) that accepts requests from HPS masters and from soft logic in the FPGA fabric via the FPGA-to-HPS SDRAM interface.

The SDRAM controller offers the following features:

- Up to 4 GB address range
- 8-, 16-, and 32-bit data widths
- Optional ECC support
- Low-voltage 1.35V DDR3L and 1.2V DDR3U support
- Full memory device power management support
- Two chip selects

The SDRAM controller provides the following features to maximize memory performance:

- Command reordering (look-ahead bank management)
- Data reordering (out of order transactions)
- Deficit round-robin arbitration with aging for bandwidth management
- High-priority bypass for latency sensitive traffic

Related Information

Functional Description—HPS Memory Controller on page 11-1

DDR PHY

The DDR PHY interfaces the single port memory controller to the HPS memory I/O.

Related Information

Functional Description—HPS Memory Controller on page 11-1

NAND Flash Controller

The NAND flash controller is based on the Cadence®Design IP® NAND Flash Memory Controller and offers the following functionality and features:

- Supports one x8 NAND flash device
- Supports Open NAND Flash Interface (ONFI) 1.0
- Supports NAND flash memories from Hynix, Samsung, Toshiba, Micron, and ST Micro
- Supports programmable 512 byte (4-, 8-, or 16-bit correction) or 1024 byte (24-bit correction) error correction code (ECC) sector size
- Supports pipeline read-ahead and write commands for enhanced read/write throughput
- Supports devices with 32, 64, 128, 256, 384, or 512 pages per block
- Supports multiplane devices
- Supports page sizes of 512 bytes, 2 kilobytes (KB), 4 KB, or 8 KB
- Supports single-level cell (SLC) and multi-level cell (MLC) devices with programmable correction capabilities
- Provides internal direct memory access (DMA)
- Provides programmable access timing

Related Information

NAND Flash Controller on page 13-1

Quad SPI Flash Controller

The Quad SPI flash controller is based on Cadence Quad SPI Flash Controller and offers the following features:

- Supports SPIx1, SPIx2, or SPIx4 (Quad SPI) serial NOR flash devices
- Supports direct access and indirect access modes.
- Supports single, dual, and quad I/O instructions
- Programmable data frame size of 8, 16, or 32 bits
- Support up to four chip selects

Related Information

Quad SPI Flash Controller on page 15-1

SD/MMC Controller

The SD/MMC controller is based on Synopsys® DesignWare® Mobile Storage Host controller and offers the following features:

- Integrated descriptor-based DMA
- Supports CE-ATA digital protocol commands
- Supports single card
- Single data rate (SDR) mode only
- Programmable card width: 1-, 4-, and 8-bit
- Programmable card types: SD, SDIO, or MMC version 4.3 and 4.4 devices
- Up to 64 KB programmable block size

Note: For an inclusive list of the programmable card types versions supported, refer to the *SD/MMC Controller* chapter.

Related Information

SD/MMC Controller on page 14-1

Support Peripherals

Clock Manager

The clock manager offers the following features:

- Manages clocks for HPS
- Supports dynamic clock tuning

Related Information

Clock Manager on page 2-1

Reset Manager

The reset manager manages resets for HPS.

Related Information

Reset Manager on page 3-1

System Manager

The system manager offers the following features:

- ECC monitoring and control
- Pin multiplexing
- Low-level control of peripheral features not accessible through the control and status registers (CSRs)
- Freeze controller that places I/O elements into a safe state for configuration

Related Information

System Manager on page 5-1

Scan Manager

The scan manager drives serial scan-chains to FPGA JTAG and HPS I/O bank configuration.

Related Information

Scan Manager on page 6-1

Timers

The four timers are based on the Synopsys DesignWare APB Timer peripheral and offer the following features:

- 32-bit timer resolution
- Free-running timer mode
- Programmable time-out period up to approximately 86 seconds (assuming a 50 MHz input clock frequency)
- Interrupt generation

Related Information

Timer on page 23-1

Watchdog Timers

The two watchdog timers are based on the Synopsys DesignWare APB Watchdog Timer peripheral and offer the following features:

- 32-bit timer resolution
- Interrupt request
- Reset request
- Programmable time-out period up to approximately 86 seconds (assuming a 50 MHz input clock frequency)

Related Information

Watchdog Timer on page 24-1

DMA Controller

The DMA controller provides high-bandwidth data transfers for modules without integrated DMA controllers. The DMA controller is based on the ARM Corelink™ DMA Controller (DMA-330) and offers the following features:

- Micro-coded to support flexible transfer types
- Supports up to eight channels
- Supports flow control with 31 peripheral handshake interfaces

Related Information

DMA Controller on page 16-1

FPGA Manager

The FPGA manager offers the following features:

- Manages configuration of the FPGA portion of the device
- 32-bit fast passive parallel configuration interface to the FPGA CSS block
- Partial reconfiguration
- Compressed FPGA configuration images
- Advanced Encryption Standard (AES) encrypted FPGA configuration images
- Monitors configuration-related signals in FPGA
- Provides 32 general-purpose inputs and 32 general-purpose outputs to the FPGA fabric

Related Information

FPGA Manager on page 4-1

Interface Peripherals

EMACs

The two EMACs are based on the Synopsys DesignWare 3504-0 Universal 10/100/1000 Ethernet MAC and offer the following features:

- Supports 10, 100, and 1000 Mbps standard
- Supports RGMII external PHY interface
- Provides full GMII interface when using FPGA interface
- Integrated DMA controller

Related Information

Ethernet Media Access Controller on page 17-1

USB Controllers

The two USB 2.0 On-The-Go (OTG) controllers are based on the Synopsys DesignWare Cores USB 2.0 Hi-Speed On-The-Go controller and offer the following features:

- Complies with both Revision 1.3 and Revision 2.0 of the "On the Go and Embedded Host Supplement to the USB Revision 2.0 Specification
- Supports software-configurable modes of operation between OTG 1.3 and OTG 2.0
- Supports all USB 2.0 speeds:
  - High speed (HS, 480-Mbps)
  - Full speed (FS, 12-Mbps)
  - Low speed (LS, 1.5-Mbps)

  Note: In host mode, all speeds are supported. However, in device mode, only high speed and full speed are supported.

Ont the integration side, the USB OTG controller supports the following features:

- Different clocks for system and PHY interfaces
- Dedicated TX FIFO buffer for each device IN endpoint in direct memory access (DMA) mode
- Packet-based, dynamic FIFO memory allocation for endpoints for small FIFO buffers and flexible, efficient use of RAM that can be dynamically sized by software
- Ability to change an endpoint's FIFO memory size during transfers
- Clock gating support during USB suspend and session-off modes
  - PHY clock gating support
  - System clock gating support
- Data FIFO RAM clock gating support
- Local buffering with error correction code (ECC) support

  Note: The USB OTG controller does not support the following protocols:

  - Enhanced Host Controller Interface (EHCI)
  - Open Host Controller Interface (OHCI)
  - Universal Host Controller Interface (UHCI)
- Supports USB 2.0 ULPI PHYs (SDR mode only)
- Supports up to 16 bidirectional endpoints, including control endpoint 0

  Note: Only seven periodic device IN endpoints are supported.
- Supports up to 16 host channels

  Note: In host mode, when the number of device endpoints is greater than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints
- Supports generic root hub
- Supports automatic ping capability

Related Information

USB 2.0 OTG Controller on page 18-1

I^{2}C Controllers

The four I^{2}C controllers are based on Synopsys DesignWare APB I^{2}C controller and offer the following features:

- Two controllers support I^{2}C management interfaces for the EMAC controllers which are for Ethernet control
- Support both 100 KBps and 400 KBps modes
- Support both 7-bit and 10-bit addressing modes
- Support master and slave operating mode
- Direct access for host processor
- DMA controller may be used for large transfers

Related Information

I^{2}C Controller on page 20-1

UARTs

The two UART modules are based on Synopsys DesignWare APB Universal Asynchronous Receiver/ Transmitter peripheral and offer the following features:

- 16550 compatible UART
- Supports the auto flow control as specified in 16750 specification
- Supports IrDA 1.0 SIR mode
- Programmable baud rate up to 115.2 Kbps
- Direct access for host processor
- DMA controller may be used for large transfers

Related Information

UART Controller on page 21-1

CAN Controllers

The two CAN controllers are based on the Bosch® D_CAN controller and offer the following features:

- Compliant with CAN protocol specification 2.0 part A & B
- Programmable communication rate up to 1 Mbps
- Holds up to 128 messages
- Supports 11-bit standard and 29-bit extended identifiers
- Programmable interrupt scheme
- Direct access for host processor
- DMA controller may be used for large transfers
- Available on certain device variants only

Related Information

CAN Controller Introduction on page 25-1

SPI Master Controllers

The two SPI master controllers are based on Synopsys DesignWare Synchronous Serial Interface (SSI) controller and offer the following features:

- Programmable data frame size from 4 to 16 bits
- Supports full- and half-duplex
- Supports up to four chip selects
- Direct access for host processor
- DMA controller may be used for large transfers

Related Information

SPI Controller on page 19-1

SPI Slave Controllers

The two SPI slave controllers are based on Synopsys DesignWare Synchronous Serial Interface (SSI) controller and offer the following features:

- Programmable data frame size from 4 to 16 bits
- Supports full- and half-duplex
- Direct access for host processor
- DMA controller may be used for large transfers

Related Information

SPI Controller on page 19-1

GPIO Interfaces

The HPS provides three GPIO interfaces that are based on Synopsys DesignWare APB General Purpose Programming I/O peripheral and offer the following features:

- Supports digital de-bounce
- Configurable interrupt mode
- Supports up to 71 I/O pins and 14 input-only pins, based on device variant
- Supports up to 67 I/O pins and 14 input-only pins

Related Information

General-Purpose I/O Interface on page 22-1

On-Chip Memory

On-Chip RAM

The on-chip RAM offers the following features:

- 64 KB size
- 64-bit slave interface
- High performance for all burst lengths

Related Information

On-Chip Memory on page 12-1

Boot ROM

The boot ROM offers the following features:

- 64 KB size
- Contains the code required to support HPS boot from cold or warm reset
- Used exclusively for booting the HPS

Related Information

On-Chip Memory on page 12-1

Endian Support

The HPS is natively a little-endian system. All HPS slaves are little-endian.

The processors masters are software configurable to interpret data as little-endian or big-endian, byte-invariant (BE8). All other masters, including the USB interface, are little-endian.

The FPGA-to-HPS, HPS-to-FPGA, FPGA-to-SDRAM, and lightweight HPS-to-FPGA interfaces are little-endian.

If a processor is set to BE8 mode, software must convert endianness for accesses to peripherals and DMA linked lists in memory.

The ARM Cortex-A9 MPU supports a single instruction to change the endianness of the processor and provides the REV and REV16 instructions to swap the endianness of bytes or half-words respectively. The MMU page tables are software configurable to be organized as little-endian or BE8.

The ARM DMA controller is software configurable to perform byte lane swapping during a transfer.

HPS-FPGA Interfaces

The HPS-FPGA interfaces provide a variety of communication channels between the HPS and the FPGA fabric. The HPS is highly integrated with the FPGA fabric, resulting in thousands of connecting signals. The HPS-FPGA interfaces include:

- FPGA-to-HPS bridge—a high-performance bus with a configurable data width of 32, 64, and 128 bits, allowing the FPGA fabric to master transactions to the slaves in the HPS. This interface allows the FPGA fabric to have full visibility into the HPS address space. This interface also provides access to the coherent memory interface.
- HPS-to-FPGA bridge—a high-performance interface with a configurable data width of 32, 64, and 128 bits, allowing the HPS to master transactions to slaves in the FPGA fabric.
- Lightweight HPS-to-FPGA bridge—an interface with a 32-bit fixed data width, allowing the HPS to master transactions to slaves in the FPGA fabric.
- FPGA clocks and resets—provide flexible clocks to and from the HPS.
- HPS-to-FPGA JTAG—allows the HPS to master the FPGA JTAG chain.
- TPIU trace—sends trace data created in the HPS to the FPGA fabric.
- FPGA System Trace Macrocell (STM) events—an interface that allows the FPGA fabric to send hardware events stored in the HPS trace using STM.
- FPGA cross-trigger—an interface that allows triggers to and from the CoreSight trigger system.
- DMA peripheral interface—multiple peripheral-request channels.
- FPGA manager interface—signals that communicate with FPGA fabric for boot and configuration.

- Interrupts—allow soft IP to supply interrupts directly to the MPU interrupt controller.
- MPU standby and events—signals that notify the FPGA fabric that the MPU is in standby mode and signals that wake up Cortex-A9 processors from a wait for event (WFE) state.
- HPS debug interface - an interface that allows HPS debug control domain (debug APB) to extend into FPGA

Related Information

HPS-FPGA Bridges on page 8-1

HPS Address Map

The address map specifies the addresses of slaves, such as memory and peripherals, as viewed by the MPU and other masters. The HPS has multiple address spaces, defined in the following section.

HPS Address Spaces

The following table shows the HPS address spaces and their sizes.

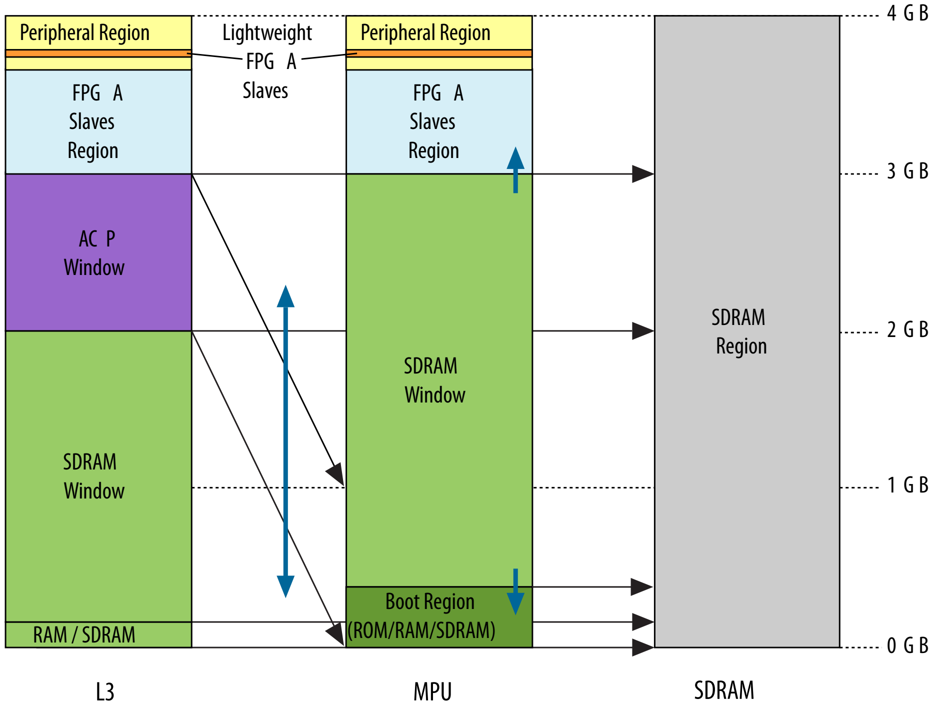
Table 1-1: HPS Address Spaces

Name	Description	Size
MPU	MPU subsystem	4 GB
L3	L3 interconnect	4 GB
SDRAM	SDRAM controller subsystem	4 GB

Address spaces are divided into one or more nonoverlapping regions. For example, the MPU address space has the peripheral, FPGA slaves, SDRAM window, and boot regions.

The following figure shows the relationships between the HPS address spaces. The figure is not to scale.

Figure 1-3: HPS Address Space Relationships



The window regions provide access to other address spaces. The thin black arrows indicate which address space is accessed by a window region (arrows point to accessed address space). For example, accesses to the ACP window in the L3 address space map to a 1 GB region of the MPU address space.

The SDRAM window in the MPU address space can grow and shrink at the top and bottom (short, blue vertical arrows) at the expense of the FPGA slaves and boot regions. For specific details, refer to “MPU Address Space”.

The ACP window can be mapped to any 1 GB region in the MPU address space (blue vertical bidirectional arrow), on gigabyte-aligned boundaries.

The following table shows the base address and size of each region that is common to the L3 and MPU address spaces.

Table 1-2: Common Address Space Regions

Identifier	Region Name	Base Address	Size
FPGASLAVES	FPGA slaves	0xC0000000	960 MB
PERIPH	Peripheral	0xFC000000	64 MB
LWFPGASLAVES [1]	Lightweight FPGA slaves	0xFF200000	2 MB

[1] This space is part of the "PERIPH" space.

Related Information

MPU Address Space on page 1-15

SDRAM Address Space

The SDRAM address space is up to 4 GB. The entire address space can be accessed through the FPGA-to-HPS SDRAM interface from the FPGA fabric. The total amount of SDRAM addressable from the other address spaces can be configured at runtime.

Related Information

- MPU Address Space on page 1-15
- L3 Address Space on page 1-15
- Interconnect on page 1-4
  For details of how to configure the SDRAM address space.

MPU Address Space

The MPU address space is 4 GB and applies to addresses generated inside the MPU. The MPU address space contains the following regions:

- The SDRAM window region provides access to a large, configurable portion of the 4 GB SDRAM address space.
- The MPU L2 cache controller contains a master connected to the L3 interconnect and a master connected to the SDRAM.
- The address filtering start and end registers in the L2 cache controller define the SDRAM window boundaries.
  - The boundaries are megabyte-aligned.
  - Addresses within the boundaries route to the SDRAM master.
  - Addresses outside the boundaries route to the L3 interconnect master.

Related Information

- HPS Address Spaces on page 1-13
  For more information, refer to the "HPS Address Space Relationships" table.
- System Interconnect on page 7-1
  For information about the L3 `remap` control register bits, refer to the *Interconnect* chapter.
- Cortex-A9 Microprocessor Unit Subsystem on page 9-1
  For details about L2 address filtering, refer to the *Cortex-A9 Microprocessor Unit Subsystem* chapter.

L3 Address Space

The L3 address space is 4 GB and applies to all L3 masters except the MPU. The L3 address space has more configuration options than the other address spaces.

Related Information

- System Interconnect on page 7-1
  For information about the L3 `remap` control register bits, refer to the *Interconnect* chapter.
- Cortex-A9 Microprocessor Unit Subsystem on page 9-1
  For more information about the ACP ID mapper, refer to the Cortex-A9 Microprocessor Unit Subsystem chapter.

HPS Peripheral Region Address Map

Table 1-5 lists the slave identifier, slave title, base address, and size of each slave in the peripheral region. The Slave Identifier column lists the names used in the HPS register map.

Table 1-5: Peripheral Region Address Map

Slave Identifier	Slave Title	Base Address	Size
STM	STM	0xFC000000	48 MB
DAP	DAP	0xFF000000	2 MB
LWFPGASLAVES	FPGA slaves accessed with lightweight FPGA-to-HPS bridge	0xFF200000	2 MB
LWHPS2FPGAREGS	Lightweight FPGA-to-HPS bridge GPV	0xFF400000	1 MB
HPS2FPGAREGS	HPS-to-FPGA bridge GPV	0xFF500000	1 MB
FPGA2HPSREGS	FPGA-to-HPS bridge GPV	0xFF600000	1 MB
EMAC0	EMAC0	0xFF700000	8 KB
EMAC1	EMAC1	0xFF702000	8 KB
SDMMC	SD/MMC	0xFF704000	4 KB
QSPIREGS	Quad SPI flash controller registers	0xFF705000	4 KB
FPGAMGRREGS	FPGA manager registers	0xFF706000	4 KB
ACPIDMAP	ACP ID mapper registers	0xFF707000	4 KB
GPIO0	GPIO0	0xFF708000	4 KB
GPIO1	GPIO1	0xFF709000	4 KB
GPIO2	GPIO2	0xFF70A000	4 KB
L3REGS	L3 interconnect GPV	0xFF800000	1 MB
NANDDATA	NAND controller data	0xFF900000	1 MB
QSPIDATA	Quad SPI flash data	0xFFA00000	1 MB

Slave Identifier	Slave Title	Base Address	Size
USB0	USB0 OTG controller registers	0xFFB00000	256 KB
USB1	USB1 OTG controller registers	0xFFB40000	256 KB
NANDREGS	NAND controller registers	0xFFB80000	64 KB
FPGAMGRDATA	FPGA manager configuration data	0xFFB90000	4 KB
CAN0	CAN0 controller registers	0xFFC00000	4 KB
CAN1	CAN1 controller registers	0xFFC01000	4 KB
UART0	UART0	0xFFC02000	4 KB
UART1	UART1	0xFFC03000	4 KB
I2C0	I2C0	0xFFC04000	4 KB
I2C1	I2C1	0xFFC05000	4 KB
I2C2	I2C2	0xFFC06000	4 KB
I2C3	I2C3	0xFFC07000	4 KB
SPTIMER0	SP Timer0	0xFFC08000	4 KB
SPTIMER1	SP Timer1	0xFFC09000	4 KB
SDRREGS	SDRAM controller subsystem registers	0xFFC20000	128 KB
OSC1TIMER0	OSC1 Timer0	0xFFD00000	4 KB
OSC1TIMER1	OSC1 Timer1	0xFFD01000	4 KB
L4WD0	Watchdog0	0xFFD02000	4 KB
L4WD1	Watchdog1	0xFFD03000	4 KB
CLKMGR	Clock manager	0xFFD04000	4 KB
RSTMGR	Reset manager	0xFFD05000	4 KB
SYSMGR	System manager	0xFFD08000	16 KB

Slave Identifier	Slave Title	Base Address	Size
DMANONSECURE	DMA nonsecure registers	0xFFE00000	4 KB
DMASECURE	DMA secure registers	0xFFE01000	4 KB
SPIS0	SPI slave0	0xFFE02000	4 KB
SPIS1	SPI slave1	0xFFE03000	4 KB
SPIM0	SPI master0	0xFFF00000	4 KB
SPIM1	SPI master1	0xFFF01000	4 KB
SCANMGR	Scan manager registers	0xFFF02000	4 KB
ROM	Boot ROM	0xFFFD0000	64 KB
MPUSCU	MPU SCU registers	0xFFFEC000	8 KB
MPUL2	MPU L2 cache controller registers	0xFFFEF000	4 KB
OCRAM	On-chip RAM	0xFFFF0000	64 KB

Related Information

Cyclone V SoC HPS Address Map and Register Definitions

Document Revision History

Table 1-6: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Maintenance release
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.3	Minor updates.
June 2012	1.2	Updated address spaces section.
May 2012	1.1	Added peripheral region address map.
January 2012	1.0	Initial release.

Clock Manager 2

2014.06.30

cv_54002 Subscribe Send Feedback

The hard processor system (HPS) clock generation is centralized in the *Clock Manager*. The *Clock Manager* is responsible for providing software-programmable clock control to configure all clocks generated in the HPS. Clocks are organized in clock groups. A clock group is a set of clock signals that originate from the same clock source. A phase-locked loop (PLL) clock group is a clock group where the clock source is a common PLL voltage-controlled oscillator (VCO).

Features of the Clock Manager

The *Clock Manager* offers the following features:

- Generates and manages clocks in the HPS
- Contains the following PLL clock groups:
  - Main—contains clocks for the Cortex-A9 microprocessor unit (MPU) subsystem, level 3 (L3) interconnect, level 4 (L4) peripheral bus, and debug
  - Peripheral—contains clocks for PLL-driven peripherals
  - SDRAM—contains clocks for the SDRAM subsystem
- Allows scaling of the MPU subsystem clocks without disabling peripheral and SDRAM clock groups
- Generates clock gate controls for enabling and disabling most clocks
- Initializes and sequences clocks for the following events:
  - Cold reset
  - Safe mode request from reset manager on warm reset

- Allows software to program clock characteristics, such as the following items discussed later in this chapter:
    - Input clock source for SDRAM and peripheral PLLs
    - Multiplier range, divider range, and six post-scale counters for each PLL
    - Output phases for SDRAM PLL outputs
    - VCO enable for each PLL
    - Bypass modes for each PLL
    - Gate off individual clocks in all PLL clock groups
    - Clear loss of lock status for each PLL
    - Safe mode for Hardware-Managed clocks
    - General-purpose I/O (GPIO) debounce clock divide
- Allows software to observe the status of all writable registers
- Supports interrupting the MPU subsystem on PLL-lock and loss-of-lock
- Supports clock gating at the signal level

The *Clock Manager* is **not** responsible for the following functional behaviors:

- Selection or management of the clocks for the FPGA-to-HPS, HPS-to-FPGA, and FPGA-to-HPS SDRAM interfaces. The FPGA logic designer is responsible for selecting and managing these clocks.
- Software must not program the *Clock Manager* with illegal values. If it does, the behavior of the clock manager is undefined and could stop the operation of the HPS. The only guaranteed means for recovery from an illegal clock setting is a cold reset.

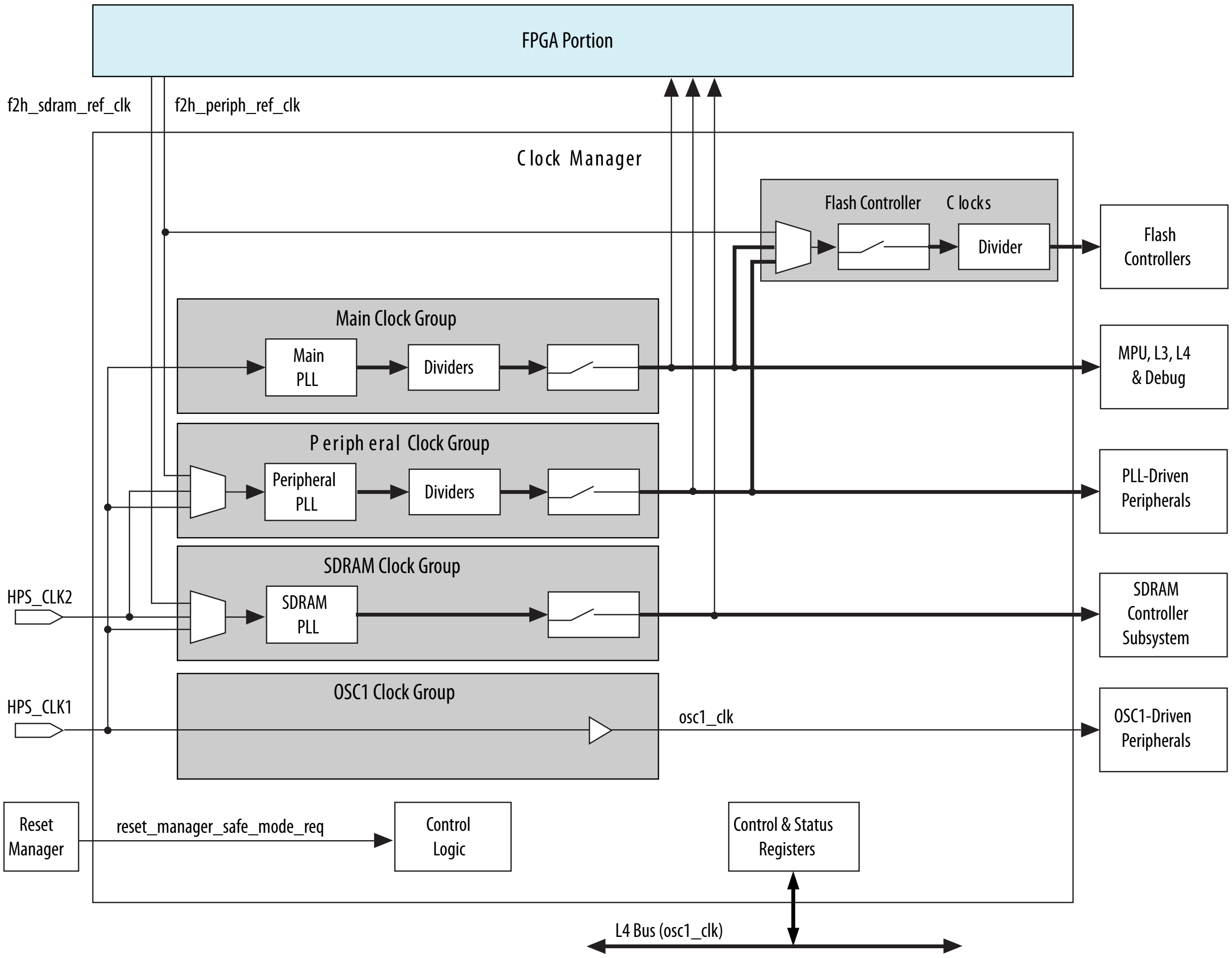
Related Information

Hardware-Managed and Software-Managed Clocks on page 2-5
When re-programming clock settings, there are no automatic glitch-free clock transitions. Software must follow a specific sequence to ensure glitch-free clock transitions. Refer to *Hardware-Managed and Software-Managed Clocks* section of this chapter.

Clock Manager Block Diagram and System Integration

Figure 2-1: Clock Manager Block Diagram

The following figure shows the major components of the clock manager and its integration in the HPS.



Functional Description of the Clock Manager

Clock Manager Building Blocks

PLLs

The *Clock Manager* contains three PLLs: main, peripherals, and SDRAM. These PLLs generate the majority of clocks in the HPS. There is no phase control between the clocks generated by the three PLLs.

Each PLL has the following features:

- Phase detector and output lock signal generation
- Registers to set VCO frequency
  - Multiplier range is 1 to 4096
  - Divider range is 1 to 64
- Six post-scale counters (C0-C5) with a range of 1 to 512
- PLL can be enabled to bypass all outputs to the `osc1_clk` clock for glitch-free transitions

The SDRAM PLL has the following additional feature:

- Phase shift of 1/8 per step
  - Phase shift range is 0 to 7

FREF, FVCO, and FOUT Equations

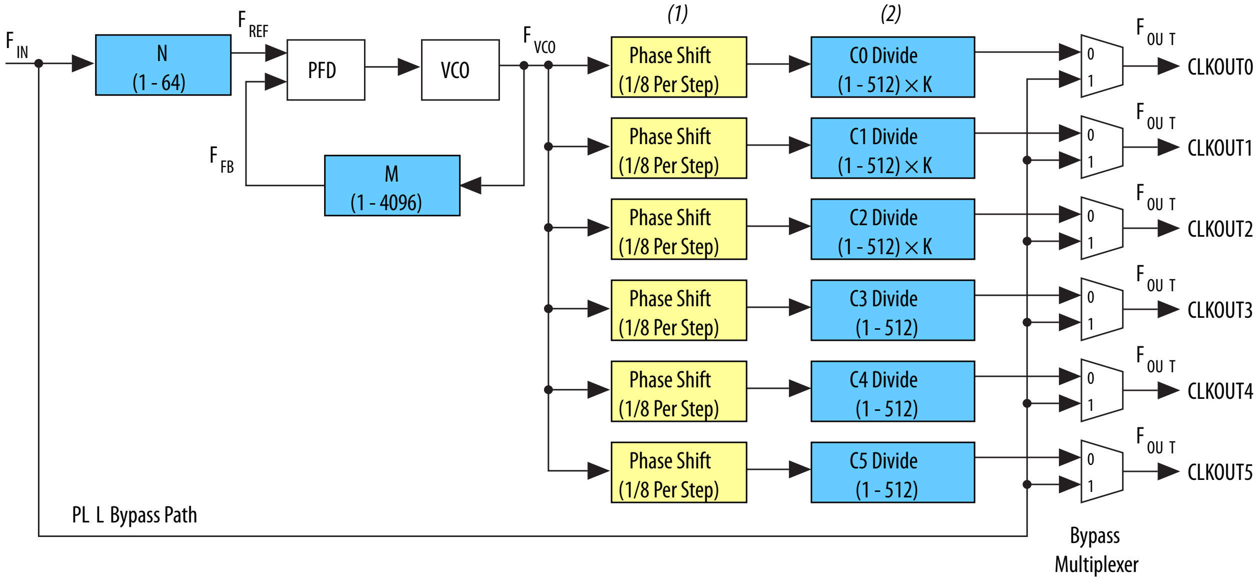
$$
\begin{aligned}
FREF &= F_{IN} / N \\
FVCO &= F_{REF} \times M = F_{IN} \times M/N \\
FOUT &= F_{VCO} / (C_i \times K) = F_{REF} \times M/ (C_i \times K) = (F_{IN} \times M)/ (N \times C_i \times K)
\end{aligned}
$$

where:

- FVCO = VCO frequency.
- FIN = input frequency.
- FREF = reference frequency.
- M = numerator, part of the clock feedback path.
- N = denominator, part of the input clock reference path.
- Ci = post-scale counter, where *i* is 0-5 for each of the six counters.
- K is an internal post-scale counter in the main PLL, where the reset values are K = 2 for C0, and K = 4 for C1 and C2.

Figure 2-2: PLL Block Diagram

Values listed for M, N, and C are actually one greater than the values stored in the CSRs.



Related Information

Cyclone V Device Datasheet

Minimum and maximum VCO frequencies for the main, peripheral, and SDRAM PLLs vary by device speed grade. For specific details, refer the Cyclone V Device Datasheet.

Dividers

Dividers subdivide the C0-C15 clocks produced by the PLL to lower frequencies. The main PLL C0-C2 clocks have an additional internal post-scale counter.

Clock Gating

Clock gating enables and disables clock signals.

Control and Status Registers

The *Clock Manager* contains registers used to configure and observe the clock manager.

Hardware-Managed and Software-Managed Clocks

When changing values on clocks, the terms *Hardware-Managed* and *Software-Managed* define who is responsible for successful transitions. Software-Managed clocks require that software manually gate off any clock affected by the change, wait for any PLL lock if required, then gate the clocks back on. Hardware-Managed clocks use hardware to ensure that a glitch-free transition to a new clock value occurs. There are three Hardware-Managed sets of clocks in the HPS, namely, clocks generated from the main PLL outputs C0, C1, and C2. All other clocks in the HPS are Software-Managed clocks.

Clock Groups

The clock manager contains one clock group for each PLL and one clock group for the `HPS_CLK1` pin.

OSC1 Clock Group

The clock in the OSC1 clock group is derived directly from the `HPS_CLK1` pin. This clock is never gated or divided.

It is used as a PLL input and also by HPS logic that does not operate on a clock output from a PLL.

Table 2-1: OSC1 Clock Group Clock

Name	Frequency	Clock Source	Destination
`osc1_clk`	10 to 50 MHz	`HPS_CLK1` pin	OSC1-driven peripherals listed in Table 2–9 on page 2–14

Main Clock Group

The main clock group consists of a PLL, dividers, and clock gating. The clocks in the main clock group are derived from the main PLL. The main PLL is always sourced from the `HPS_CLK1` pin of the device.

Table 2-2: Main PLL Output Assignments

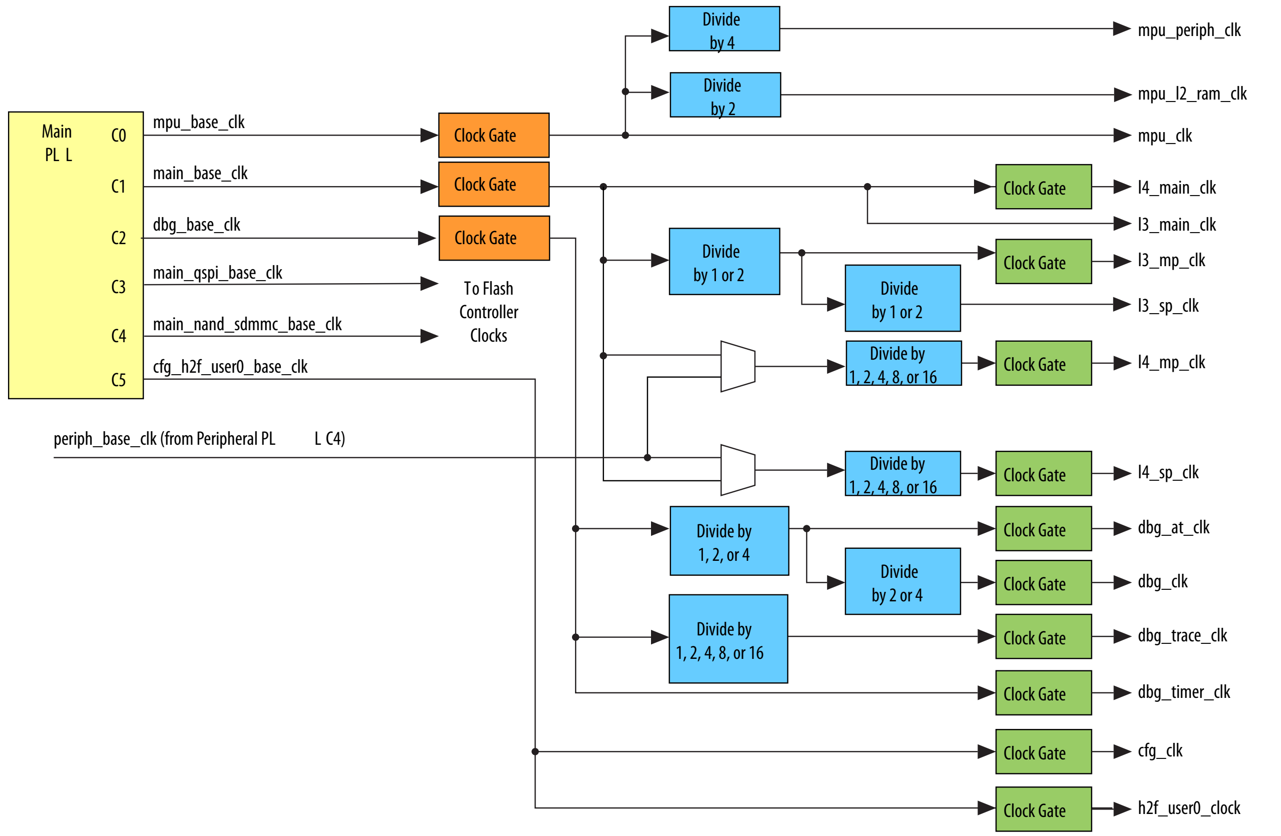
PLL	Output Counter	Clock Name	Frequency	Phase Shift Control
Main	C0	mpu_base_clk	osc1_clk to varies [2]	No
	C1	main_base_clk	osc1_clk to varies [2]	No
	C2	dbg_base_clk	osc1_clk/4 to mpu_base_clk/2	No
	C3	main_qspi_base_clk	Up to 432 MHz	No
	C4	main_nand_sdmmc_base_clk	Up to 250 MHz for the NAND flash controller and up to 200 MHz for the SD/MMC controller	No
	C5	cfg_h2f_user0_base_clk	osc1_clk to 125 MHz for driving configuration and 100 MHz for the user clock	No

The counter outputs from the main PLL can have their frequency further divided by programmable dividers external to the PLL. Transitions to a different divide value occur on the fastest output clock, one clock cycle prior to the slowest clock's rising edge. For example, cycle 15 of the divide-by-16 divider for the main C2 output and cycle 3 of the divide-by-4 divider for the main C0 output.

The following figure shows how each counter output from the main PLL can have its frequency further divided by programmable post-PLL dividers. Green-colored clock gating logic is directly controlled by software writing to a register. Orange-colored clock gating logic is controlled by hardware. Orange-colored clock gating logic allows hardware to seamlessly transition a synchronous set of clocks, for example, all the MPU subsystem clocks.

[2] The maximum frequency depends on the speed grade of the device.

Figure 2-3: Main Clock Group Divide and Gating



The clocks derived from main PLL C0-C2 outputs are hardware-managed, meaning hardware ensures that a clean transition occurs, and can have the following control values changed dynamically by software write accesses to the control registers:

- PLL bypass
- PLL numerator, denominator, and counters
- External dividers

For these registers, hardware detects that the write has occurred and performs the correct sequence to ensure that a glitch-free transition to the new clock value occurs. These clocks can pause during the transition.

Table 2-3: Main Clock Group Clocks

System Clock Name	Freq uency	Constraints and Notes
`mpu_clk`	Main PLL C0	Clock for MPU subsystem, including CPU0 and CPU1
`mpu_l2_ram_clk`	`mpu_clk/2`	Clock for MPU level 2 (L2) RAM

System Clock Name	Freq uency	Constraints and Notes
mpu_periph_clk	mpu_clk/4	Clock for MPU snoop control unit (SCU) peripherals , such as the general interrupt controller (GIC)
l3_main_clk	Main PLL C1	Clock for L3 main switch
l3_mp_clk	l3_main_clk or l3_main_clk/2	Clock for L3 master peripherals (MP) switch
l3_sp_clk	l3_mp_clk or l3_mp_clk/2	Clock for L3 slave peripherals (SP) switch
l4_main_clk	Main PLL C1	Clock for L4 main bus
l4_mp_clk	osc1_clk/16 to 100 MHz divided from main PLL C1 or peripheral PLL C4	Clock for L4 MP bus
l4_sp_clk	osc1_clk/16 to 100 MHz divided from main PLL C1 or peripheral PLL C4	Clock for L4 SP bus
dbg_at_clk	osc1_clk/4 to main PLL C2/2	Clock for CoreSight™ debug trace bus
dbg_trace_clk	osc1_clk/16 to main PLL C2	Clock for CoreSight™ debug Trace Port Interface Unit (TPIU)
dbg_timer_clk	osc1_clk to main PLL C2	Clock for the trace timestamp generator
dbg_clk	dbg_at_clk/2 or dbg_at_clk/4	Clock for Debug Access Port (DAP) and debug peripheral bus
main_qspi_clk	Main PLL C3	Quad SPI flash internal logic clock
main_nand_sdmmc_clk	Main PLL C4	Input clock to flash controller clocks block

System Clock Name	Freq uency	Constraints and Notes
cfg_clk	osc1_clk to 125_MHz divided from main PLL C5	FPGA manager configuration clock
h2f_user0_clock	osc1_clk to 100_MHz divided from main PLL C5	Auxiliary user clock to the FPGA fabric

Changing Values That Affect Main Clock Group PLL Lock

To change any value that affects VCO lock of the main clock group PLL, including the hardware-managed clocks, software must put the main PLL in bypass mode, which causes all the main PLL output clocks to be driven by the osc1_clk clock. Software must detect PLL lock by reading the lock status register prior to taking the main PLL out of bypass mode.

Once a PLL is locked, changes to any PLL VCO frequency that are 20 percent or less do not cause the PLL to lose lock. Iteratively changing the VCO frequency in increments of 20 percent or less allow a slow ramp of the VCO base frequency without loss of lock.For example, to change a VCO frequency by 40% without losing lock, change the frequency by 20%, then change it again by 16.7%.

Peripheral Clock Group

The peripheral clock group consists of a PLL, dividers, and clock gating. The clocks in the peripheral clock group are derived from the peripheral PLL. The peripheral PLL can be programmed to be sourced from the HPS_CLK1 pin, the HPS_CLK2 pin, or the f2h_periph_ref_clk clock provided by the FPGA fabric.

The counter outputs from the main PLL can have their frequency further divided by external dividers. Transitions to a different divide value occur on the fastest output clock, one clock cycle prior to the slowest clock’s rising edge. For example, cycle 15 of the divide-by-16 divider for the main C2 output and cycle 3 of the divide-by-4 divider for the C1 output.

Table 2-4: Peripheral PLL Output Assignments

PLL	Output Counter	Clock Name	Frequency	Phase Shift Control
Peripheral	C0	emac0_base_clk	Up to 250 MHz	No
	C1	emac1_base_clk	Up to 250 MHz	No
	C2	periph_qspi_base_clk	Up to 432 MHz	No
	C3	periph_nand_sdmmc_base_clk	Up to 250 MHz for the NAND flash controller and up to 200 MHz for the SD/MMC controller	No
	C4	periph_base_base_clk	Up to 240 MHz for the SPI masters and up to 200 MHz for the scan manager	No
	C5	h2f_user1_base_clk	osc1_clk to 100 MHz	No

The following figure shows programmable post-PLL dividers and clock gating for the peripheral clock group. Clock gate blocks in the diagram indicate clocks which may be gated off under software control. Software is expected to gate these clocks off prior to changing any PLL or divider settings that might create incorrect behavior on these clocks.

Figure 2-4: Peripheral Clock Group Divide and Gating

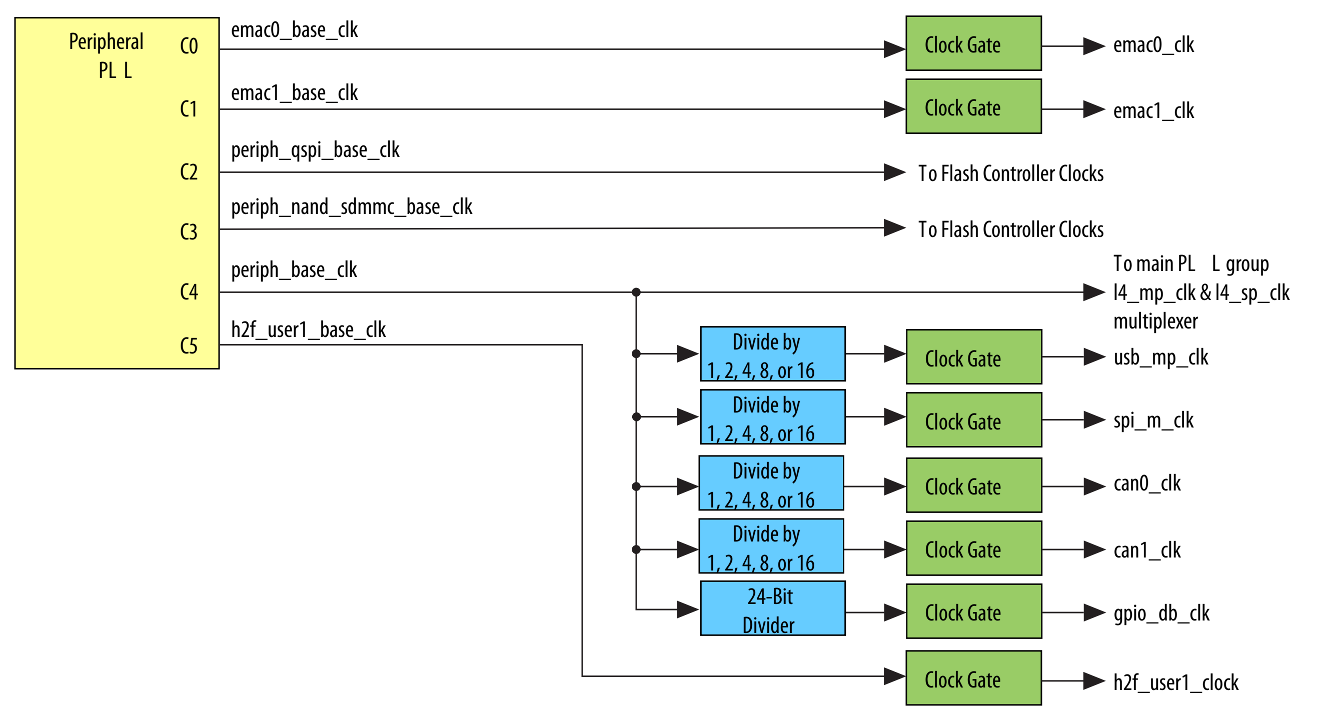


Table 2-5: Peripheral Clock Group Clocks

System Clock Name	Frequency	Divided From	Constraints and Notes
`usb_mp_clk`	Up to 200 MHz	Peripheral PLL C4	Clock for USB
`spi_m_clk`	Up to 240 MHz for the SPI masters and up to 200 MHz for the scan manager	Peripheral PLL C4	Clock for L4 SPI master bus and scan manager
`emac0_clk`	Up to 250 MHz	Peripheral PLL C0	EMAC0 clock. The 250 MHz clock is divided internally by the EMAC into the typical 125/25/2.5 MHz speeds for 1000/100/10 Mbps operation.

System Clock Name	Frequency	Divided From	Constraints and Notes
`emac1_clk`	Up to 250 MHz	Peripheral PLL C1	EMAC1 clock The 250 MHz clock is divided internally by the EMAC into the typical 125/25/2.5 MHz speeds for 1000/100/10 Mbps operation.
`l4_mp_clk`	Up to 100 MHz	Main PLL C1 or peripheral PLL C4	Clock for L4 master peripheral bus
`l4_sp_clk`	Up to 100 MHz	Main PLL C1 or peripheral PLL C4	Clock for L4 slave peripheral bus
`can0_clk`	Up to 100 MHz	Peripheral PLL C4	Controller area network (CAN) controller 0 clock
`can1_clk`	Up to 100 MHz	Peripheral PLL C4	CAN controller 1 clock
`gpio_db_clk`	Up to 1 MHz	Peripheral PLL C4	Used to debounce GPIO0, GPIO1, and GPIO2
`h2f_user1_clock`	Peripheral PLL C5	Peripheral PLL C5	Auxiliary user clock to the FPGA fabric

SDRAM Clock Group

The SDRAM clock group consists of a PLL and clock gating. The clocks in the SDRAM clock group are derived from the SDRAM PLL. The SDRAM PLL can be programmed to be sourced from the `HPS_CLK1` pin, the `HPS_CLK2` pin, or the `f2h_sdram_ref_clk` clock provided by the FPGA fabric.

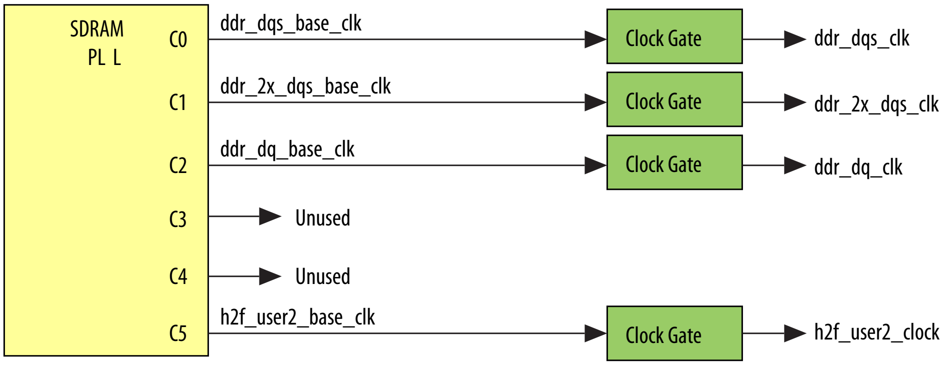
The counter outputs from the SDRAM PLL can be gated off directly under software control. The divider values for each clock are set by registers in the clock manager.

Table 2-6: SDRAM PLL Output Assignments

PLL	Output Counter	Clock Name	Frequency	Phase Shift Control
SDRAM	C0	ddr_dqs_base_clk	Varies [3]	Yes
	C1	ddr_2x_dqs_base_clk	ddr_dqs_base_clk x 2	Yes
	C2	ddr_dq_base_clk	ddr_dqs_base_clk	Yes
	C5	h2f_user2_base_clk	osc1_clk to varies [3]	Yes

The following figure shows clock gating for SDRAM PLL clock group. Clock gate blocks in the diagram indicate clocks which may be gated off under software control. Software is expected to gate these clocks off prior to changing any PLL or divider settings that might create incorrect behavior on these clocks.

Figure 2-5: SDRAM Clock Group Divide and Gating



The SDRAM PLL output clocks can be phase shifted in real time in increments of 1/8 the VCO frequency. Maximum number of phase shift increments is 4096.

Table 2-7: SDRAM Clock Group Clocks

Name	Frequency	Constraints and Notes
ddr_dqs_clk	SDRAM PLL C0	Clock for MPFE, single-port controller, CSR access, and PHY

[3] The maximum frequency depends on the speed grade of the device.

Name	Frequency	Constraints and Notes
ddr_2x_dqs_clk	SDRAM PLL C1	Clock for PHY
ddr_dq_clk	SDRAM PLL C2	Clock for PHY
h2f_user2_clock	SDRAM PLL C5	Auxiliary user clock to the FPGA fabric

Flash Controller Clocks

Flash memory peripherals can be driven by the main PLL, the peripheral PLL, or from clocks provided by the FPGA fabric.

Figure 2-6: Flash Peripheral Clock Divide and Gating

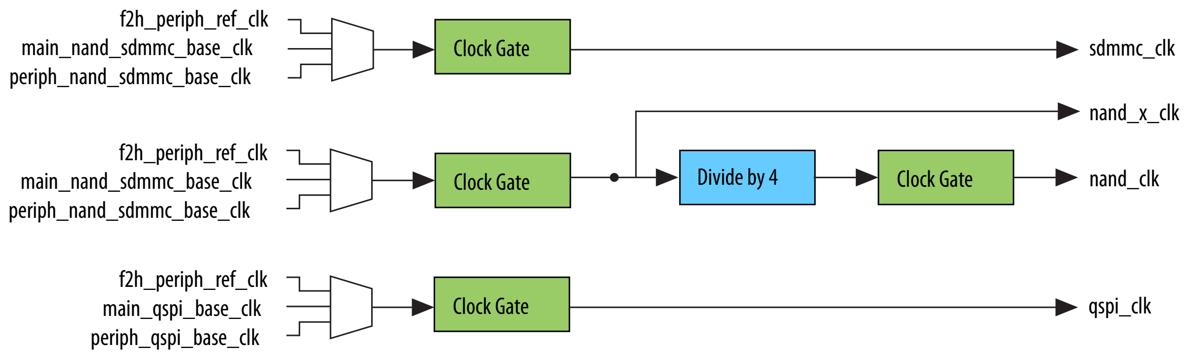


Table 2-8: Flash Controller Clocks

System Clock Name	Freq uency	Divided From	Constraints and Notes
qspi_clk	Up to 432 MHz	Peripheral PLL C2, main PLL C3, or f2h_periph_ref_clk	Clock for quad SPI, typically 108 and 80 MHz
nand_x_clk	Up to 250 MHz	Peripheral PLL C3, main PLL C4, or f2h_periph_ref_clk	NAND flash controller master and slave clock
nand_clk	nand_x_clk/4	Peripheral PLL C3, main PLL C4, or f2h_periph_ref_clk	Main clock for NAND flash controller, sets base frequency for NAND transactions

System Clock Name	Freq uency	Divided From	Constraints and Notes
`sdmmc_clk`	Up to 200 MHz	Peripheral PLL C3, main PLL C4, or `f2h_periph_ref_clk`	• Less than or equal to memory maximum operating frequency • 45% to 55% duty cycle • Typical frequencies are 26 and 52 MHz • SD/MMC has a subclock tree divided down from this clock

Resets

Cold Reset

Cold reset places the hardware-managed clocks into safe mode, the software-managed clocks into their default state, and asynchronously resets all registers in the clock manager.

Related Information

Safe Mode on page 2-15

Warm Reset

Registers in the clock manager control how the clock manager responds to warm reset. Typically, software places the clock manager into a safe state in order to generate a known set of clocks for the ROM code to boot the system. The behavior of the system on warm reset as a whole, including how the FPGA fabric, boot code, and debug systems are configured to behave, must be carefully considered when choosing how the clock manager responds to warm reset.

The reset manager can request that the clock manager go into safe mode as part of the reset manager's warm reset sequence. Before asserting safe mode to the clock manager, the reset manager ensures that the reset signal is asserted on all modules that receive warm reset.

Related Information

Reset Manager on page 3-1
For more information, refer to "Reset Sequencing" in the *Reset Manager* chapter.

Safe Mode

Safe mode is enabled in the HPS by the assertion of a safe mode request from the reset manager or by a cold reset. Assertion of the safe mode request from the reset manager sets the safe mode bit in the clock manager control register. No other control register bits are affected by the safe mode request from the reset manager.

When safe mode is enabled, the main PLL hardware-managed clocks (C0-C2) are bypassed to the `osc1_clk` clock and are directly generated from the `osc1_clk` clock. While in safe mode, clock manager register settings, which control clock behavior, are not changed. However, the hardware bypasses these settings and uses safe, default settings.

The hardware-managed clocks are forced to their safe mode values such that the following conditions occur:

- The hardware-managed clocks are bypassed to the `osc1_clk` clock, including counters in the main PLL.
- Programmable dividers select the reset default values.
- The flash controller clocks multiplexer selects the output from the peripheral PLL.
- All clocks are enabled.

A write by software is the only way to clear the safe mode bit (`safemode`) of the `ctrl` register.

Note: Before coming out of safe mode, all registers and clocks need to be configured correctly. It is possible to program the clock manager in such a way that only a cold reset can return the clocks to a functioning state. Altera strongly recommends using Altera-provided libraries to configure and control HPS clocks.

Interrupts

The clock manger provides one interrupt output, enabled using the interrupt enable register (`intren`). The source of the interrupt is six inputs, namely, an achieving lock and a losing lock bit in the interrupt status register (`inter`) for each PLL.

Clock Usage By Module

The following table lists every clock input generated by the clock manager to all modules in the HPS. System clock names are global for the entire HPS and system clocks with the same name are phase-aligned at all endpoints.

Table 2-9: Clock Usage By Module

Module Name	System Clock Name	Use
MPU subsystem	`mpu_clk`	Main clock for the MPU subsystem
	`mpu_periph_clk`	Clock for peripherals inside the MPU subsystem
	`dbg_at_clk`	Trace bus clock
	`dbg_clk`	Debug clock
	`mpu_l2_ram_clk`	Clock for the L2 cache and Accelerator Coherency Port (ACP) ID mapper
	`l4_mp_clk`	Clock for the ACP ID mapper control slave

Module Name	System Clock Name	Use
Interconnect	l3_main_clk	Clock for the L3 main switch
	dbg_at_clk	Clock for the System Trace Macrocell (STM) slave and Embedded Trace Router (ETR) master connections
	dbg_clk	Clock for the DAP master connection
	l3_mp_clk	Clock for the L3 master peripheral switch
	l4_mp_clk	Clock for the L4 MP bus, Secure Digital (SD) / MultiMediaCard (MMC) master, and EMAC masters
	usb_mp_clk	Clock for the USB masters and slaves
	nand_x_clk	Clock for the NAND master
	cfg_clk	Clock for the FPGA manager configuration data slave
	l3_sp_clk	Clock for the L3 slave peripheral switch
	l3_main_clk	Clock for the L4 SPIS bus master
	mpu_l2_ram_clk	Clock for the ACP ID mapper slave and L2 master connections
	osc1_clk	Clock for the L4 OSC1 bus master
	spi_m_clk	Clock for the L4 SPIM bus master
	l4_sp_clk	Clock for the L4 SP bus master
	l4_mp_clk	Clock for the quad SPI bus slave
Boot ROM	l3_main_clk	Clock for the boot ROM

Module Name	System Clock Name	Use
On-chip RAM	`l3_main_clk`	Clock for the on-chip RAM
DMA controller	`l4_main_clk`	Clock for the DMA
	`dbg_at_clk`	Clock synchronous to the STM module
	`l4_mp_clk`	Clock synchronous to the quad SPI flash
FPGA manager	`cfg_clk`	Clock for the control block (CB) data interface and configuration data slave
	`l4_mp_clk`	Clock for the control slave
HPS-to-FPGA bridge	`l3_main_clk`	Clock for the data slave
	`l4_mp_clk`	Clock for the global programmer's view (GPV) slave
FPGA-to-HPS bridge	`l3_main_clk`	Clock for the data master
	`l4_mp_clk`	Clock for the GPV slave
Lightweight HPS-to-FPGA bridge	`l4_mp_clk`	Clock for the GPV masters, and the data and GPV slave
Quad SPI flash controller	`l4_mp_clk`	Clock for the control slave
	`qspi_clk`	Reference clock for serialization
SD/MMC controller	`l4_mp_clk`	Clock for the master and slave
	`sdmmc_clk`	Clock for the SD/MMC internal logic

Module Name	System Clock Name	Use
EMAC 0	`l4_mp_clk`	Clock for the master
	`emac0_clk`	EMAC 0 internal logic clock
	`osc1_clk`	IEEE 1588 timestamp clock
EMAC 1	`l4_mp_clk`	Clock for the master
	`emac1_clk`	EMAC 1 internal logic clock
	`osc1_clk`	IEEE 1588 timestamp clock
USB 0	`usb_mp_clk`	Clock for the master and slave
USB 1	`usb_mp_clk`	Clock for the master and slave
NAND flash controller	`nand_x_clk`	NAND high-speed master and slave clock
	`nand_clk`	NAND flash clock
OSC1 timer 0	`osc1_clk`	Clock for the OSC1 timer 0
OSC1 timer 1	`osc1_clk`	Clock for the OSC1 timer 1
SP timer 0	`l4_sp_clk`	Clock for the SP timer 0
SP timer 1	`l4_sp_clk`	Clock for the SP timer 1
I^{2}C controller 0	`l4_sp_clk`	Clock for the I^{2}C 0
I^{2}C controller 1	`l4_sp_clk`	Clock for the I^{2}C 1
I^{2}C controller 2	`l4_sp_clk`	Clock for the I^{2}C 2
I^{2}C controller 3	`l4_sp_clk`	Clock for the I^{2}C 3

Module Name	System Clock Name	Use
UART controller 0	`l4_sp_clk`	Clock for the UART 0
UART controller 1	`l4_sp_clk`	Clock for the UART 1
CAN controller 0	`l4_sp_clk`	Clock for the slave
	`can0_clk`	CAN 0 controller clock
CAN controller 1	`l4_sp_clk`	Clock for the slave
	`can1_clk`	CAN 1 controller clock
GPIO interface 0	`l4_mp_clk`	Clock for the slave
	`gpio_db_clk`	Debounce clock
GPIO interface 1	`l4_mp_clk`	Clock for the slave
	`gpio_db_clk`	Debounce clock
GPIO interface 2	`l4_mp_clk`	Clock for the slave
	`gpio_db_clk`	Debounce clock
System manager	`osc1_clk`	Clock for the system manager

Module Name	System Clock Name	Use
SDRAM subsystem	`l4_sp_clk`	Clock for the control slave
	`ddr_dq_clk`	Off-chip data clock
	`ddr_dqs_clk`	Clock for the MPFE, single-port controller, CSRs, and PHY
	`ddr_2x_dqs_clk`	Off-chip strobe data clock
	`mpu_l2_ram_clk`	Clock for the slave connected to MPU subsystem L2 cache
	`l3_main_clk`	Clock for the slave connected to L3 interconnect
L4 watchdog timer 0	`osc1_clk`	Clock for the L4 watchdog timer 0
L4 watchdog timer 1	`osc1_clk`	Clock for the L4 watchdog timer 1
SPI master controller 0	`spi_m_clk`	Clock for the SPI master 0
SPI master controller 1	`spi_m_clk`	Clock for the SPI master 1
SPI slave controller 0	`l4_main_clk`	Clock for the SPI slave 0
SPI slave controller 1	`l4_main_clk`	Clock for the SPI slave 1
Debug subsystem	`l4_mp_clk`	System bus clock
	`dbg_clk`	Debug clock
	`dbg_at_clk`	Trace bus clock
	`dbg_trace_clk`	Trace port clock
Reset manager	`osc1_clk`	Clock for the reset manager
	`l4_sp_clk`	Clock for the slave

Module Name	System Clock Name	Use
Scan manager	`spi_m_clk`	Clock for the scan manager
Timestamp generator	`dbg_timer_clk`	Clock for the timestamp generator

Clock Manager Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridge consist of the following regions:

- Clock Manager Module

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

Clock Manager Module Address Map

Registers in the Clock Manager module

Base Address: `0xFFD04000`

Clock Manager Module

Register	Offset	Width	Access	Reset Value	Description
ctrl on page 2-24	0x0	32	RW	0x5	Control Register
bypass on page 2-25	0x4	32	RW	0xB	PLL Bypass Register
inter on page 2-27	0x8	32	RW	0x0	Interrupt Status Register
intren on page 2-28	0xC	32	RW	0x0	Interrupt Enable Register
dbctrl on page 2-29	0x10	32	RW	0x3	Debug clock Control Register
stat on page 2-30	0x14	32	RO	0x0	Status Register

Main PLL Group

Register	Offset	Width	Access	Reset Value	Description
vco on page 2-32	0x40	32	RW	0x8001000D	Main PLL VCO Control Register
misc on page 2-34	0x44	32	RW	0x4002	Main PLL VCO Advanced Control Register
mpuclk on page 2-35	0x48	32	RW	0x0	Main PLL C0 Control Register for Clock mpu_clk

Register	Offset	Width	Access	Reset Value	Description
mainclk on page 2-35	0x4C	32	RW	0x0	Main PLL C1 Control Register for Clock main_clk
dbgatclk on page 2-36	0x50	32	RW	0x0	Main PLL C2 Control Register for Clock dbg_base_clk
mainqspiclk on page 2-36	0x54	32	RW	0x3	Main PLL C3 Control Register for Clock main_qspi_clk
mainnandsdmmcclk on page 2-37	0x58	32	RW	0x3	Main PLL C4 Control Register for Clock main_nand_sdmmc_clk
cfgs2fuser0clk on page 2-37	0x5C	32	RW	0xF	Main PLL C5 Control Register for Clock cfg_s2f_user0_clk
en on page 2-38	0x60	32	RW	0x3FF	Enable Register
maindiv on page 2-39	0x64	32	RW	0x0	Main Divide Register
dbgdiv on page 2-40	0x68	32	RW	0x4	Debug Divide Register
tracediv on page 2-41	0x6C	32	RW	0x0	Debug Trace Divide Register
l4src on page 2-42	0x70	32	RW	0x0	L4 MP SP APB Clock Source
stat on page 2-43	0x74	32	RO	0x0	Main PLL Output Counter Reset Ack Status Register

Peripheral PLL Group

Register	Offset	Width	Access	Reset Value	Description
vco on page 2-45	0x80	32	RW	0x8001000D	Peripheral PLL VCO Control Register
misc on page 2-47	0x84	32	RW	0x4002	Peripheral PLL VCO Advanced Control Register
emac0clk on page 2-47	0x88	32	RW	0x1	Peripheral PLL C0 Control Register for Clock emac0_clk
emac1clk on page 2-48	0x8C	32	RW	0x1	Peripheral PLL C1 Control Register for Clock emac1_clk
perqspiclk on page 2-48	0x90	32	RW	0x1	Peripheral PLL C2 Control Register for Clock periph_qspi_clk
pernandsdmmcclk on page 2-49	0x94	32	RW	0x1	Peripheral PLL C3 Control Register for Clock periph_nand_sdmmc_clk
perbaseclk on page 2-49	0x98	32	RW	0x1	Peripheral PLL C4 Control Register for Clock periph_base_clk
s2fuser1clk on page 2-50	0x9C	32	RW	0x1	Peripheral PLL C5 Control Register for Clock s2f_user1_clk
en on page 2-50	0xA0	32	RW	0xFFF	Enable Register

Register	Offset	Width	Access	Reset Value	Description
div on page 2-52	0xA4	32	RW	0x0	Divide Register
gpiodiv on page 2-54	0xA8	32	RW	0x1	GPIO Divide Register
src on page 2-54	0xAC	32	RW	0x15	Flash Clock Source Register
stat on page 2-56	0xB0	32	RO	0x0	Peripheral PLL Output Counter Reset Ack Status Register

SDRAM PLL Group

Register	Offset	Width	Access	Reset Value	Description
vco on page 2-57	0xC0	32	RW	0x8001000D	SDRAM PLL VCO Control Register
ctrl on page 2-59	0xC4	32	RW	0x4002	SDRAM PLL VCO Advanced Control Register
ddrdqsclk on page 2-60	0xC8	32	RW	0x1	SDRAM PLL C0 Control Register for Clock ddr_dqs_clk
ddr2xdqsclk on page 2-61	0xCC	32	RW	0x1	SDRAM PLL C1 Control Register for Clock ddr_2x_dqs_clk
ddrdqclk on page 2-61	0xD0	32	RW	0x1	SDRAM PLL C2 Control Register for Clock ddr_dq_clk
s2fuser2clk on page 2-62	0xD4	32	RW	0x1	SDRAM PLL C5 Control Register for Clock s2f_user2_clk
en on page 2-63	0xD8	32	RW	0xF	Enable Register
stat on page 2-64	0xDC	32	RO	0x0	SDRAM PLL Output Counter Reset Ack Status Register

ctrl

Contains fields that control the entire Clock Manager.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													ensfmdwr RW 0x1	Reserved	safemode RW 0x1

ctrl Fields

Bit	Name	Description	Access	Reset
2	ensfmdwr	When set the Clock Manager will respond to a Safe Mode request from the Reset Manager on a warm reset by setting the Safe Mode bit. When clear the clock manager will not set the the Safe Mode bit on a warm reset This bit is cleared on a cold reset. Warm reset has no affect on this bit.	RW	0x1
0	safemode	When set the Clock Manager is in Safe Mode. In Safe Mode Clock Manager register settings defining clock behavior are ignored and clocks are set to a Safe Mode state.In Safe Mode all clocks with the optional exception of debug clocks, are directly generated from the EOSC1 clock input, all PLLs are bypassed, all programmable dividers are set to 1 and all clocks are enabled. This bit should only be cleared when clocks have been correctly configured This field is set on a cold reset and optionally on a warm reset and may not be set by SW.	RW	0x1

bypass

Contains fields that control bypassing each PLL.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											perpllsrc RW 0x0	perpll RW 0x1	sdrpllsrc RW 0x0	sdrpll RW 0x1	mainpll RW 0x1

bypass Fields

Bit	Name	Description	Access	Reset
4	perpllsrc	This bit defines the bypass source forPeripheral PLL. When changing fields that affect VCO lock the PLL must be bypassed and this bit must be set to OSC1_CLK. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit. **Value** **Description** 0x0 Select EOSC1 0x1 Select PLL Input Mux	RW	0x0
3	perpll	When set, causes the Peripheral PLL VCO and counters to be bypassed so that all clocks generated by the Peripheral PLL are directly driven from either eosc1_clk or the Peripheral PLL input clock. The bypass clock source for Peripheral PLL is determined by the Peripheral PLL Bypass Source Register bit. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit.	RW	0x1
2	sdrpllsrc	This bit defines the bypass source forSDRAM PLL. When changing fields that affect VCO lock the PLL must be bypassed and this bit must be set to OSC1_CLK. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit. **Value** **Description** 0x0 Select EOSC1 0x1 Select PLL Input Mux	RW	0x0

Bit	Name	Description	Access	Reset
1	sdrpll	When set, causes the SDRAM PLL VCO and counters to be bypassed so that all clocks generated by the SDRAM PLL are directly driven from either eosc1_clk or the SDRAM PLL input clock. The bypass clock source for SDRAM PLL is determined by the SDRAM PLL Bypass Source Register bit. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit.	RW	0x1
0	mainpll	When set, causes the Main PLL VCO and counters to be bypassed so that all clocks generated by the Main PLL are directly driven from the Main PLL input clock. The bypass source for Main PLL is the external eosc1_clk. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit.	RW	0x1

inter

Contains fields that indicate the PLL lock status. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							sdrplllocked RO 0x0	perplllocked RO 0x0	mainplllocked RO 0x0	sdrplllost RW 0x0	perplllost RW 0x0	mainplllost RW 0x0	sdrpllachieved RW 0x0	perpllachieved RW 0x0	mainpllachieved RW 0x0

inter Fields

Bit	Name	Description	Access	Reset
8	sdrplllocked	If 1, the SDRAM PLL is currently locked. If 0, the SDRAM PLL is currently not locked.	RO	0x0
7	perplllocked	If 1, the Peripheral PLL is currently locked. If 0, the Peripheral PLL is currently not locked.	RO	0x0

Bit	Name	Description	Access	Reset
6	mainplllocked	If 1, the Main PLL is currently locked. If 0, the Main PLL is currently not locked.	RO	0x0
5	sdrplllost	If 1, the SDRAM PLL has lost lock at least once since this bit was cleared. If 0, the SDRAM PLL has not lost lock since this bit was cleared.	RW	0x0
4	perplllost	If 1, the Peripheral PLL has lost lock at least once since this bit was cleared. If 0, the Peripheral PLL has not lost lock since this bit was cleared.	RW	0x0
3	mainplllost	If 1, the Main PLL has lost lock at least once since this bit was cleared. If 0, the Main PLL has not lost lock since this bit was cleared.	RW	0x0
2	sdrpllachieved	If 1, the SDRAM PLL has achieved lock at least once since this bit was cleared. If 0, the SDRAM PLL has not achieved lock since this bit was cleared.	RW	0x0
1	perpllachieved	If 1, the Peripheral PLL has achieved lock at least once since this bit was cleared. If 0, the Peripheral PLL has not achieved lock since this bit was cleared.	RW	0x0
0	mainpllachieved	If 1, the Main PLL has achieved lock at least once since this bit was cleared. If 0, the Main PLL has not achieved lock since this bit was cleared.	RW	0x0

intren

Contain fields that enable the interrupt. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD0400C

Offset: 0xC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										sdrplllost RW 0x0	perplllost RW 0x0	mainplllost RW 0x0	sdrpllachieved RW 0x0	perpllachieved RW 0x0	mainpllachieved RW 0x0

intren Fields

Bit	Name	Description	Access	Reset
5	sdrplllost	When set to 1, the SDRAM PLL lost lock bit is ORed into the Clock Manager interrupt output. When set to 0 the SDRAM PLL lost lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0
4	perplllost	When set to 1, the Peripheral PLL lost lock bit is ORed into the Clock Manager interrupt output. When set to 0 the Peripheral PLL lost lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0
3	mainplllost	When set to 1, the Main PLL lost lock bit is ORed into the Clock Manager interrupt output. When set to 0 the Main PLL lost lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0
2	sdrpllachieved	When set to 1, the SDRAM PLL achieved lock bit is ORed into the Clock Manager interrupt output. When set to 0 the SDRAM PLL achieved lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0
1	perpllachieved	When set to 1, the Peripheral PLL achieved lock bit is ORed into the Clock Manager interrupt output. When set to 0 the Peripheral PLL achieved lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0
0	mainpllachieved	When set to 1, the Main PLL achieved lock bit is ORed into the Clock Manager interrupt output. When set to 0 the Main PLL achieved lock bit is not ORed into the Clock Manager interrupt output.	RW	0x0

dbctrl

Contains fields that control the debug clocks.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ensfmdwr RW 0x1	stayosc1 RW 0x1

dbctrl Fields

Bit	Name	Description	Access	Reset
1	ensfmdwr	When this bit is set the debug clocks will be affected by the assertion of Safe Mode on a warm reset if Stay OSC1 is not set. When this bit is clear the debug clocks will not be affected by the assertion of Safe Mode on a warm reset. If Debug Clocks are in Safe Mode they are taken out of Safe Mode when the Safe Mode bit is cleared independent of this bit.The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x1
0	stayosc1	When this bit is set the debug root clock (Main PLL C2 output) will always be bypassed to the EOSC1_clk independent of any other clock manager settings. When clear the debug source will be a function of register settings in the clock manager. Clocks affected by this bit are dbg_at_clk, dbg_clk, dbg_trace_clk, and dbg_timer_clk. The reset value for this bit is applied on a cold reset. Warm reset has no affect on this bit.	RW	0x1

stat

Provides status of Hardware Managed Clock transition State Machine.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04014

Offset: 0x14

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															busy RO 0x0

stat Fields

Bit	Name	Description	Access	Reset
0	busy	This read only bit indicates that the Hardware Managed clock's state machine is active. If the state machine is active, then the clocks are in transition. Software should poll this bit after changing the source of internal clocks when writing to the BYPASS, CTRL or DBCTRL registers. Immediately following writes to any of these registers, SW should wait until this bit is IDLE before proceeding with any other register writes in the Clock Manager. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit. **Value** — **Description** 0x0 — Clocks stable 0x1 — Clocks in transition	RO	0x0

Main PLL Group Register Descriptions

Contains registers with settings for the Main PLL.

Offset: `0x40`

vco on page 2-32
Contains settings that control the Main PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). The VCO input clock source is always eosc1_clk. Fields are only reset by a cold reset.

misc on page 2-34
Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

mpuclk on page 2-35
Contains settings that control clock mpu_clk generated from the C0 output of the Main PLL. Only reset by a cold reset.

mainclk on page 2-35
Contains settings that control clock main_clk generated from the C1 output of the Main PLL. Only reset by a cold reset.

dbgatclk on page 2-36
Contains settings that control clock dbg_base_clk generated from the C2 output of the Main PLL. Only reset by a cold reset.

mainqspiclk on page 2-36
Contains settings that control clock main_qspi_clk generated from the C3 output of the Main PLL. Only reset by a cold reset.

mainnandsdmmcclk on page 2-37
Contains settings that control clock main_nand_sdmmc_clk generated from the C4 output of the Main PLL. Only reset by a cold reset.

cfgs2fuser0clk on page 2-37
Contains settings that control clock cfg_s2f_user0_clk generated from the C5 output of the Main PLL. Qsys and user documenation refer to cfg_s2f_user0_clk as cfg_h2f_user0_clk. Only reset by a cold reset.

en on page 2-38
Contains fields that control clock enables for clocks derived from the Main PLL. 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

maindiv on page 2-39
Contains fields that control clock dividers for main clocks derived from the Main PLL Fields are only reset by a cold reset.

dbgdiv on page 2-40
Contains fields that control clock dividers for debug clocks derived from the Main PLL Fields are only reset by a cold reset.

tracediv on page 2-41
Contains a field that controls the clock divider for the debug trace clock derived from the Main PLL Only reset by a cold reset.

l4src on page 2-42
Contains fields that select the clock source for L4 MP and SP APB interconnect Fields are only reset by a cold reset.

stat on page 2-43
Contains Output Clock Counter Reset acknowledge status.

vco

Contains settings that control the Main PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). The VCO input clock source is always eosc1_clk. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04040`

Offset: `0x40`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>regextsel
RW 0x1</td><td colspan="6">outreset
RW 0x0</td><td>outresetall
RW 0x0</td><td colspan="2">Reserved</td><td colspan="6">denom
RW 0x1</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">numer
RW 0x1</td><td>pwrdn
RW 0x1</td><td>en
RW 0x0</td><td>bgpwrdn
RW 0x1</td></tr>
</table>

vco Fields

Bit	Name	Description	Access	Reset
31	regextsel	If set to '1', the external regulator is selected for the PLL. If set to '0', the internal regulator is slected. It is strongly recommended to select the external regulator while the PLL is not enabled (in reset), and then disable the external regulater once the PLL becomes enabled. Software should simulateously update the 'Enable' bit and the 'External Regulator Input Select' in the same write access to the VCO register. When the 'Enable' bit is clear, the 'External Regulator Input Select' should be set, and vice versa. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x1
30:25	outreset	Resets the individual PLL output counter. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. LSB 'outreset[0]' corresponds to PLL output clock C0, etc. If set to '1', reset output divider, no clock output from counter. If set to '0', counter is not reset. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x0
24	outresetall	Before releasing Bypass, All Output Counter Reset must be set and cleared by software for correct clock operation. If '1', Reset phase multiplexer and all output counter state. So that after the assertion all the clocks output are start from rising edge align. If '0', phase multiplexer and output counter state not reset and no change to the phase of the clock outputs.	RW	0x0
21:16	denom	Denominator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1
15:3	numer	Numerator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1

Bit	Name	Description	Access	Reset
2	pwrdn	If '1', power down analog circuitry. If '0', analog circuitry not powered down.	RW	0x1
1	en	If '1', VCO is enabled. If '0', VCO is in reset.	RW	0x0
0	bgpwrdn	If '1', powers down bandgap. If '0', bandgap is not power down.	RW	0x1

misc

Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04044

Offset: 0x44

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	saten RW 0x1	fasten RW 0x0	bwadj RW 0x1												bwadjen RW 0x0

misc Fields

Bit	Name	Description	Access	Reset
14	saten	Enables saturation behavior.	RW	0x1
13	fasten	Enables fast locking circuit.	RW	0x0
12:1	bwadj	Provides Loop Bandwidth Adjust value.	RW	0x1
0	bwadjen	If set to 1, the Loop Bandwidth Adjust value comes from the Loop Bandwidth Adjust field. If set to 0, the Loop Bandwidth Adjust value equals the M field divided by 2 value of the VCO Control Register. The M divided by 2 is the upper 12 bits (12:1) of the M field in the VCO register.	RW	0x0

mpuclk

Contains settings that control clock mpu_clk generated from the C0 output of the Main PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04048`

Offset: `0x48`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">cnt
RW 0x0</td></tr>
</table>

mpuclk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO/2 frequency by the value+1 in this field.	`RW`	`0x0`

mainclk

Contains settings that control clock main_clk generated from the C1 output of the Main PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD0404C`

Offset: `0x4C`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="7">Reserved</td><td colspan="9">cnt
RW 0x0</td></tr>
</table>

mainclk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO/4 frequency by the value+1 in this field.	`RW`	`0x0`

dbgatclk

Contains settings that control clock dbg_base_clk generated from the C2 output of the Main PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x0								

dbgatclk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO/4 frequency by the value+1 in this field.	RW	0x0

mainqspiclk

Contains settings that control clock main_qspi_clk generated from the C3 output of the Main PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04054

Offset: 0x54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x3								

mainqspiclk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x3

mainnandsdmmcclk

Contains settings that control clock main_nand_sdmmc_clk generated from the C4 output of the Main PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04058

Offset: 0x58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x3								

mainnandsdmmcclk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x3

cfgs2fuser0clk

Contains settings that control clock cfg_s2f_user0_clk generated from the C5 output of the Main PLL. Qsys and user documenation refer to cfg_s2f_user0_clk as cfg_h2f_user0_clk. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD0405C

Offset: 0x5C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0xF								

cfgs2fuser0clk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0xF

en

Contains fields that control clock enables for clocks derived from the Main PLL. 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04060

Offset: 0x60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						s2fuser0clk RW 0x1	cfgclk RW 0x1	dbgtimerclk RW 0x1	dbgtraceclk RW 0x1	dbgclk RW 0x1	dbgatclk RW 0x1	l4spclk RW 0x1	l4mpclk RW 0x1	l3mpclk RW 0x1	l4mainclk RW 0x1

en Fields

Bit	Name	Description	Access	Reset
9	s2fuser0clk	Enables clock s2f_user0_clk output. Qsys and user documenation refer to s2f_user0_clk as h2f_user0_clk.	RW	0x1
8	cfgclk	Enables clock cfg_clk output	RW	0x1
7	dbgtimerclk	Enables clock dbg_timer_clk output	RW	0x1
6	dbgtraceclk	Enables clock dbg_trace_clk output	RW	0x1
5	dbgclk	Enables clock dbg_clk output	RW	0x1
4	dbgatclk	Enables clock dbg_at_clk output	RW	0x1
3	l4spclk	Enables clock l4_sp_clk output	RW	0x1
2	l4mpclk	Enables clock l4_mp_clk output	RW	0x1
1	l3mpclk	Enables clock l3_mp_clk output	RW	0x1
0	l4mainclk	Enables clock l4_main_clk output	RW	0x1

maindiv

Contains fields that control clock dividers for main clocks derived from the Main PLL Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04064

Offset: 0x64

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="3">l4spclk
RW 0x0</td><td colspan="3">l4mpclk
RW 0x0</td><td colspan="2">l3spclk
RW 0x0</td><td colspan="2">l3mpclk
RW 0x0</td></tr>
</table>

maindiv Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>9:7</td><td>l4spclk</td><td>The l4_sp_clk is divided down from the periph_base_clk by the value specified in this field.<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Divide By 1</td></tr>
<tr><td>0x1</td><td>Divide By 2</td></tr>
<tr><td>0x2</td><td>Divide By 4</td></tr>
<tr><td>0x3</td><td>Divide By 8</td></tr>
<tr><td>0x4</td><td>Divide By 16</td></tr>
<tr><td>0x5</td><td>Reserved</td></tr>
<tr><td>0x6</td><td>Reserved</td></tr>
<tr><td>0x7</td><td>Reserved</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
6:4	`l4mpclk`	The l4_mp_clk is divided down from the periph_base_clk by the value specified in this field. **Value** — **Description** 0x0 — Divide By 1 0x1 — Divide By 2 0x2 — Divide By 4 0x3 — Divide By 8 0x4 — Divide By 16 0x5 — Reserved 0x6 — Reserved 0x7 — Reserved	RW	0x0
3:2	`l3spclk`	The l3_sp_clk is divided down from the l3_mp_clk by the value specified in this field. **Value** — **Description** 0x0 — Divide by 1 0x1 — Divide by 2	RW	0x0
1:0	`l3mpclk`	The l3_mp_clk is divided down from the l3_main_clk by the value specified in this field. **Value** — **Description** 0x0 — Divide by 1 0x1 — Divide by 2	RW	0x0

dbgdiv

Contains fields that control clock dividers for debug clocks derived from the Main PLL Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04068

Offset: `0x68`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												dbgclk RW 0x1		dbgatclk RW 0x0	

dbgdiv Fields

Bit	Name	Description	Access	Reset
3:2	dbgclk	The dbg_clk is divided down from the dbg_at_clk by the value specified in this field. **Value** **Description** 0x1 Divide by 2 0x2 Divide by 4	RW	0x1
1:0	dbgatclk	The dbg_at_clk is divided down from the C2 output of the Main PLL by the value specified in this field. **Value** **Description** 0x0 Divide by 1 0x1 Divide by 2 0x2 Divide by 4	RW	0x0

tracediv

Contains a field that controls the clock divider for the debug trace clock derived from the Main PLL Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD0406C

Offset: 0x6C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													traceclk RW 0x0		

tracediv Fields

Bit	Name	Description	Access	Reset
2:0	`traceclk`	The dbg_trace_clk is divided down from the C2 output of the Main PLL by the value specified in this field. **Value** — **Description** 0x0 — Divide By 1 0x1 — Divide By 2 0x2 — Divide By 4 0x3 — Divide By 8 0x4 — Divide By 16 0x5 — Reserved 0x6 — Reserved 0x7 — Reserved	`RW`	`0x0`

l4src

Contains fields that select the clock source for L4 MP and SP APB interconnect Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04070`

Offset: `0x70`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														l4sp RW 0x0	l4mp RW 0x0

l4src Fields

Bit	Name	Description	Access	Reset
1	`l4sp`	Selects the source for l4_sp_clk **Value** — **Description** 0x0 — main_clk 0x1 — periph_base_clk	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	`l4mp`	Selects the source for l4_mp_clk **Value** **Description** 0x0 main_clk 0x1 periph_base_clk	RW	0x0

stat

Contains Output Clock Counter Reset acknowledge status.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04074`

Offset: `0x74`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`										`outresetack` `RO 0x0`					

stat Fields

Bit	Name	Description	Access	Reset
5:0	`outresetack`	These read only bits per PLL output indicate that the PLL has received the Output Reset Counter request and has gracefully stopped the respective PLL output clock. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit. **Value** **Description** 0x0 Idle 0x1 Output Counter Acknowledge received.	RO	0x0

Peripheral PLL Group Register Descriptions

Contains registers with settings for the Peripheral PLL.

Offset: `0x80`

vco on page 2-45
Contains settings that control the Peripheral PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). Fields are only reset by a cold reset.

misc on page 2-47
Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

emac0clk on page 2-47
Contains settings that control clock emac0_clk generated from the C0 output of the Peripheral PLL. Only reset by a cold reset.

emac1clk on page 2-48
Contains settings that control clock emac1_clk generated from the C1 output of the Peripheral PLL. Only reset by a cold reset.

perqspiclk on page 2-48
Contains settings that control clock periph_qspi_clk generated from the C2 output of the Peripheral PLL. Only reset by a cold reset.

pernandsdmmcclk on page 2-49
Contains settings that control clock periph_nand_sdmmc_clk generated from the C3 output of the Peripheral PLL. Only reset by a cold reset.

perbaseclk on page 2-49
Contains settings that control clock periph_base_clk generated from the C4 output of the Peripheral PLL. Only reset by a cold reset.

s2fuser1clk on page 2-50
Contains settings that control clock s2f_user1_clk generated from the C5 output of the Peripheral PLL. Qsys and user documenation refer to s2f_user1_clk as h2f_user1_clk. Only reset by a cold reset.

en on page 2-50
Contains fields that control clock enables for clocks derived from the Peripheral PLL 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

div on page 2-52
Contains fields that control clock dividers for clocks derived from the Peripheral PLL Fields are only reset by a cold reset.

gpiodiv on page 2-54
Contains a field that controls the clock divider for the GPIO De-bounce clock. Only reset by a cold reset.

src on page 2-54
Contains fields that select the source clocks for the flash controllers. Fields are only reset by a cold reset.

stat on page 2-56
Contains Output Clock Counter Reset acknowledge status.

vco

Contains settings that control the Peripheral PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04080`

Offset: `0x80`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
regextsel RW 0x1	outreset RW 0x0						outresetall RW 0x0	psrc RW 0x0		denom RW 0x1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
numer RW 0x1													pwrdn RW 0x1	en RW 0x0	bgpwrdn RW 0x1

vco Fields

Bit	Name	Description	Access	Reset
31	`regextsel`	If set to '1', the external regulator is selected for the PLL. If set to '0', the internal regulator is slected. It is strongly recommended to select the external regulator while the PLL is not enabled (in reset), and then disable the external regulater once the PLL becomes enabled. Software should simulateously update the 'Enable' bit and the 'External Regulator Input Select' in the same write access to the VCO register. When the 'Enable' bit is clear, the 'External Regulator Input Select' should be set, and vice versa. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x1

Bit	Name	Description	Access	Reset
30:25	outreset	Resets the individual PLL output counter. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. LSB 'outreset[0]' corresponds to PLL output clock C0, etc. If set to '1', reset output divider, no clock output from counter. If set to '0', counter is not reset. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x0
24	outresetall	Before releasing Bypass, All Output Counter Reset must be set and cleared by software for correct clock operation. If '1', Reset phase multiplexer and all output counter state. So that after the assertion all the clocks output are start from rising edge align. If '0', phase multiplexer and output counter state not reset and no change to the phase of the clock outputs.	RW	0x0
23:22	psrc	Controls the VCO input clock source. Qsys and user documenation refer to f2s_periph_ref_clk as f2h_periph_ref_clk. **Value** — **Description** 0x0 — eosc1_clk 0x1 — eosc2_clk 0x2 — f2s_periph_ref_clk	RW	0x0
21:16	denom	Denominator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1
15:3	numer	Numerator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1
2	pwrdn	If '1', power down analog circuitry. If '0', analog circuitry not powered down.	RW	0x1

Bit	Name	Description	Access	Reset
1	en	If '1', VCO is enabled. If '0', VCO is in reset.	RW	0x0
0	bgpwrdn	If '1', powers down bandgap. If '0', bandgap is not power down.	RW	0x1

misc

Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04084

Offset: 0x84

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	saten RW 0x1	fasten RW 0x0	bwadj RW 0x1												bwadjen RW 0x0

misc Fields

Bit	Name	Description	Access	Reset
14	saten	Enables saturation behavior.	RW	0x1
13	fasten	Enables fast locking circuit.	RW	0x0
12:1	bwadj	Provides Loop Bandwidth Adjust value.	RW	0x1
0	bwadjen	If set to 1, the Loop Bandwidth Adjust value comes from the Loop Bandwidth Adjust field. If set to 0, the Loop Bandwidth Adjust value equals the M field divided by 2 value of the VCO Control Register. The M divided by 2 is the upper 12 bits (12:1) of the M field in the VCO register.	RW	0x0

emac0clk

Contains settings that control clock emac0_clk generated from the C0 output of the Peripheral PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04088`

Offset: `0x88`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`							`cnt` `RW 0x1`								

emac0clk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

emac1clk

Contains settings that control clock emac1_clk generated from the C1 output of the Peripheral PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD0408C`

Offset: `0x8C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`							`cnt` `RW 0x1`								

emac1clk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

perqspiclk

Contains settings that control clock periph_qspi_clk generated from the C2 output of the Peripheral PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x1								

perqspiclk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x1

pernandsdmmcclk

Contains settings that control clock periph_nand_sdmmc_clk generated from the C3 output of the Peripheral PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD04094

Offset: 0x94

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x1								

pernandsdmmcclk Fields

Bit	Name	Description	Access	Reset
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x1

perbaseclk

Contains settings that control clock periph_base_clk generated from the C4 output of the Peripheral PLL. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD04098`

Offset: `0x98`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x1								

perbaseclk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

s2fuser1clk

Contains settings that control clock s2f_user1_clk generated from the C5 output of the Peripheral PLL. Qsys and user documenation refer to s2f_user1_clk as h2f_user1_clk. Only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD0409C`

Offset: `0x9C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							cnt RW 0x1								

s2fuser1clk Fields

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

en

Contains fields that control clock enables for clocks derived from the Peripheral PLL 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				qspiclk RW 0x1	nandclk RW 0x1	nandxclk RW 0x1	sdmmcclk RW 0x1	s2fuser1clk RW 0x1	gpioclk RW 0x1	can1clk RW 0x1	can0clk RW 0x1	spimclk RW 0x1	usbclk RW 0x1	emac1clk RW 0x1	emac0clk RW 0x1

en Fields

Bit	Name	Description	Access	Reset
11	qspiclk	Enables clock qspi_clk output	RW	0x1
10	nandclk	Enables clock nand_clk output nand_clk Enable should always be de-asserted before the nand_x_clk Enable, and the nand_x_clk Enable should always be asserted before the nand_clk Enable is asserted. A brief delay is also required between switching the enables (8 * nand_clk period).	RW	0x1
9	nandxclk	Enables clock nand_x_clk output nand_clk Enable should always be de-asserted before the nand_x_clk Enable, and the nand_x_clk Enable should always be asserted before the nand_clk Enable is asserted. A brief delay is also required between switching the enables (8 * nand_clk period).	RW	0x1
8	sdmmcclk	Enables clock sdmmc_clk output	RW	0x1
7	s2fuser1clk	Enables clock s2f_user1_clk output. Qsys and user documenation refer to s2f_user1_clk as h2f_user1_clk.	RW	0x1
6	gpioclk	Enables clock gpio_clk output	RW	0x1
5	can1clk	Enables clock can1_clk output	RW	0x1
4	can0clk	Enables clock can0_clk output	RW	0x1

Bit	Name	Description	Access	Reset
3	spimclk	Enables clock spi_m_clk output	RW	0x1
2	usbclk	Enables clock usb_mp_clk output	RW	0x1
1	emac1clk	Enables clock emac1_clk output	RW	0x1
0	emac0clk	Enables clock emac0_clk output	RW	0x1

div

Contains fields that control clock dividers for clocks derived from the Peripheral PLL Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040A4

Offset: 0xA4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				can1clk RW 0x0			can0clk RW 0x0			spimclk RW 0x0			usbclk RW 0x0		

div Fields

Bit	Name	Description	Access	Reset
11:9	can1clk	The can1_clk is divided down from the periph_base_ clk by the value specified in this field. **Value** **Description** 0x0 Divide By 1 0x1 Divide By 2 0x2 Divide By 4 0x3 Divide By 8 0x4 Divide By 16 0x5 Reserved 0x6 Reserved 0x7 Reserved	RW	0x0

Bit	Name	Description	Access	Reset
8:6	can0clk	The can0_clk is divided down from the periph_base_clk by the value specified in this field. **Value** **Description** 0x0 Divide By 1 0x1 Divide By 2 0x2 Divide By 4 0x3 Divide By 8 0x4 Divide By 16 0x5 Reserved 0x6 Reserved 0x7 Reserved	RW	0x0
5:3	spimclk	The spi_m_clk is divided down from the periph_base_clk by the value specified in this field. **Value** **Description** 0x0 Divide By 1 0x1 Divide By 2 0x2 Divide By 4 0x3 Divide By 8 0x4 Divide By 16 0x5 Reserved 0x6 Reserved 0x7 Reserved	RW	0x0

Bit	Name	Description	Access	Reset
2:0	usbclk	The usb_mp_clk is divided down from the periph_base_clk by the value specified in this field. **Value** **Description** 0x0 Divide By 1 0x1 Divide By 2 0x2 Divide By 4 0x3 Divide By 8 0x4 Divide By 16 0x5 Reserved 0x6 Reserved 0x7 Reserved	RW	0x0

gpiodiv

Contains a field that controls the clock divider for the GPIO De-bounce clock. Only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040A8

Offset: 0xA8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								gpiodbclk RW 0x1							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpiodbclk RW 0x1															

gpiodiv Fields

Bit	Name	Description	Access	Reset
23:0	gpiodbclk	The gpio_db_clk is divided down from the periph_base_clk by the value plus one specified in this field. The value 0 (divide by 1) is illegal. A value of 1 indicates divide by 2, 2 divide by 3, etc.	RW	0x1

src

Contains fields that select the source clocks for the flash controllers. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040AC

Offset: 0xAC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										qspi RW 0x1		nand RW 0x1		sdmmc RW 0x1	

src Fields

Bit	Name	Description	Access	Reset
5:4	qspi	Selects the source clock for the QSPI. Qsys and user documenation refer to f2s_periph_ref_clk as f2h_periph_ref_clk. **Value** **Description** 0x0 f2s_periph_ref_clk 0x1 main_qspi_clk 0x2 periph_qspi_clk	RW	0x1
3:2	nand	Selects the source clock for the NAND. Qsys and user documenation refer to f2s_periph_ref_clk as f2h_periph_ref_clk. **Value** **Description** 0x0 f2s_periph_ref_clk 0x1 main_nand_sdmmc_clk 0x2 periph_nand_sdmmc_clk	RW	0x1
1:0	sdmmc	Selects the source clock for the SDMMC. Qsys and user documenation refer to f2s_periph_ref_clk as f2h_periph_ref_clk. **Value** **Description** 0x0 f2s_periph_ref_clk 0x1 main_nand_sdmmc_clk 0x2 periph_nand_sdmmc_clk	RW	0x1

stat

Contains Output Clock Counter Reset acknowledge status.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040B0

Offset: 0xB0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										outresetack RO 0x0					

stat Fields

Bit	Name	Description	Access	Reset
5:0	outresetack	These read only bits per PLL output indicate that the PLL has received the Output Reset Counter request and has gracefully stopped the respective PLL output clock. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit. **Value** **Description** 0x0 Idle 0x1 Output Counter Acknowledge received.	RO	0x0

SDRAM PLL Group Register Descriptions

Contains registers with settings for the SDRAM PLL.

Offset: 0xc0

vco on page 2-57
Contains settings that control the SDRAM PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). Fields are only reset by a cold reset.

ctrl on page 2-59
Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

ddrdqsclk on page 2-60
Contains settings that control clock ddr_dqs_clk generated from the C0 output of the SDRAM PLL. Fields are only reset by a cold reset.

ddr2xdqsclk on page 2-61
Contains settings that control clock ddr_2x_dqs_clk generated from the C1 output of the SDRAM PLL. Fields are only reset by a cold reset.

ddrdqclk on page 2-61
Contains settings that control clock ddr_dq_clk generated from the C2 output of the SDRAM PLL. Fields are only reset by a cold reset.

s2fuser2clk on page 2-62
Contains settings that control clock s2f_user2_clk generated from the C5 output of the SDRAM PLL. Qsys and user documenation refer to s2f_user2_clk as h2f_user2_clk Fields are only reset by a cold reset.

en on page 2-63
Contains fields that control the SDRAM Clock Group enables generated from the SDRAM PLL clock outputs. 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

stat on page 2-64
Contains Output Clock Counter Reset acknowledge status.

vco

Contains settings that control the SDRAM PLL VCO. The VCO output frequency is the input frequency multiplied by the numerator (M+1) and divided by the denominator (N+1). Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD040C0`

Offset: `0xC0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
regextsel RW 0x1	outreset RW 0x0						outresetall RW 0x0	ssrc RW 0x0		denom RW 0x1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
numer RW 0x1													pwrdn RW 0x1	en RW 0x0	bgpwrdn RW 0x1

vco Fields

Bit	Name	Description	Access	Reset
31	`regextsel`	If set to '1', the external regulator is selected for the PLL. If set to '0', the internal regulator is slected. It is strongly recommended to select the external regulator while the PLL is not enabled (in reset), and then disable the external regulater once the PLL becomes enabled. Software should simulateously update the 'Enable' bit and the 'External Regulator Input Select' in the same write access to the VCO register. When the 'Enable' bit is clear, the 'External Regulator Input Select' should be set, and vice versa. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x1
30:25	`outreset`	Resets the individual PLL output counter. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. LSB 'outreset[0]' corresponds to PLL output clock C0, etc. If set to '1', reset output divider, no clock output from counter. If set to '0', counter is not reset. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit.	RW	0x0
24	`outresetall`	Before releasing Bypass, All Output Counter Reset must be set and cleared by software for correct clock operation. If '1', Reset phase multiplexer and output counter state. So that after the assertion all the clocks output are start from rising edge align. If '0', phase multiplexer and output counter state not reset and no change to the phase of the clock outputs.	RW	0x0
23:22	`ssrc`	Controls the VCO input clock source. The PLL must by bypassed to eosc1_clk before changing this field. Qsys and user documenation refer to f2s_sdram_ref_clk as f2h_sdram_ref_clk. **Value** **Description** 0x0 eosc1_clk 0x1 eosc2_clk 0x2 f2s_sdram_ref_clk	RW	0x0

Bit	Name	Description	Access	Reset
21:16	denom	Denominator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1
15:3	numer	Numerator in VCO output frequency equation. For incremental frequency change, if the new value lead to less than 20% of the frequency change, this value can be changed without resetting the PLL. The Numerator and Denominator can not be changed at the same time for incremental frequency changed.	RW	0x1
2	pwrdn	If '1', power down analog circuitry. If '0', analog circuitry not powered down.	RW	0x1
1	en	If '1', VCO is enabled. If '0', VCO is in reset.	RW	0x0
0	bgpwrdn	If '1', powers down bandgap. If '0', bandgap is not power down.	RW	0x1

ctrl

Contains VCO control signals and other PLL control signals need to be controllable through register. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040C4

Offset: 0xC4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	saten RW 0x1	fasten RW 0x0	bwadj RW 0x1												bwadjen RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
14	saten	Enables saturation behavior.	RW	0x1

Bit	Name	Description	Access	Reset
13	`fasten`	Enables fast locking circuit.	RW	0x0
12:1	`bwadj`	Provides Loop Bandwidth Adjust value.	RW	0x1
0	`bwadjen`	If set to 1, the Loop Bandwidth Adjust value comes from the Loop Bandwidth Adjust field. If set to 0, the Loop Bandwidth Adjust value equals the M field divided by 2 value of the VCO Control Register. The M divided by 2 is the upper 12 bits (12:1) of the M field in the VCO register.	RW	0x0

ddrdqsclk

Contains settings that control clock ddr_dqs_clk generated from the C0 output of the SDRAM PLL. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD040C8`

Offset: `0xC8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											phase RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
phase RW 0x0							cnt RW 0x1								

ddrdqsclk Fields

Bit	Name	Description	Access	Reset
20:9	`phase`	Increment the phase of the VCO output by the value in this field multiplied by 45 degrees. The accumulated phase shift is the total shifted amount since the last assertion of the 'SDRAM All Output Divider Reset' bit in the SDRAM vco control register. In order to guarantee the phase shift to a known value, 'SDRAM clocks output phase align' bit should be asserted before programming this field. This field is only writeable by SW when it is zero. HW updates this field in real time as the phase adjustment is being made. SW may poll this field waiting for zero indicating the phase adjustment has completed by HW.	RW	0x0

Bit	Name	Description	Access	Reset
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

ddr2xdqsclk

Contains settings that control clock ddr_2x_dqs_clk generated from the C1 output of the SDRAM PLL. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
`clkmgr`	`0xFFD04000`	`0xFFD040CC`

Offset: `0xCC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											phase RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
phase RW 0x0							cnt RW 0x1								

ddr2xdqsclk Fields

Bit	Name	Description	Access	Reset
20:9	`phase`	Increment the phase of the VCO output by the value in this field multiplied by 45 degrees. The accumulated phase shift is the total shifted amount since the last assertion of the 'SDRAM All Output Divider Reset' bit in the SDRAM vco control register. In order to guarantee the phase shift to a known value, 'SDRAM clocks output phase align' bit should be asserted before programming this field. This field is only writeable by SW when it is zero. HW updates this field in real time as the phase adjustment is being made. SW may poll this field waiting for zero indicating the phase adjustment has completed by HW.	`RW`	`0x0`
8:0	`cnt`	Divides the VCO frequency by the value+1 in this field.	`RW`	`0x1`

ddrdqclk

Contains settings that control clock ddr_dq_clk generated from the C2 output of the SDRAM PLL. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040D0

Offset: 0xD0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											phase RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
phase RW 0x0							cnt RW 0x1								

ddrdqclk Fields

Bit	Name	Description	Access	Reset
20:9	phase	Increment the phase of the VCO output by the value in this field multiplied by 45 degrees. The accumulated phase shift is the total shifted amount since the last assertion of the 'SDRAM All Output Divider Reset' bit in the SDRAM vco control register. In order to guarantee the phase shift to a known value, 'SDRAM clocks output phase align' bit should be asserted before programming this field. This field is only writeable by SW when it is zero. HW updates this field in real time as the phase adjustment is being made. SW may poll this field waiting for zero indicating the phase adjustment has completed by HW.	RW	0x0
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x1

s2fuser2clk

Contains settings that control clock s2f_user2_clk generated from the C5 output of the SDRAM PLL. Qsys and user documenation refer to s2f_user2_clk as h2f_user2_clk Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040D4

Offset: 0xD4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											phase RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
phase RW 0x0							cnt RW 0x1								

s2fuser2clk Fields

Bit	Name	Description	Access	Reset
20:9	phase	Increment the phase of the VCO output by the value in this field multiplied by 45 degrees. The accumulated phase shift is the total shifted amount since the last assertion of the 'SDRAM All Output Divider Reset' bit in the SDRAM vco control register. In order to guarantee the phase shift to a known value, 'SDRAM clocks output phase align' bit should be asserted before programming this field. This field is only writeable by SW when it is zero. HW updates this field in real time as the phase adjustment is being made. SW may poll this field waiting for zero indicating the phase adjustment has completed by HW.	RW	0x0
8:0	cnt	Divides the VCO frequency by the value+1 in this field.	RW	0x1

en

Contains fields that control the SDRAM Clock Group enables generated from the SDRAM PLL clock outputs. 1: The clock is enabled. 0: The clock is disabled. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040D8

Offset: 0xD8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												s2fuser2clk RW 0x1	ddrdqclk RW 0x1	ddr2xdqsclk RW 0x1	ddrdqsclk RW 0x1

en Fields

Bit	Name	Description	Access	Reset
3	s2fuser2clk	Enables clock s2f_user2_clk output. Qsys and user documenation refer to s2f_user2_clk as h2f_user2_clk.	RW	0x1
2	ddrdqclk	Enables clock ddr_dq_clk output	RW	0x1
1	ddr2xdqsclk	Enables clock ddr_2x_dqs_clk output	RW	0x1
0	ddrdqsclk	Enables clock ddr_dqs_clk output	RW	0x1

stat

Contains Output Clock Counter Reset acknowledge status.

Module Instance	Base Address	Register Address
clkmgr	0xFFD04000	0xFFD040DC

Offset: 0xDC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										outresetack RO 0x0					

stat Fields

Bit	Name	Description	Access	Reset
5:0	`outresetack`	These read only bits per PLL output indicate that the PLL has received the Output Reset Counter request and has gracefully stopped the respective PLL output clock. For software to change the PLL output counter without producing glitches on the respective clock, SW must set the VCO register respective Output Counter Reset bit. Software then polls the respective Output Counter Reset Acknowledge bit in the Output Counter Reset Ack Status Register. Software then writes the appropriate counter register, and then clears the respective VCO register Output Counter Reset bit. The reset value of this bit is applied on a cold reset; warm reset has no affect on this bit. **Value** **Description** 0x0 Idle 0x1 Output Counter Acknowledge received.	RO	0x0

Document Revision History

Table 2-10: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	E0SC1 changed to HPS_CLK1 E0SC2 changed to HPS_CLK2 Added Address Map and Register Descriptions
February 2014	2014.02.28	Updated content in the "Peripheral Clock Group" section
December 2013	2013.12.30	Minor formatting updates.
November 2012	1.2	Minor updates.
May 2012	1.1	• Reorganized and expanded functional description section. • Added address map and register definitions section.
January 2012	1.0	Initial release.

Reset Manager 3

2014.06.30

cv_54003 Subscribe Send Feedback

The reset manager generates module reset signals based on reset requests from the various sources in the HPS and FPGA fabric, and software writing to the module-reset control registers. The reset manager ensures that a reset request from the FPGA fabric can occur only after the FPGA portion of the system-on-a-chip (SoC) device is configured.

The HPS contains three reset domains. Each reset domain can be reset independently. Each register in the HPS that can be reset belongs to one particular reset domain.

Table 3-1: HPS Reset Domains

Domain Name	Domain Logic
TAP	JTAG test access port (TAP) controller, which is used by the debug access port (DAP).
Debug	All debug logic including most of the DAP, CoreSight™ components connected to the debug peripheral bus, trace, the microprocessor unit (MPU) subsystem, and the FPGA fabric.
System	All HPS logic except what is in the TAP and debug reset domains. Includes nondebug logic in the FPGA fabric connected to the HPS reset signals.

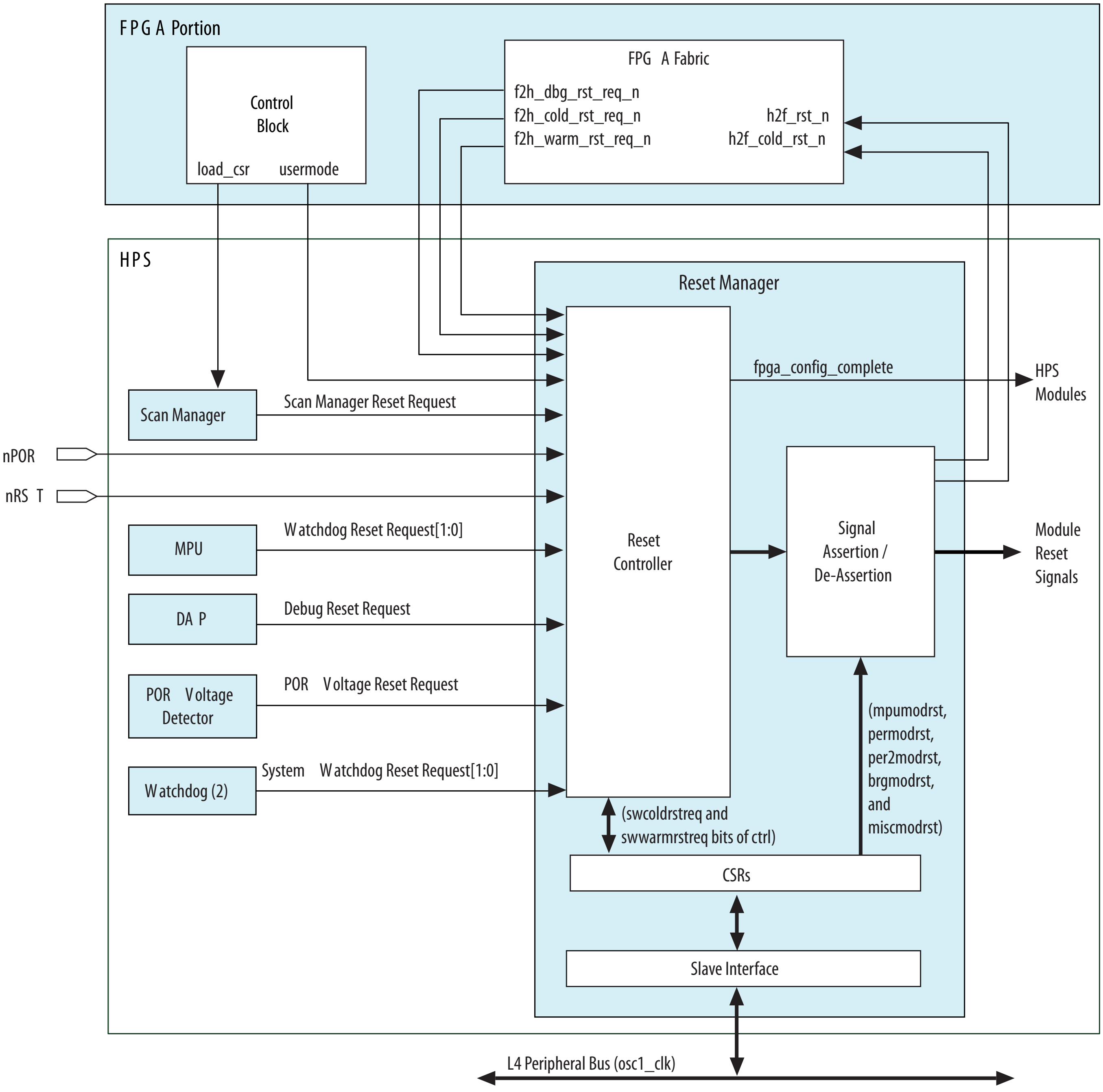
The HPS supports the following reset types:

- Cold reset (power-on reset)
  - Used to ensure the HPS is placed in a default state sufficient for software to boot
  - Triggered by a power-on reset and other sources
  - Resets all HPS logic that can be reset
  - Affects all reset domains
- Warm reset
  - Occurs after HPS has already completed a cold reset
  - Used to recover system from a non-responsive condition
  - Resets a subset of the HPS state reset by a cold reset
  - Only affects the system reset domain, which allows debugging (including trace) to operate through the warm reset
- Debug reset
  - Occurs after HPS has already been through a cold reset
  - Used to recover debug logic from a non-responsive condition
  - Only affects the debug reset domain

Reset Manager Block Diagram and System Integration

Figure 3-1: Reset Manager Block Diagram

The following figure shows a block diagram of the reset manager in the SoC device. For clarity, reset-related handshaking signals to other HPS modules and to the clock manager module are omitted.



HPS External Reset Sources

The following table lists the reset sources external to the HPS. All signals are synchronous to the osc1_clk clock. The reset signals from the HPS to the FPGA fabric must be synchronized to your user logic clock domain.

Table 3-2: HPS External Reset Sources

Source	Description
f2h_cold_rst_req_n	Cold reset request from FPGA fabric (active low)
f2h_warm_rst_req_n	Warm reset request from FPGA fabric (active low)
f2h_dbg_rst_req_n	Debug reset request from FPGA fabric (active low)
h2f_cold_rst_n	Cold-only reset to FPGA fabric (active low)
h2f_rst_n	Cold or warm reset to FPGA fabric (active low)
h2f_dbg_rst_n	Debug reset (dbg_rst_n) to FPGA fabric (active low)
load_csr	Cold-only reset from FPGA control block (CB) and scan manager
nPOR	Power-on reset pin (active low)
nRST	Warm reset pin (active low)

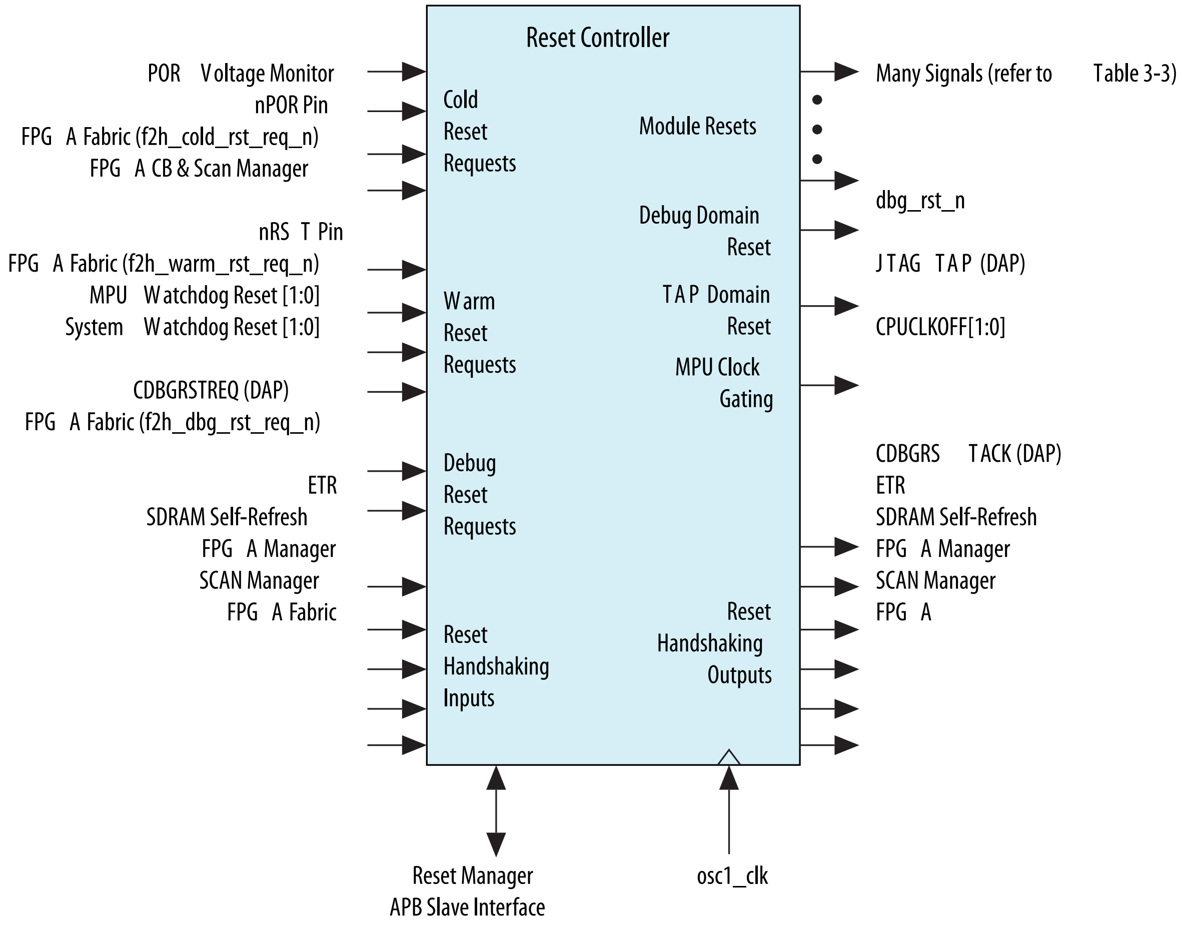
Reset Controller

The reset controller performs the following functions:

- Accepts reset requests from the FPGA CB, FPGA fabric, modules in the HPS, and reset pins
- Generates an individual reset signal for each module instance for all modules in the HPS
- Provides reset handshaking signals to support system reset behavior

The reset controller generates module reset signals from external reset requests and internal reset requests. External reset requests originate from sources external to the reset manager. Internal reset requests originate from control registers in the reset manager.

Figure 3-2: Reset Controller Signals



The reset controller supports the following cold reset requests:

- Power-on reset (POR) voltage monitor
- Cold reset request pin (nPOR)
- FPGA fabric
- FPGA CB and scan manager
- Software cold reset request bit (swcoldrstreq) of the control register (ctrl)

The reset controller supports the following warm reset requests:

- Warm reset request pin (nRST)
- FPGA fabric
- Software warm reset request bit (swwarmrstreq) of the ctrl register
- MPU watchdog reset requests for CPU0 and CPU1
- System watchdog timer 0 and 1 reset requests

The reset controller supports the following debug reset requests:

- CDBGRSTREQ from DAP
- FPGA fabric

Module Reset Signals

The following tables list the module reset signals. The module reset signals are organized in groups for the MPU, peripherals, bridges, the level 3 (L3) interconnect, and miscellaneous.

In the following tables, columns marked for Cold Reset, Warm Reset, and Debug Reset denote reset signals asserted by each type of reset. For example, writing a 1 to the swwarmrstreq bit in the ctrl register resets all the modules that have a checkmark in the Warm Reset column.

The column marked for Software Deassert denotes reset signals that are left asserted by the reset manager. To activate the related modules, software can deassert these reset signals as needed by writing to the following reset manager registers:

- MPU module reset register (mpumodrst)
- Peripheral module reset register (permodrst)
- Peripheral 2 module reset register (per2modrst)
- Bridge module reset register (brgmodrst)

Table 3-3: MPU Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
mpu_cpu_rst_n[0]	Resets each processor in the MPU	System	X	X		
mpu_cpu_rst_n[1]	Resets each processor in the MPU	System	X	X		X
mpu_wd_rst_n	Resets both per-processor watchdogs in the MPU	System	X	X		
mpu_scu_periph_rst_n	Resets Snoop Control Unit (SCU) and peripherals	System	X	X		
mpu_l2_rst_n	Level 2 (L2) cache reset	System	X	X		

Table 3-4: PER Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
emac_rst_n[1:0]	Resets each EMAC	System	X	X		X
usb_rst_n[1:0]	Resets each USB	System	X	X		X
nand_flash_rst_n	Resets NAND flash controller	System	X	X		X

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`qspi_flash_rst_n`	Resets quad SPI flash controller	System	X	X		X
`watchdog_rst_n[1:0]`	Resets each system watchdog timer	System	X	X		X
`osc1_timer_rst_n[1:0]`	Resets each OSC1 timer	System	X	X		X
`sp_timer_rst_n[1:0]`	Resets each SP timer	System	X	X		X
`i2c_rst_n[3:0]`	Resets each I^{2}C controller	System	X	X		X
`uart_rst_n[1:0]`	Resets each UART	System	X	X		X
`spim_rst_n[1:0]`	Resets SPI master controller	System	X	X		X
`spis_rst_n[1:0]`	Resets SPI slave controller	System	X	X		X
`sdmmc_rst_n`	Resets SD/MMC controller	System	X	X		X
`can_rst_n[1:0]`	Resets each CAN controller	System	X	X		X
`gpio_rst_n[2:0]`	Resets each GPIO interface	System	X	X		X
`dma_rst_n`	Resets DMA controller	System	X	X		X
`sdram_rst_n`	Resets SDRAM subsystem (resets logic associated with cold or warm reset)	System	X	X		X

Table 3-5: PER2 Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`dma_periph_if_rst_n[7:0]`	DMA controller request interface from FPGA fabric to DMA controller	System	X	X		X

Table 3-6: Bridge Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`hps2fpga_bridge_rst_n`	Resets HPS-to-FPGA AMBA® Advanced eXtensible Interface (AXI™) bridge	System	X	X		X
`fpga2hps_bridge_rst_n`	Resets FPGA-to-HPS AXI bridge	System	X	X		X
`lwhps2fpga_bridge_rst_n`	Resets lightweight HPS-to-FPGA AXI bridge	System	X	X		X

Table 3-7: MISC Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`boot_rom_rst_n`	Resets boot ROM	System	X	X		
`onchip_ram_rst_n`	Resets on-chip RAM	System	X	X		
`sys_manager_rst_n`	Resets system manager (resets logic associated with cold or warm reset)	System	X	X		
`sys_manager_cold_rst_n`	Resets system manager (resets logic associated with cold reset only)	System	X			
`fpga_manager_rst_n`	Resets FPGA manager	System	X	X		
`acp_id_mapper_rst_n`	Resets ACP ID mapper	System	X	X		
`h2f_rst_n`	Resets user logic in FPGA fabric (resets logic associated with cold or warm reset)	System	X	X		
`h2f_cold_rst_n`	Resets user logic in FPGA fabric (resets logic associated with cold reset only)	System	X			
`rst_pin_rst_n`	Pulls nRST pin low	System	X	X		
`timestamp_cold_rst_n`	Resets debug timestamp to 0x0	System	X			

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`clk_manager_cold_rst_n`	Resets clock manager (resets logic associated with cold reset only)	System	X			
`scan_manager_rst_n`	Resets scan manager	System	X	X		
`frz_ctrl_cold_rst_n`	Resets freeze controller (resets logic associated with cold reset only)	System	X			
`sys_dbg_rst_n`	Resets debug masters and slaves connected to L3 interconnect and level 4 (L4) buses	System	X	X		
`dbg_rst_n`	Resets debug components including DAP, trace, MPU debug logic, and any user debug logic in the FPGA fabric	Debug	X		X	
`tap_cold_rst_n`	Resets portion of TAP controller in the DAP that must be reset on a cold reset	TAP	X			
`sdram_cold_rst_n`	Resets SDRAM subsystem (resets logic associated with cold reset only)	System	X	X		

Table 3-8: L3 Group, Generated Module Resets

Module Reset Signal	Description	Reset Domain	Cold Reset	Warm Reset	Debug Reset	Software Deassert
`l3_rst_n`	Resets L3 interconnect and L4 buses	System	X	X		

Slave Interface and Status Register

The reset manager slave interface is used to control and monitor the reset states.

The status register (`stat`) in the reset manager contains the status of the reset requester. The register contains a bit for each reset request. The `stat` register captures all reset requests that have occurred. Software is responsible for clearing the bits.

Functional Description of the Reset Manager

The reset manager generates reset signals to modules in the HPS and to the FPGA fabric. The following actions generate reset signals:

- Software writing a 1 to the `swcoldrstreq` or `swwarmrstreq` bits in the `ctrl` register. Writing either bit causes the reset controller to perform a reset sequence.
- Software writing to the `mpumodrst`, `permodrst`, `per2modrst`, `brgmodrst`, or `miscmodrst` module reset control registers.
- Asserting reset request signals triggers the reset controller. All external reset requests cause the reset controller to perform a reset sequence.

Multiple reset requests can be driven to the reset manager at the same time. Cold reset requests take priority over warm and debug reset requests. Higher priority reset requests preempt lower priority reset requests. There is no priority difference among reset requests within the same domain.

If a cold reset request is issued while another cold reset is already underway, the reset manager extends the reset period for all the module reset outputs until all cold reset requests are removed. If a cold reset request is issued while the reset manager is removing other modules out of the reset state, the reset manager returns those modules back to the reset state.

If a warm reset request is issued while another warm reset is already underway, the first warm reset completes before the second warm reset begins. If the second warm reset request is removed before the first warm reset completes, the warm first reset is extended to meet the timing requirements of the second warm reset request.

The `nPOR` pin can be used to extend the cold reset beyond what the POR voltage monitor automatically provides. The use of the `nPOR` pin is optional and can be tied high when it is not required.

For information regarding HPS power supply operating conditions, refer to the device datasheet.

Related Information

Cyclone V Device Datasheet
For information about the required duration of reset request signal assertion, refer to the Cyclone V Device Datasheet.

Reset Sequencing

The reset controller sequences resets without software assistance. Module reset signals are asserted asynchronously at the same time. The reset manager deasserts the module reset signals synchronous to the `osc1_clk` clock. Module reset signals are deasserted in groups in a fixed sequence. All module reset signals in a group are deasserted at the same time.

The reset manager sends a safe mode request to the clock manager to put the clock manager in safe mode, which creates a fixed and known relationship between the `osc1_clk` clock and all other clocks generated by the clock manager.

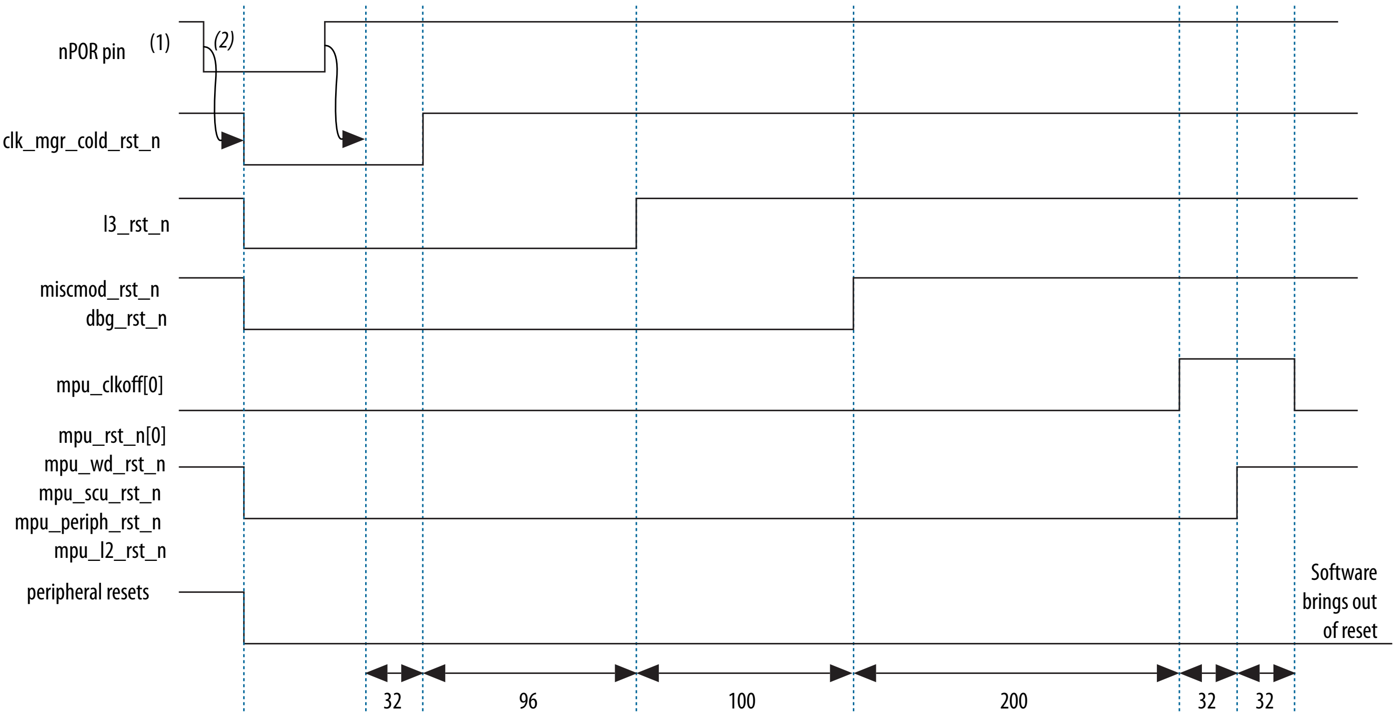
After the reset manager releases the MPU subsystem from reset, CPU1 is left in reset and CPU0 begins executing code from the reset vector address. Software is responsible for deasserting CPU1 and other resets, as shown in MPU Group, Generated Module Resets Table. Software deasserts resets by writing the `mpumodrst`, `permodrst`, `per2modrst`, `brgmodrst`, and `miscmodrst` module-reset control registers.

Software can also bypass the reset controller and generate reset signals directly through the module-reset control registers. In this case, software is responsible for asserting module reset signals, driving them for the appropriate duration, and deasserting them in the correct order. The clock manager is not typically in safe mode during this time, so software is responsible for knowing the relationship between the clocks generated by the clock manager. Software must not assert a module reset signal that would prevent software from deasserting the module reset signal. For example, software should not assert the module reset to the processor executing the software.

Table 3-9: Minimum Pulse Width

Reset Type	Value
Warm Reset	6 `osc1_clk` cycles
Cold Reset	6 `osc1_clk` cycles

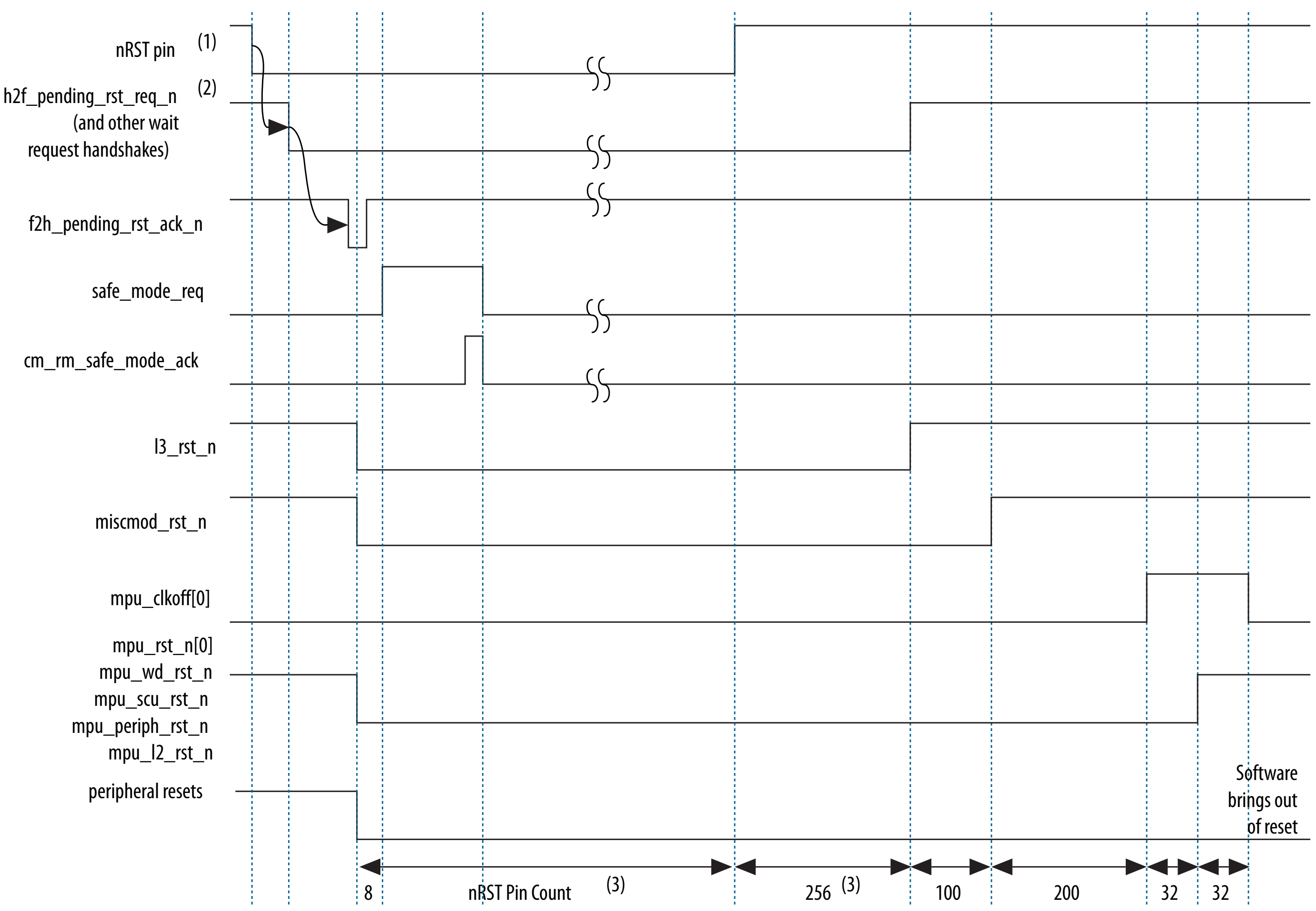
Figure 3-3: Cold Reset Timing Diagram



(1) Cold reset can be initiated from several other sources: FPGA CB, FPGA fabric, modules in the HPS, and reset pins.

(2) This dependency applies to all the reset signals.

Figure 3-4: Warm Reset Timing Diagram



(1) Cold reset can be initiated from several other sources: FPGA CB, FPGA fabric, modules in the HPS, and reset pins.

(2) When the nRSTpin count is zero, the 256 cycle stretch count is skipped and the start of the deassertion sequence is determined by the safe mode acknowledge signal or the userreleasing the warm reset button, whichever occurs later.

The cold and warm reset sequences consist of different reset assertion sequences and the same deassertion sequence. The following sections describe the sequences.

Note: Cold and Warm reset affect only the cpu0 and by default cpu1 is held in reset until the software running in the cpu0 releases it.

Related Information

Clock Manager on page 2-1

For more information about safe mode, refer to the *Clock Manager* chapter.

Cold Reset Assertion Sequence

The following list describes the assertion steps for cold reset shown in the cold Reset Timing Diagram:

1. Assert module resets.
2. Wait for 32 cycles. Deassert clock manager cold reset.
3. Wait for 96 cycles (so clocks can stabilize).
4. Proceed to the "Cold and Warm Reset Deassertion Sequence" section using the following link.

Related Information

Cold and Warm Reset Deassertion Sequence on page 3-13

Warm Reset Assertion Sequence

The following list describes the assertion steps for warm reset shown in the Warm Reset Timing Diagram:

1. Optionally, handshake with the embedded trace router (ETR) and wait for acknowledge.
2. Optionally, handshake with the FPGA fabric and wait for acknowledge.
3. Optionally, handshake with the SDRAM controller, scan manager, and FPGA manager, and wait for acknowledges.
4. Assert module resets (except the MPU watchdog timer resets when the MPU watchdog timers are the only request sources).
5. Wait for 8 cycles and send a safe mode request to the clock manager.
6. Wait for the greater of the `nRST` pin count + 256 stretch count, or the warm reset counter, or the clock manager safe mode acknowledge, then deassert all handshakes except warm reset ETR handshake (which is deasserted by software).
7. Proceed to the "Cold and Warm Reset Deassertion Sequence" section using the following link.

Note: The HPS_nRST is a bidirectional signal that is driven out when a warm reset is generated in the chip.

Related Information

Cold and Warm Reset Deassertion Sequence on page 3-13

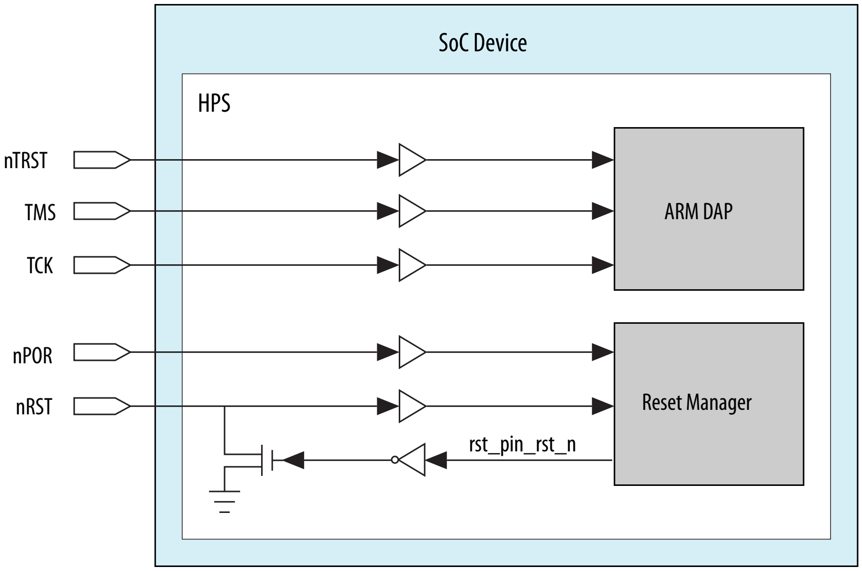
Cold and Warm Reset Deassertion Sequence

The following list describes the deassertion steps for both cold and warm reset shown in the Cold Reset Timing Diagram and Warm Reset Timing Diagram:

1. Deassert L3 reset.
2. Wait for 100 cycles. Deassert resets for miscellaneous-type and debug (cold only) modules.
3. Wait for 200 cycles. Assert `mpu_clkoff` for CPU0 and CPU1.
4. Wait for 32 cycles. Deassert resets for MPU modules.
5. Wait for 32 cycles. Deassert `mpu_clkoff` for CPU0 and CPU1.
6. Peripherals remain held in reset until software brings them out of reset.

Reset Pins

Figure 3-5: Reset Pins



The test reset (nTRST), test mode select (TMS), and test clock (TCK) pins are associated with the TAP reset domain and are used to reset the TAP controller in the DAP. These pins are not connected to the reset manager.

The nPOR and nRST pins are used to request cold and warm resets respectively. The nRST pin is an open drain output as well. Any warm reset request causes the reset manager to drive the rst_pin_rst_n signal output low, which drives the nRST pin low. The amount of time the reset manager pulls nRST low is controlled by the nRST pin count field (nrstcnt) of the reset cycles count register (counts). This technique can be used to reset external devices (such as external memories) connected to the HPS.

Reset Effects

The following list describes how reset affects HPS logic:

- The TAP reset domain ignores warm reset.
- The debug reset domain ignores warm reset.
- System reset domain cold resets ignore warm reset.
- Each module defines reset behavior individually.

Related Information

Cyclone V Device Handbook Volume 3: Hard Processor System Technical Reference Manual

For more information, refer to the individual chapters in the *Cyclone V Device Handbook, Volume 3*.

Altering Warm Reset System Response

Registers in the clock manager, system manager, and reset manager control how warm reset affects the HPS. You can control the impact of a warm reset on the clocks and I/O elements.

Altera strongly recommends using Altera-provided libraries to configure and control this functionality.

The default warm reset behavior takes all clocks and I/O elements through a cold reset response. As your software becomes more stable or for debug purposes, you can alter the system response to a warm reset. The following suggestions provide ways to alter the system response to a warm reset. None of the register bits that control these items are affected by warm reset.

- Boot from on-chip RAM—enables warm boot from on-chip RAM instead of the boot ROM. When enabled, the boot ROM code validates the RAM code and jumps to it, making no changes to clocks or any other system settings prior to executing user code from on-chip RAM.
- Disable safe mode on warm reset—allows software to transition through a warm reset without affecting the clocks. Because the boot ROM code indirectly configures the clock settings after warm reset, Altera recommends to only disable safe mode when the HPS is not booting from a flash device.
- Disable safe mode on warm reset for the debug clocks—keeps the debug clocks from being affected by the assertion of safe mode request on a warm reset. This technique allows fast debug clocks, such as trace, to continue running through a warm reset. When enabled, the clock manager puts the debug clocks to their safe frequencies to respond to a safe mode request from the reset manager on a warm reset. Disable safe mode on warm reset for the debug clocks only when you are running the debug clocks off the main PLL VCO and you are certain the main PLL cannot be impacted by the event which caused the warm reset.
- Use the `osc1_clk` clock for debug control—keeps the debug base clock (main PLL C2 output) always bypassed to the `osc1_clk` external clock, independent of other clock manager settings. When implemented, disabling safe mode on warm reset for the debug clocks has no effect.

Related Information

Clock Manager on page 2-1
For more information about safe mode, refer to the *Clock Manager* chapter.

Reset Handshaking

The reset manager participates in several reset handshaking protocols to ensure other modules are safely reset.

Before issuing a warm reset, the reset manager performs a handshake with several modules to allow them to prepare for a warm reset. The handshake logic ensures the following conditions:

- ETR master has no pending master transactions to the L3 interconnect
- Optionally preserve SDRAM contents during warm reset by issuing self-refresh mode request
- FPGA manager stops generating configuration clock
- Scan manager stops generating JTAG and I/O configuration clocks
- Warns the FPGA fabric of the forthcoming warm reset

Similarly, the handshake logic associated with ETR also occurs during the debug reset to ensure that the ETR master has no pending master transactions to the L3 interconnect before the debug reset is issued. This action ensures that when ETR undergoes a debug reset, the reset has no adverse effects on the system domain portion of the ETR.

Reset Manager Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridge consist of the following regions:

- Reset Manager Module

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

Reset Manager Module Address Map

Registers in the Reset Manager module

Base Address: 0xFFD05000

Reset Manager Module

Register	Offset	Width	Access	Reset Value	Description
stat on page 3-16	0x0	32	RW	0x0	Status Register
ctrl on page 3-18	0x4	32	RW	0x100000	Control Register
counts on page 3-22	0x8	32	RW	0x80080	Reset Cycles Count Register
mpumodrst on page 3-23	0x10	32	RW	0x2	MPU Module Reset Register
permodrst on page 3-24	0x14	32	RW	0x3FFFFFFF	Peripheral Module Reset Register
per2modrst on page 3-26	0x18	32	RW	0xFF	Peripheral 2 Module Reset Register
brgmodrst on page 3-27	0x1C	32	RW	0x7	Bridge Module Reset Register
miscmodrst on page 3-28	0x20	32	RW	0x0	Miscellaneous Module Reset Register

stat

The STAT register contains bits that indicate the reset source or a timeout event. For reset sources, a field is 1 if its associated reset requester caused the reset. For timeout events, a field is 1 if its associated timeout occured as part of a hardware sequenced warm/debug reset. Software clears bits by writing them with a value of 1. Writes to bits with a value of 0 are ignored. After a cold reset is complete, all bits are reset to their reset value except for the bit(s) that indicate the source of the cold reset. If multiple cold reset requests overlap with each other, the source de-asserts the request last will be logged. The other reset request source(s) de-assert the request in the same cycle will also be logged, the rest of the fields are reset to default value of 0. After a warm reset is complete, the bit(s) that indicate the source of the warm reset are set to 1. A warm reset doesn't clear any of the bits in the STAT register; these bits must be cleared by software writing the STAT register.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05000

Offset: 0x0

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			etrstalltimeout RW 0x0	fpgahstimeout RW 0x0	scanhstimeout RW 0x0	fpgamgrhstimeout RW 0x0	sdrselfreftimeout RW 0x0	Reserved				cdbgreqrst RW 0x0	fpgadbgrst RW 0x0	Reserved	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l4wd1rst RW 0x0	l4wd0rst RW 0x0	mpuwd1rst RW 0x0	mpuwd0rst RW 0x0	Reserved	swwarmrst RW 0x0	fpgawarmrst RW 0x0	nrstpinrst RW 0x0	Reserved			swcoldrst RW 0x0	configiocoldrst RW 0x0	fpgacoldrst RW 0x0	nporpinrst RW 0x0	porvoltrst RW 0x0

stat Fields

Bit	Name	Description	Access	Reset
28	etrstalltimeout	A 1 indicates that Reset Manager's request to the ETR (Embedded Trace Router) to stall its AXI master port before starting a hardware sequenced warm reset timed-out and the Reset Manager had to proceed with the warm reset anyway.	RW	0x0
27	fpgahstimeout	A 1 indicates that Reset Manager's handshake request to FPGA before starting a hardware sequenced warm reset timed-out and the Reset Manager had to proceed with the warm reset anyway.	RW	0x0
26	scanhstimeout	A 1 indicates that Reset Manager's request to the SCAN manager to stop driving JTAG clock to FPGA CB before starting a hardware sequenced warm reset timed-out and the Reset Manager had to proceed with the warm reset anyway.	RW	0x0
25	fpgamgrhstimeout	A 1 indicates that Reset Manager's request to the FPGA manager to stop driving configuration clock to FPGA CB before starting a hardware sequenced warm reset timed-out and the Reset Manager had to proceed with the warm reset anyway.	RW	0x0

Bit	Name	Description	Access	Reset
24	`sdrselfreftimeout`	A 1 indicates that Reset Manager's request to the SDRAM Controller Subsystem to put the SDRAM devices into self-refresh mode before starting a hardware sequenced warm reset timed-out and the Reset Manager had to proceed with the warm reset anyway.	RW	0x0
19	`cdbgreqrst`	DAP triggered debug reset	RW	0x0
18	`fpgadbgrst`	FPGA triggered debug reset (f2h_dbg_rst_req_n = 1)	RW	0x0
15	`l4wd1rst`	L4 Watchdog 1 triggered a hardware sequenced warm reset	RW	0x0
14	`l4wd0rst`	L4 Watchdog 0 triggered a hardware sequenced warm reset	RW	0x0
13	`mpuwd1rst`	MPU Watchdog 1 triggered a hardware sequenced warm reset	RW	0x0
12	`mpuwd0rst`	MPU Watchdog 0 triggered a hardware sequenced warm reset	RW	0x0
10	`swwarmrst`	Software wrote CTRL.SWWARMRSTREQ to 1 and triggered a hardware sequenced warm reset	RW	0x0
9	`fpgawarmrst`	FPGA core triggered a hardware sequenced warm reset (f2h_warm_rst_req_n = 1)	RW	0x0
8	`nrstpinrst`	nRST pin triggered a hardware sequenced warm reset	RW	0x0
4	`swcoldrst`	Software wrote CTRL.SWCOLDRSTREQ to 1 and triggered a cold reset	RW	0x0
3	`configiocoldrst`	FPGA entered CONFIG_IO mode and a triggered a cold reset	RW	0x0
2	`fpgacoldrst`	FPGA core triggered a cold reset (f2h_cold_rst_req_n = 1)	RW	0x0
1	`nporpinrst`	nPOR pin triggered a cold reset (por_pin_req = 1)	RW	0x0
0	`porvoltrst`	Built-in POR voltage detector triggered a cold reset (por_voltage_req = 1)	RW	0x0

ctrl

The CTRL register is used by software to control reset behavior.It includes fields for software to initiate the cold and warm reset, enable hardware handshake with other modules before warm reset, and perform software handshake. The software handshake sequence must match the hardware sequence. Software mustde-assert the handshake request after asserting warm reset and before de-assert the warm reset. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								etrstallwarmrst RW 0x0	etrstallack RO 0x0	etrstallreq RW 0x0	etrstallen RW 0x1	Reserved	fpgahsack RO 0x0	fpgahsreq RW 0x0	fpgahsen RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	scanmgrhsack RO 0x0	scanmgrhsreq RW 0x0	scanmgrhsen RW 0x0	Reserved	fpgamgrhsack RO 0x0	fpgamgrhsreq RW 0x0	fpgamgrhsen RW 0x0	Reserved	sdrselfreqack RO 0x0	sdrselfrefreq RW 0x0	sdrselfrefen RW 0x0	Reserved		swwarmrstreq RW 0x0	swcoldrstreq RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
23	etrstallwarmrst	If a warm reset occurs and ETRSTALLEN is 1, hardware sets this bit to 1 to indicate that the stall of the ETR AXI master is pending. Hardware leaves the ETR stalled until software clears this field by writing it with 1. Software must only clear this field when it is ready to have the ETR AXI master start making AXI requests to write trace data.	RW	0x0
22	etrstallack	This is the acknowlege for a ETR AXI master stall initiated by the ETRSTALLREQ field. A 1 indicates that the ETR has stalled its AXI master	RO	0x0

Bit	Name	Description	Access	Reset
21	`etrstallreq`	Software writes this field 1 to request to the ETR that it stalls its AXI master to the L3 Interconnect. Software waits for the ETRSTALLACK to be 1 and then writes this field to 0. Note that it is possible for the ETR to never assert ETRSTALLACK so software should timeout if ETRSTALLACK is never asserted.	RW	0x0
20	`etrstallen`	This field controls whether the ETR is requested to idle its AXI master interface (i.e. finish outstanding transactions and not initiate any more) to the L3 Interconnect before a warm or debug reset. If set to 1, the Reset Manager makes a request to the ETR to stall its AXI master and waits for it to finish any outstanding AXI transactions before a warm reset of the L3 Interconnect or a debug reset of the ETR. This stalling is required because the debug logic (including the ETR) is reset on a debug reset and the ETR AXI master is connected to the L3 Interconnect which is reset on a warm reset and these resets can happen independently.	RW	0x1
18	`fpgahsack`	This is the acknowlege (high active) that the FPGA handshake acknowledge has been received by Reset Manager.	RO	0x0
17	`fpgahsreq`	Software writes this field 1 to initiate handshake request to FPGA . Software waits for the FPGAHSACK to be active and then writes this field to 0. Note that it is possible for the FPGA to never assert FPGAHSACK so software should timeout in this case.	RW	0x0
16	`fpgahsen`	This field controls whether to perform handshake with FPGA before asserting warm reset. If set to 1, the Reset Manager makes a request to the FPGAbefore asserting warm reset signals. However if FPGA is already in warm reset state, the handshake is not performed. If set to 0, the handshake is not performed	RW	0x0
14	`scanmgrhsack`	This is the acknowlege (high active) that the SCAN manager has successfully idled its output clocks.	RO	0x0
13	`scanmgrhsreq`	Software writes this field 1 to request to the SCAN manager to idle its output clocks. Software waits for the SCANMGRHSACK to be 1 and then writes this field to 0. Note that it is possible for the Scan Manager to never assert SCANMGRHSACK (e.g. its input clock is disabled) so software should timeout in this case.	RW	0x0

Bit	Name	Description	Access	Reset
12	scanmgrhsen	Enables a handshake between the Reset Manager and Scan Manager before a warm reset. The handshake is used to warn the Scan Manager that a warm reset it coming so it can prepare for it. When the Scan Manager receives a warm reset handshake, the Scan Manager drives its output clocks to a quiescent state to avoid glitches. If set to 1, the Reset Manager makes a request to the Scan Managerbefore asserting warm reset signals. However if the Scan Manager is already in warm reset, the handshake is skipped. If set to 0, the handshake is skipped.	RW	0x0
10	fpgamgrhsack	This is the acknowlege (high active) that the FPGA manager has successfully idled its output clock.	RO	0x0
9	fpgamgrhsreq	Software writes this field 1 to request to the FPGA Manager to idle its output clock. Software waits for the FPGAMGRHSACK to be 1 and then writes this field to 0. Note that it is possible for the FPGA Manager to never assert FPGAMGRHSACK so software should timeout in this case.	RW	0x0
8	fpgamgrhsen	Enables a handshake between the Reset Manager and FPGA Manager before a warm reset. The handshake is used to warn the FPGA Manager that a warm reset it coming so it can prepare for it. When the FPGA Manager receives a warm reset handshake, the FPGA Manager drives its output clock to a quiescent state to avoid glitches. If set to 1, the Manager makes a request to the FPGA Managerbefore asserting warm reset signals. However if the FPGA Manager is already in warm reset, the handshake is skipped. If set to 0, the handshake is skipped.	RW	0x0
6	sdrselfreqack	This is the acknowlege for a SDRAM self-refresh mode request initiated by the SDRSELFREFREQ field. A 1 indicates that the SDRAM Controller Subsystem has put the SDRAM devices into self-refresh mode.	RO	0x0
5	sdrselfrefreq	Software writes this field 1 to request to the SDRAM Controller Subsystem that it puts the SDRAM devices into self-refresh mode. This is done to preserve SDRAM contents across a software warm reset. Software waits for the SDRSELFREFACK to be 1 and then writes this field to 0. Note that it is possible for the SDRAM Controller Subsystem to never assert SDRSELFREFACK so software should timeout if SDRSELFREFACK is never asserted.	RW	0x0

Bit	Name	Description	Access	Reset
4	`sdrselfrefen`	This field controls whether the contents of SDRAM devices survive a hardware sequenced warm reset. If set to 1, the Reset Manager makes a request to the SDRAM Controller Subsystem to put the SDRAM devices into self-refresh mode before asserting warm reset signals. However, if SDRAM is already in warm reset, Handshake with SDRAM is not performed.	RW	0x0
1	`swwarmrstreq`	This is a one-shot bit written by software to 1 to trigger a hardware sequenced warm reset. It always reads the value 0.	RW	0x0
0	`swcoldrstreq`	This is a one-shot bit written by software to 1 to trigger a cold reset. It always reads the value 0.	RW	0x0

counts

The COUNTS register is used by software to control reset behavior.It includes fields for software to control the behavior of the warm reset and nRST pin. Fields are only reset by a cold reset.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nrstcnt RW 0x800											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nrstcnt RW 0x800								warmrstcycles RW 0x80							

counts Fields

Bit	Name	Description	Access	Reset
27:8	`nrstcnt`	The Reset Manager pulls down the nRST pin on a warm reset for the number of cycles specified in this register. A value of 0x0 prevents the Reset Manager from pulling down the nRST pin.	RW	0x800
7:0	`warmrstcycles`	On a warm reset, the Reset Manager releases the reset to the Clock Manager, and then waits for the number of cycles specified in this register before releasing the rest of the hardware controlled resets. Value must be greater than 16.	RW	0x80

mpumodrst

The MPUMODRST register is used by software to trigger module resets (individual module reset signals). Software explicitly asserts and de-asserts module reset signals by writing bits in the appropriate *MODRST register. It is up to software to ensure module reset signals are asserted for the appropriate length of time and are de-asserted in the correct order. It is also up to software to not assert a module reset signal that would prevent software from de-asserting the module reset signal. For example, software should not assert the module reset to the CPU executing the software. Software writes a bit to 1 to assert the module reset signal and to 0 to de-assert the module reset signal. All fields except CPU1 are only reset by a cold reset. The CPU1 field is reset by a cold reset. The CPU1 field is also reset by a warm reset if not masked by the corresponding MPUWARMMASK field.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											l2 RW 0x0	scuper RW 0x0	wds RW 0x0	cpu1 RW 0x1	cpu0 RW 0x0

mpumodrst Fields

Bit	Name	Description	Access	Reset
4	l2	Resets L2 cache controller	RW	0x0
3	scuper	Resets SCU and peripherals. Peripherals consist of the interrupt controller, global timer, both per-CPU private timers, and both per-CPU watchdogs (except for the Watchdog Reset Status registers).	RW	0x0
2	wds	Resets both per-CPU Watchdog Reset Status registers in MPU.	RW	0x0
1	cpu1	Resets Cortex-A9 CPU1 in MPU. It is reset to 1 on a cold or warm reset. This holds CPU1 in reset until software is ready to release CPU1 from reset by writing 0 to this field. On single-core devices, writes to this field are ignored.On dual-core devices, writes to this field trigger the same sequence as writes to the CPU0 field (except the sequence is performed on CPU1).	RW	0x1

Bit	Name	Description	Access	Reset
0	cpu0	Resets Cortex-A9 CPU0 in MPU. Whe software changes this field from 0 to 1, ittriggers the following sequence: 1. CPU0 reset is asserted. cpu0 clkoff is de-asserted 2. after 32 osc1_clk cycles, cpu0 clkoff is asserted. When software changes this field from 1 to 0, it triggers the following sequence: 1.CPU0 reset is de-asserted. 2. after 32 cycles, cpu0 clkoff is de-asserted. Software needs to wait for at least 64 osc1_clk cycles between each change of this field to keep the proper reset/clkoff sequence.	RW	0x0

permodrst

The PERMODRST register is used by software to trigger module resets (individual module reset signals). Software explicitly asserts and de-asserts module reset signals by writing bits in the appropriate *MODRST register. It is up to software to ensure module reset signals are asserted for the appropriate length of time and are de-asserted in the correct order. It is also up to software to not assert a module reset signal that would prevent software from de-asserting the module reset signal. For example, software should not assert the module reset to the CPU executing the software. Software writes a bit to 1 to assert the module reset signal and to 0 to de-assert the module reset signal. All fields are reset by a cold reset.All fields are also reset by a warm reset if not masked by the corresponding PERWARMMASK field. The reset value of all fields is 1. This holds the corresponding module in reset until software is ready to release the module from reset by writing 0 to its field.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		sdr RW 0x1	dma RW 0x1	gpio2 RW 0x1	gpio1 RW 0x1	gpio0 RW 0x1	can1 RW 0x1	can0 RW 0x1	sdmmc RW 0x1	spis1 RW 0x1	spis0 RW 0x1	spim1 RW 0x1	spim0 RW 0x1	uart1 RW 0x1	uart0 RW 0x1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
i2c3 RW 0x1	i2c2 RW 0x1	i2c1 RW 0x1	i2c0 RW 0x1	sptimer1 RW 0x1	sptimer0 RW 0x1	osc1timer1 RW 0x1	osc1timer0 RW 0x1	l4wd1 RW 0x1	l4wd0 RW 0x1	qspi RW 0x1	nand RW 0x1	usb1 RW 0x1	usb0 RW 0x1	emac1 RW 0x1	emac0 RW 0x1

permodrst Fields

Bit	Name	Description	Access	Reset
29	sdr	Resets SDRAM Controller Subsystem affected by a warm or cold reset.	RW	0x1
28	dma	Resets DMA controller	RW	0x1
27	gpio2	Resets GPIO2	RW	0x1
26	gpio1	Resets GPIO1	RW	0x1
25	gpio0	Resets GPIO0	RW	0x1
24	can1	Resets CAN1 controller. Writes to this field on devices not containing CAN controllers will be ignored.	RW	0x1
23	can0	Resets CAN0 controller. Writes to this field on devices not containing CAN controllers will be ignored.	RW	0x1
22	sdmmc	Resets SD/MMC controller	RW	0x1
21	spis1	Resets SPIS1 controller	RW	0x1
20	spis0	Resets SPIS0 controller	RW	0x1
19	spim1	Resets SPIM1 controller	RW	0x1
18	spim0	Resets SPIM0 controller	RW	0x1
17	uart1	Resets UART1	RW	0x1
16	uart0	Resets UART0	RW	0x1
15	i2c3	Resets I2C3 controller	RW	0x1
14	i2c2	Resets I2C2 controller	RW	0x1
13	i2c1	Resets I2C1 controller	RW	0x1
12	i2c0	Resets I2C0 controller	RW	0x1
11	sptimer1	Resets SP timer 1 connected to L4	RW	0x1
10	sptimer0	Resets SP timer 0 connected to L4	RW	0x1
9	osc1timer1	Resets OSC1 timer 1 connected to L4	RW	0x1

Bit	Name	Description	Access	Reset
8	osc1timer0	Resets OSC1 timer 0 connected to L4	RW	0x1
7	l4wd1	Resets watchdog 1 connected to L4	RW	0x1
6	l4wd0	Resets watchdog 0 connected to L4	RW	0x1
5	qspi	Resets QSPI flash controller	RW	0x1
4	nand	Resets NAND flash controller	RW	0x1
3	usb1	Resets USB1	RW	0x1
2	usb0	Resets USB0	RW	0x1
1	emac1	Resets EMAC1	RW	0x1
0	emac0	Resets EMAC0	RW	0x1

per2modrst

The PER2MODRST register is used by software to trigger module resets (individual module reset signals). Software explicitly asserts and de-asserts module reset signals by writing bits in the appropriate *MODRST register. It is up to software to ensure module reset signals are asserted for the appropriate length of time and are de-asserted in the correct order. It is also up to software to not assert a module reset signal that would prevent software from de-asserting the module reset signal. For example, software should not assert the module reset to the CPU executing the software. Software writes a bit to 1 to assert the module reset signal and to 0 to de-assert the module reset signal. All fields are reset by a cold reset.All fields are also reset by a warm reset if not masked by the corresponding PERWARMMASK field. The reset value of all fields is 1. This holds the corresponding module in reset until software is ready to release the module from reset by writing 0 to its field.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dmaif7 RW 0x1	dmaif6 RW 0x1	dmaif5 RW 0x1	dmaif4 RW 0x1	dmaif3 RW 0x1	dmaif2 RW 0x1	dmaif1 RW 0x1	dmaif0 RW 0x1

per2modrst Fields

Bit	Name	Description	Access	Reset
7	dmaif7	Resets DMA channel 7 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
6	dmaif6	Resets DMA channel 6 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
5	dmaif5	Resets DMA channel 5 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
4	dmaif4	Resets DMA channel 4 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
3	dmaif3	Resets DMA channel 3 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
2	dmaif2	Resets DMA channel 2 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
1	dmaif1	Resets DMA channel 1 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1
0	dmaif0	Resets DMA channel 0 interface adapter between FPGA Fabric and HPS DMA Controller	RW	0x1

brgmodrst

The BRGMODRST register is used by software to trigger module resets (individual module reset signals). Software explicitly asserts and de-asserts module reset signals by writing bits in the appropriate *MODRST register. It is up to software to ensure module reset signals are asserted for the appropriate length of time and are de-asserted in the correct order. It is also up to software to not assert a module reset signal that would prevent software from de-asserting the module reset signal. For example, software should not assert the module reset to the CPU executing the software. Software writes a bit to 1 to assert the module reset signal and to 0 to de-assert the module reset signal. All fields are reset by a cold reset.All fields are also reset by a warm reset if not masked by the corresponding BRGWARMMASK field. The reset value of all fields is 1. This holds the corresponding module in reset until software is ready to release the module from reset by writing 0 to its field.

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD0501C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													fpga2hps RW 0x1	lwhps2fpga RW 0x1	hps2fpga RW 0x1

brgmodrst Fields

Bit	Name	Description	Access	Reset
2	fpga2hps	Resets FPGA2HPS Bridge	RW	0x1
1	lwhps2fpga	Resets LWHPS2FPGA Bridge	RW	0x1
0	hps2fpga	Resets HPS2FPGA Bridge	RW	0x1

miscmodrst

The MISCMODRST register is used by software to trigger module resets (individual module reset signals). Software explicitly asserts and de-asserts module reset signals by writing bits in the appropriate *MODRST register. It is up to software to ensure module reset signals are asserted for the appropriate length of time and are de-asserted in the correct order. It is also up to software to not assert a module reset signal that would prevent software from de-asserting the module reset signal. For example, software should not assert the module reset to the CPU executing the software. Software writes a bit to 1 to assert the module reset signal and to 0 to de-assert the module reset signal. All fields are only reset by a cold reset

Module Instance	Base Address	Register Address
rstmgr	0xFFD05000	0xFFD05020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															sdrcold RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tapcold RW 0x0	dbg RW 0x0	sysdbg RW 0x0	frzctrlcold RW 0x0	scanmgr RW 0x0	clkmgrcold RW 0x0	timestampcold RW 0x0	nrstpin RW 0x0	s2fcold RW 0x0	s2f RW 0x0	acpidmap RW 0x0	fpgamgr RW 0x0	sysmgrcold RW 0x0	sysmgr RW 0x0	ocram RW 0x0	rom RW 0x0

miscmodrst Fields

Bit	Name	Description	Access	Reset
16	sdrcold	Resets logic in SDRAM Controller Subsystem affected only by a cold reset.	RW	0x0
15	tapcold	Resets portion of DAP JTAG TAP controller no reset by a debug probe reset (i.e. nTRST pin). Cold reset only.	RW	0x0
14	dbg	Resets logic located only in the debug domain.	RW	0x0
13	sysdbg	Resets logic that spans the system and debug domains.	RW	0x0
12	frzctrlcold	Resets Freeze Controller in System Manager (cold reset only)	RW	0x0
11	scanmgr	Resets Scan Manager	RW	0x0
10	clkmgrcold	Resets Clock Manager (cold reset only)	RW	0x0
9	timestampcold	Resets debug timestamp to 0 (cold reset only)	RW	0x0
8	nrstpin	Pulls nRST pin low	RW	0x0
7	s2fcold	Resets logic in FPGA core that is only reset by a cold reset (ignores warm reset) (h2f_cold_rst_n = 1)	RW	0x0
6	s2f	Resets logic in FPGA core that doesn't differentiate between HPS cold and warm resets (h2f_rst_n = 1)	RW	0x0
5	acpidmap	Resets ACP ID Mapper	RW	0x0

Bit	Name	Description	Access	Reset
4	`fpgamgr`	Resets FPGA Manager	RW	0x0
3	`sysmgrcold`	Resets logic in System Manager that is only reset by a cold reset (ignores warm reset)	RW	0x0
2	`sysmgr`	Resets logic in System Manager that doesn't differentiate between cold and warm resets	RW	0x0
1	`ocram`	Resets On-chip RAM	RW	0x0
0	`rom`	Resets Boot ROM	RW	0x0

Document Revision History

Table 3-10: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Updated "Functional Description of Reset Manager" • Added address map and register descriptions
February 2014	2014.02.28	Updated content in sections: • Reset Sequencing • Warm Reset Assertion Sequence
December 2013	2013.12.30	Minor formatting issues
November 2012	1.2	• Added cold and warm reset timing diagrams. • Minor updates.
May 2012	1.1	Added reset controller, functional description, and address map and register definitions sections.
January 2012	1.0	Initial release.

FPGA Manager 4

2014.06.30

cv_54013 Subscribe Send Feedback

The FPGA manager in the hard processor system (HPS) manages and monitors the FPGA portion of the system on a chip (SoC) device. The FPGA manager can configure the FPGA fabric from the HPS, monitor the state of the FPGA, and drive or sample signals to or from the FPGA fabric.

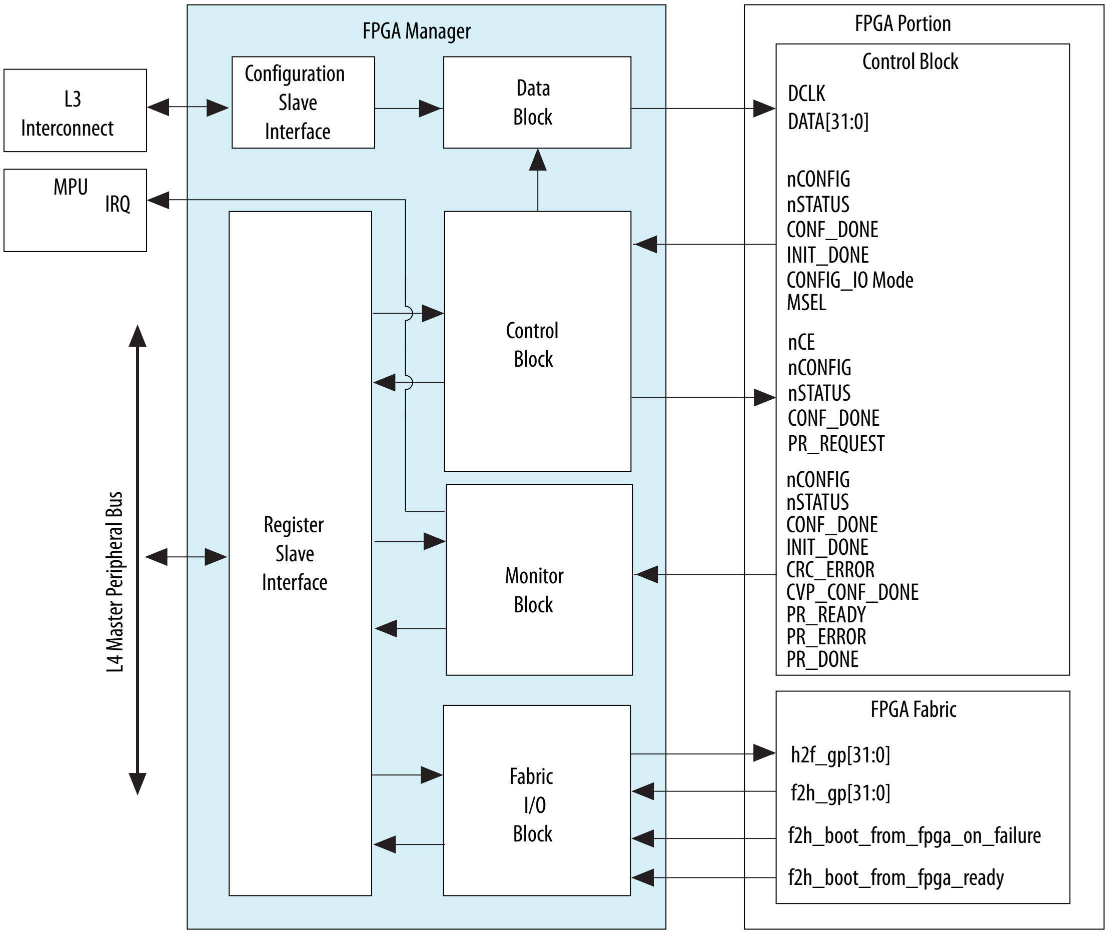
Features of the FPGA Manager

The FPGA manager provides the following functionality and features:

- Full configuration and partial reconfiguration of the FPGA portion of the SoC device
- Drives 32 general-purpose output signals to the FPGA fabric
- Receives 32 general-purpose input signals from the FPGA fabric
- Receives two boot handshaking input signals from the FPGA fabric (used when the HPS boots from the FPGA)
- Monitors the FPGA configuration and power status
- Generates interrupts based on the FPGA status changes
- Can reset the FPGA

FPGA Manager Block Diagram and System Integration

Figure 4-1: FPGA Manager Block Diagram



The register slave interface connects to the level 4 (L4) master peripheral bus for control and status register (CSR) access. The configuration slave interface connects to the level 3 (L3) interconnect for the microprocessor unit (MPU) subsystem or other masters to write the FPGA configuration image to the FPGA control block (CB) when configuring the FPGA portion of the SoC device.

The general-purpose I/O and boot handshaking input interfaces connect to the FPGA fabric. The FPGA manager also connects to the FPGA CB signals to monitor and control the FPGA portion of the device.

The FPGA manager consists of the following blocks:

- Configuration slave interface—accepts and transfers the configuration image to the data interface.
- Register slave interface—accesses the CSRs in the FPGA manager.
- Data—accepts the FPGA configuration image from the configuration slave interface and sends it to the FPGA CB.
- Control—controls the FPGA CB.
- Monitor—monitors the configuration signals in the FPGA CB and sends interrupts to the MPU subsystem.
- Fabric I/O—reads and writes signals from or to the FPGA fabric.

Functional Description of the FPGA Manager

FPGA Manager Building Blocks

The FPGA manager has the two blocks - fabric I/O and monitor. These blocks are to monitor the signals coming from the FPGA portion of the device.

Related Information

FPGA Manager Address Map and Register Definitions on page 4-9

Fabric I/O

The fabric I/O block contains the following registers to allow simple low-latency communication between the HPS and the FPGA fabric:

- General-purpose input register (gpi)
- General-purpose output register (gpo)
- Boot handshaking input register (misci)

These registers are only valid when the FPGA is in user mode. Reading from these registers while the FPGA is not in user mode provides undefined data.

The 32 general-purpose input signals from the FPGA fabric are read by reading the gpi register using the register slave interface. The 32 general-purpose output signals to the FPGA fabric are generated from writes to the gpo register. For more information about FPGA manager registers, refer to "C**FPGA Manager Address Map and Register Definitions" on page 13–7.

The boot handshaking input signals from the FPGA fabric are read by reading the misci register. The f2h_boot_from_fpga_ready signal indicates to the boot ROM when logic in the FPGA fabric is ready to accept configuration interface requests from the HPS-to-FPGA bridge when the boot ROM is booting from the FPGA. The f2h_boot_from_fpga_on_failure signal serves as a fallback in the event that the boot ROM code fails to boot from the primary boot flash device. In this case, the boot ROM code checks these two handshaking signals to determine if it should use the boot code hosted in the FPGA memory as the next stage in the boot process.

There is no interrupt support for this block.

Monitor

The monitor block is an instance of the Synopsys DesignWareGPIO IP (DW_apb_gpio), which is a separate instance of the IP that comprises the three HPS GPIO interfaces. The monitor block connects to the configuration signals in the FPGA. This block monitors key signals related to FPGA configuration such as INIT_DON E, CRC_ERROR, and PR_DONE. Software configures the monitor block through the register slave interface, and can either poll FPGA signals or be interrupted. The **mon** address map within the FPGA manager register address map contains the monitor registers. For more information about FPGA manager registers, refer to **Figure 4-1**

You can program the FPGA manager to treat any of the monitor signals as interrupt sources. Independent of the interrupt source type, the monitor block always drives an active-high level interrupt to the MPU. Each interrupt source can be of the following types:

- Active-high level
- Active-low level
- Rising edge
- Falling edge

FPGA Configuration

You can configure the FPGA using an external device or through the HPS. This section covers configuring the FPGA through the HPS.

For information about configuring the FPGA using an external device, refer to the *Configuration, Design Security, and Remote System Upgrades* chapter in the Cyclone V Device Handbook, Volume 1.

The FPGA CB uses the FPGA mode select (`MSEL`) pins to determine which configuration scheme to use. The `MSEL` pins must be tied to the appropriate values for the configuration scheme. The table below lists supported `MSEL` values when the FPGA is configured by the HPS.

HPS software sets the clock-to-data ratio field (`cdratio`) and configuration data width bit (`cfgwdth`) in the control register (`ctrl`) to match the `MSEL` pins. The `cdratio` field and `cfgwdth` bit must be set before the start of configuration.

The FPGA manager connects to the configuration logic in the FPGA portion of the device using a mode similar to how external logic (for example, MAX II or an intelligent host) configures the FPGA in fast passive parallel (FPP) mode. FPGA configuration through the HPS supports all the capabilities of FPP mode, including the following items:

- FPGA configuration
- Partial FPGA reconfiguration
- FPGA I/O configuration, followed by PCI Express® (PCIe®) configuration of the remainder of FPGA
- External single event upset (SEU) scrubbing
- Decompression
- Advanced Encryption Standard (AES) encryption
- FPGA `DCLK` clock used for initialization phase clock

Note: The FPGA manager supports a data width of 32 or 16 bits. When configuring the FPGA fabric from the HPS, Altera recommends that you always set the data width to 32 bits. For partial reconfiguration, the 16-bit data width is the only option.

The following table lists the supported configuration schemes and their respective MSEL and control register settings when the HPS configures the FPGA.

[4] For information about POR delay, refer to the Configuration, Design Security, and Remote System Upgrades in the Cyclone V Device Handbook, Volume 1.

[5] Other MSEL values are allowed when the FPGA is configured from a non-HPS source. For information, refer to the Configuration, DesignSecurity, and Remote System Upgrades in the Cyclone V Device Handbook, Volume 1.

[6] You can select to enable or disable this feature.

[7] You can select to enable or disable this feature.

Table 4-1: Configuration Schemes for FPGA Configuration by the HPS

Configuration Scheme	Compression Feature	Design Security Feature	POR Delay [4]	MSEL[4..0] [5]	cfgwdth	cdratio	Supports Partial Reconfiguration
FPP ×16	Disabled	AES Disabled	Fast	00000	0	1	Yes
			Standard	00100	0	1	No
	Disabled	AES Enabled	Fast	00001	0	2	Yes
			Standard	00101	0	2	No
	Enabled	Optional [6]	Fast	00010	0	4	Yes
			Standard	00110	0	4	No
FPP ×32	Disabled	AES Disabled	Fast	01000	1	1	No
			Standard	01100	1	1	No
	Disabled	AES Enabled	Fast	01001	1	4	No
			Standard	01101	1	4	No
	Enabled	Optional [7]	Fast	01010	1	8	No
			Standard	01110	1	8	No

Configuring the FPGA portion of the SoC device comprises the following phases:

1. Power up phase
2. Reset phase
3. Configuration phase
4. Initialization phase
5. User mode

Related Information

http://www.altera.com/literature/hb/cyclone-v/cv_52007.pdf
For more information about configuring the FPGA through the HPS, refer to the *Configuration, Design Security, and Remote System Upgrade* appendix in the Cyclone V Device Handbook, Volume 1.

Power Up Phase

In this phase, the VCC is ramping up and has yet to reach normal levels. This phase completes when the on-chip voltage detector determines that the VCC has reached normal levels.

Reset Phase

The FPGA manager resets the FPGA portion of the SoC device when the FPGA configuration signal (`nCONFIG`) is driven low. The HPS configures the FPGA by writing a 1 to the `nconfigpull` bit of the `ctrl` register. This action causes the FPGA portion of the device to reset and perform the following actions:

1. Clear the FPGA configuration RAM bits
2. Tri-state all FPGA user I/O pins
3. Pull the `nSTATUS` and `CONF_DONE` pins low
4. Use the FPGA CB to read the values of the `MSEL` pins to determine the configuration scheme

The `nconfigpull` bit of the `ctrl` register needs to be set to 0 when the FPGA has successfully entered the reset phase. Setting the bit releases the FPGA from the reset phase and transitions to the configuration phase.

Note: You must set the `cdratio` and `cfgwdth` bits of the `ctrl` register appropriately before the FPGA enters the reset phase.

Configuration Phase

To configure the FPGA using the HPS, software sets the `axicfgen` bit of the `ctrl` register to 1. Software then sends configuration data to the FPGA by writing data to the write data register (`data`) in the FPGA manager module configuration data address map. Software polls the `CONF_DONE` pin by reading the `gpio_instatus` register to determine if the FPGA configuration is successful. When configuration is successful, software sets the `axicfgen` bit of the `ctrl` register to 0. The FPGA user I/O pins are still tri-stated in this phase.

After successfully completing the configuration phase, the FPGA transitions to the initialization phase. To delay configuring the FPGA, set the `confdonepull` bit of the `ctrl` register to 1.

Related Information

Booting and Configuration Introduction on page 30-1
For more information about configuring the FPGA through the HPS, refer to the *Booting and Configuration* appendix.

Initialization Phase

In this phase, the FPGA prepares to enter user mode. The internal oscillator in the FPGA portion of the device is the default clock source for the initialization phase. Alternatively, the configuration image can specify the `CLKUSR` or the `DCLK` pins as the clock source. The alternate clock source controls when the FPGA enters user mode.

If `DCLK` is selected as the clock source, software uses the `DCLK` count (`dclkcnt`) register to drive `DCLK` pulses to the FPGA. Writing to the `cnt` field of the `dclkcnt` register triggers the FPGA manager to generate the specified number of `DCLK` pulses. When all of the `DCLK` pulses have been sent, the `dcntdone` bit of the `DCLK` status (`dclkstat`) register is set to 1. Software polls the `dcntdone` bit to know when all of the `DCLK` pulses have been sent.

Note: Before another write to the `dclkcnt` register, software needs to write a value of 1 to the `dcntdone` bit to clear the done state.

The FPGA user I/O pins are still tri-stated in this phase. When the initialization phase completes, the FPGA releases the optional `INIT_DONE` pin and an external resistor pulls the pin high.

User Mode

The FPGA enters the user mode after exiting the initialization phase. The FPGA user I/O pins are no longer tri-stated in this phase and the configured soft logic in the FPGA becomes active.

The FPGA remains in user mode until the `nCONFIG` pin is driven low. If the `nCONFIG` pin is driven low, the FPGA reenters the reset phase. The internal oscillator is disabled in user mode, but is enabled as soon as the `nCONFIG` pin is driven low.

Related Information

- **http://www.altera.com/literature/hb/cyclone-v/cv_52007.pdf**
  For more information about configuring the FPGA through the HPS, refer to the *Configuration, Design Security, and Remote System Upgrade* appendix in the Cyclone V Device Handbook, Volume 1.
- **Booting and Configuration Introduction** on page 30-1
  For more information about configuring the FPGA through the HPS, refer to the *Booting and Configuration* appendix.

FPGA Status

Configuration signals from the FPGA CSS such as INIT_DONE, CRC_ERROR and PR_DONE are monitored by the FPGA Manager. Software configures the monitor block through the register slave interface, and can either poll FPGA signals or be interrupted. Monitored signals can be read through an unmasked status register as well as a masked status register. Each of the monitored signals can generate an interrupt to the MPU Global Interrupt Controller. Each interrupt source can be enabled and the polarity of the signals generating the interrupt can also be selected through registers in the FPGA Manager.

Error Message Extraction

Cyclic redundancy check (CRC) errors from the FPGA fabric are monitored by the FPGA Manager. Upon assertion of a CRC error signal from the FPGA, the FPGA Manager extracts information about the error including:

- Error syndrome
- Error location
- Error type

A CRC error interrupt from the FPGA Manager can be enabled via software. Software can then extract the CRC error information from the EMR data registers. The number of valid error information bits in the EMR data registers depends on the specific FPGA device.

Data AES Decryption

The Configuration Data Interface can also be used to transmit data from the HPS to the CSS Advanced Encryption Standard (AES) decryption engine. Software should sample the status registers in FPGA manager before switching modes between FPGA configuration and AES decryption. The FPGA manager does not do anything specific to handle a situation where software starts an AES decryption while in the middle of an FPGA configuration initiated by HPS, and the end result in this case is undefined.

JTAG Hosting

The FPGA Manager has the capability to host the JTAG interface to the CSS. In this mode the FPGA Manger will take over the FPGA JTAG interface to CSS via mux overrides thus bypassing the JTAG device pins. This interface is disabled by default. When JTAG hosting is turned on the FPGA Manager also provides a JTAG interface to the FPGA core which could be used by soft logic in the FPGA.

JTAG hosting provides the capability for:

- Secure remote debug
- Secure remote key insertion
- Remote FPGA configuration

Boot Handshake

There are two input signals from the FPGA to control HPS boot from the FPGA. Both are synchronized within the FPGA Manager. Boot software will read these signals before accessing a boot image in the FPGA. The following table describes the functionality of these signals.

Signal	Description
f2h_boot_from_fpga_on_failure	Indicates whether a fallback preloader image is available in an FPGA on-chip RAM at memory location 0. The fallback preloader image is to be used only if the HPS boot ROM does not find a valid preloader image in the selected flash memory device.
f2h_boot_from_fpga_ready	Indicates a preloader image is available in an FPGA on-chip RAM at memory location 0 and it is ready to be accessed.

General Purpose I/O

Thirty-two general purpose inputs and thirty-two general purpose outputs are provided to the FPGA and are controlled through registers in the FPGA Manager.

No interrupts are generated through the input pins. All inputs are synchronized within the FPGA Manager. Output signals should be synchronized in the FPGA.

Clock

The FPGA manager has two clock input signals which are asynchronous to each other. The clock manager generates these two clocks:

- cfg_clk—the configuration slave interface clock input and also the DCLK output reference for FPGA configuration. Enable this clock in the clock manager only when configuration is active or when the configuration slave interface needs to respond to master requests.
- l4_mp_clk—the register slave interface clock.

Related Information

Clock Manager on page 2-1

Reset

The FPGA manager has one reset signal. The reset manager drives this signal to FPGA manager on a cold or warm reset. All distributed reset signals in the FPGA manager are asserted asynchronously at the same time and de-asserted synchronously to their associated clocks.

Related Information

Reset Manager on page 3-1

FPGA Manager Address Map and Register Definitions

The address map and register definitions for the FPGA Manager consist of the following regions:

- FPGA Manager Module
- FPGA Manager Module Configuration Data

Related Information

- FPGA Manager Module Configuration Data Address Map on page 4-9
- FPGA Manager Module Address Map on page 4-10
- Configuration Monitor (MON) Registers Register Descriptions on page 4-21
- Introduction to Cyclone V Hard Processor System on page 1-1

FPGA Manager Module Configuration Data Address Map

Registers in the FPGA Manager module accessible via its AXI slave

Base Address: `0xFFB90000`

FPGA Manager Module Configuration Data

Register	Offset	Width	Acces s	Reset Value	Description
data on page 4-9	0x0	32	RW	0x0	Write Data Register

data

Used to send configuration image to FPGA. The DATA register accepts 4 bytes of the configuration image on each write. The configuration image byte-stream is converted into a 4-byte word with little-endian ordering. If the configuration image is not an integer multiple of 4 bytes, software should pad the configuration image with extra zero bytes to make it an integer multiple of 4 bytes. The FPGA Manager converts the DATA to 16 bits wide when writing CB.DATA for partial reconfiguration. The FPGA Manager waits to transmit the data to the CB until the FPGA is able to receive it. For a full configuration, the FPGA Manager waits until the FPGA exits the Reset Phase and enters the Configuration Phase. For a partial reconfiguration, the FPGA Manager waits until the CB.PR_READY signal indicates that the FPGA is ready.

Module Instance	Base Address	Register Address
fpgamgrdata	0xFFB90000	0xFFB90000

Offset: `0x0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

data Fields

Bit	Name	Description	Access	Reset
31:0	value	Accepts configuration image to be sent to CB when the HPS configures the FPGA. Software normally just writes this register. If software reads this register, it returns the value 0 and replies with an AXI SLVERR error.	RW	0x0

FPGA Manager Module Address Map

Registers in the FPGA Manager module accessible via its APB slave

Base Address: 0xFF706000

FPGA Manager Module

Register	Offset	Width	Access	Reset Value	Description
stat on page 4-11	0x0	32	RW	0x45	Status Register
ctrl on page 4-14	0x4	32	RW	0x200	Control Register
dclkcnt on page 4-17	0x8	32	RW	0x0	DCLK Count Register
dclkstat on page 4-18	0xC	32	RW	0x0	DCLK Status Register
gpo on page 4-19	0x10	32	RW	0x0	General-Purpose Output Register
gpi on page 4-19	0x14	32	RO	0x0	General-Purpose Input Register
misci on page 4-20	0x18	32	RO	0x0	Miscellaneous Input Register

Configuration Monitor (MON) Registers

Register	Offset	Width	Access	Reset Value	Description
gpio_inten on page 4-23	0x830	32	RW	0x0	Interrupt Enable Register
gpio_intmask on page 4-25	0x834	32	RW	0x0	Interrupt Mask Register

Register	Offset	Width	Access	Reset Value	Description
gpio_inttype_level on page 4-28	0x838	32	RW	0x0	Interrupt Level Register
gpio_int_polarity on page 4-31	0x83C	32	RW	0x0	Interrupt Polarity Register
gpio_intstatus on page 4-34	0x840	32	RO	0x0	Interrupt Status Register
gpio_raw_intstatus on page 4-36	0x844	32	RO	0x0	Raw Interrupt Status Register
gpio_porta_eoi on page 4-39	0x84C	32	WO	0x0	Clear Interrupt Register
gpio_ext_porta on page 4-41	0x850	32	RO	0x0	External Port A Register
gpio_ls_sync on page 4-42	0x860	32	RW	0x0	Synchronization Level Register
gpio_ver_id_code on page 4-43	0x86C	32	RO	0x3230382A	GPIO Version Register
gpio_config_reg2 on page 4-44	0x870	32	RO	0x39CEB	Configuration Register 2
gpio_config_reg1 on page 4-45	0x874	32	RO	0x1F50F2	Configuration Register 1

stat

Provides status fields for software for the FPGA Manager. The Mode field tells software what configuration phase the FPGA currently is in. For regular configuration through the PINs or through the HPS, these states map directly to customer configuration documentation. For Configuration Via PCI Express (CVP), the IOCSR configuration is done through the PINS or through HPS. Then the complete configuration is done through the PCI Express Bus. When CVP is being done, InitPhase indicates only IOCSR configuration has completed. CVP_CONF_DONE is available in the CB Monitor for observation by software. The MSEL field provides a read only register for software to read the MSEL value driven from the external pins.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								msel RO 0x8					mode RW 0x5		

stat Fields

Bit	Name	Description	Access	Reset
7:3	msel	This read-only field allows software to observe the MSEL inputs from the device pins. The MSEL pins define the FPGA configuration mode. **Value** **Description** 0x0 16-bit Passive Parallel with Fast Power on Reset Delay; No AES Encryption; No Data Compression. CDRATIO must be programmed to x1 0x1 16-bit Passive Parallel with Fast Power on Reset Delay; With AES Encryption; No Data Compression. CDRATIO must be programmed to x4 0x2 16-bit Passive Parallel with Fast Power on Reset Delay; AES Optional; With Data Compression. CDRATIO must be programmed to x8 0x3 Reserved 0x4 16-bit Passive Parallel with Slow Power on Reset Delay; No AES Encryption; No Data Compression. CDRATIO must be programmed to x1 0x5 16-bit Passive Parallel with Slow Power on Reset Delay; With AES Encryption; No Data Compression. CDRATIO must be programmed to x4 0x6 16-bit Passive Parallel with Slow Power on Reset Delay; AES Optional; With Data Compression. CDRATIO must be programmed to x8 0x7 Reserved 0x8 32-bit Passive Parallel with Fast Power on Reset Delay; No AES Encryption; No Data Compression. CDRATIO must be programmed to x1 0x9 32-bit Passive Parallel with Fast Power on Reset Delay; With AES Encryption; No Data Compression. CDRATIO must be programmed to x4 0xa 32-bit Passive Parallel with Fast Power on Reset Delay; AES Optional; With Data Compression. CDRATIO must be programmed to x8 0xb Reserved 0xc 32-bit Passive Parallel with Slow Power on Reset Delay; No AES Encryption; No Data Compression. CDRATIO must be programmed to x1 0xd 32-bit Passive Parallel with Slow Power on Reset Delay; With AES Encryption; No Data Compression. CDRATIO must be	RO	0x8

Bit	Name	Description	Access	Reset
2:0	`mode`	Reports FPGA state **Value** **Description** 0x0 FPGA Powered Off 0x1 FPGA in Reset Phase 0x2 FPGA in Configuration Phase 0x3 FPGA in Initialization Phase. In CVP configuration, this state indicates IO configuration has completed. 0x4 FPGA in User Mode 0x5 FPGA state has not yet been determined. This only occurs briefly after reset.	`RW`	`0x5`

ctrl

Allows HPS to control FPGA configuration. The NCONFIGPULL, NSTATUSPULL, and CONFDONEPULL fields drive signals to the FPGA Control Block that are logically ORed into their respective pins. These signals are always driven independent of the value of EN. The polarity of the NCONFIGPULL, NSTATUSPULL, and CONFDONEPULL fields is inverted relative to their associated pins. The MSEL (external pins), CDRATIO and CFGWDTH signals determine the mode of operation for Normal Configuration. For Partial Reconfiguration, CDRATIO is used to set the appropriate clock to data ratio, and CFGWDTH should always be set to 16-bit Passive Parallel. AXICFGEN is used to enable transfer of configuration data by enabling or disabling DCLK during data transfers.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF706004`

Offset: `0x4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						cfgwdth RW 0x1	axicfgen RW 0x0	cdratio RW 0x0		prreq RW 0x0	confdonepull RW 0x0	nstatuspull RW 0x0	nconfigpull RW 0x0	nce RW 0x0	en RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
9	cfgwdth	This field determines the Configuration Passive Parallel data bus width when HPS configures the FPGA. Only 32-bit Passive Parallel or 16-bit Passive Parallel are supported. When HPS does Normal Configuration, configuration should use 32-bit Passive Parallel Mode. The external pins MSEL must be set appropriately for the configuration selected. For Partial Reconfiguration, 16-bit Passive Parallel must be used. **Value** — **Description** 0x0 — 16-bit Passive Parallel 0x1 — 32-bit Passive Parallel	RW	0x1
8	axicfgen	There are strict SW initialization steps for configuration, partial configuration and error cases. When SW is sending configuration files, this bit must be set before the file is transferred on the AXI bus. This bit enables the DCLK during the AXI configuration data transfers. Note, the AXI and configuration datapaths remain active irregardless of the state of this bit. Simply, if the AXI slave is enabled, the DCLK to the CB will be active. If disabled, the DCLK to the CB will not be active. So AXI transfers destined to the FPGA Manager when AXIEN is 0, will complete normally from the HPS perspective. This field only affects the FPGA if CTRL.EN is 1. **Value** — **Description** 0x0 — Incoming AXI data transfers will be ignored. DCLK will not toggle during data transfer. 0x1 — AXI data transfer to CB is active. DCLK will toggle during data transfer.	RW	0x0

Bit	Name	Description	Access	Reset
7:6	`cdratio`	This field controls the Clock to Data Ratio (CDRATIO) for Normal Configuration and Partial Reconfiguration data transfer from the AXI Slave to the FPGA. For Normal Configuration, the value in this field must be set to be consistent to the implied CD ratio of the MSEL setting. For Partial Reconfiguration, the value in this field must be set to the same clock to data ratio in the options bits in the Normal Configuration file. **Value** / **Description** 0x0 CDRATIO of 1 0x1 CDRATIO of 2 0x2 CDRATIO of 4 0x3 CDRATIO of 8	`RW`	`0x0`
5	`prreq`	This field is used to assert PR_REQUEST to request partial reconfiguration while the FPGA is in User Mode. This field only affects the FPGA if CTRL.EN is 1. **Value** / **Description** 0x0 De-assert PR_REQUEST (driven to 0). 0x1 Assert PR_REQUEST (driven to 1).	`RW`	`0x0`
4	`confdonepull`	Pulls down CONF_DONE input to the CB **Value** / **Description** 0x0 Don't pull-down CONF_DONE input to the CB. 0x1 Pull-down CONF_DONE input to the CB.	`RW`	`0x0`
3	`nstatuspull`	Pulls down nSTATUS input to the CB **Value** / **Description** 0x0 Don't pull-down nSTATUS input to the CB. 0x1 Pull-down nSTATUS input to the CB.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
2	`nconfigpull`	The nCONFIG input is used to put the FPGA into its reset phase. If the FPGA was configured, its operation stops and it will have to be configured again to start operation. **Value** **Description** 0x0 Don't pull-down nCONFIG input to the CB. 0x1 Pull-down nCONFIG input to the CB. This puts the FPGA in reset phase and restarts configuration.	RW	0x0
1	`nce`	This field drives the active-low Chip Enable (nCE) signal to the CB. It should be set to 0 (configuration enabled) before CTRL.EN is set. This field only effects the FPGA if CTRL.EN is 1. **Value** **Description** 0x0 Configuration is enabled. The nCE to the CB is driven to 0. 0x1 Configuration is disabled. The nCE to the CB is driven to 1.	RW	0x0
0	`en`	Controls whether the FPGA configuration pins or HPS FPGA Manager drive configuration inputs to the CB. **Value** **Description** 0x0 FPGA configuration pins drive configuration inputs to the CB. Used when FPGA is configured by means other than the HPS. 0x1 FPGA Manager drives configuration inputs to the CB. Used when HPS configures the FPGA.	RW	0x0

dclkcnt

Used to give software control in enabling DCLK at any time. SW will need control of the DCLK in specific configuration and partial reconfiguration initialization steps to send spurious DCLKs required by the CB. SW takes ownership for DCLK during normal configuration, partial reconfiguration, error scenerio handshakes including SEU CRC error during partial reconfiguration, SW early abort of partial reconfiguration, and initializatin phase DCLK driving. During initialization phase, a configuration image loaded into the FPGA can request that DCLK be used as the initialization phase clock instead of the default internal oscillator or optionally the CLKUSR pin. In the case that DCLK is requested, the DCLKCNT register is used by software to control DCLK during the initialization phase. Software should poll the DCLKSTAT.DCNTDONE write one to clear register to be set when the correct number of

DCLKs have completed. Software should clear DCLKSTAT.DCNTDONE before writing to the DCLKCNT register again. This field only affects the FPGA if CTRL.EN is 1.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF706008`

Offset: `0x8`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">cnt
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">cnt
RW 0x0</td></tr>
</table>

dclkcnt Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Controls DCLK counter. Software writes a non-zero value into CNT and the FPGA Manager generates the specified number of DCLK pulses and decrements COUNT. This register will read back the original value written by software. Software can write CNT at any time.	`RW`	`0x0`

dclkstat

This write one to clear register indicates that the DCLKCNT has counted down to zero. The DCLKCNT is used by software to drive spurious DCLKs to the FPGA. Software will poll this bit after writing DCLKCNT to know when all of the DCLKs have been sent.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF70600C`

Offset: `0xC`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>dcntdone
RW 0x0</td></tr>
</table>

dclkstat Fields

Bit	Name	Description	Access	Reset
0	`dcntdone`	This bit is write one to clear. This bit gets set after the DCLKCNT has counted down to zero (transition from 1 to 0). **Value** **Description** 0x0 DCLKCNT is still counting down. 0x1 DCLKCNT is done counting down.	RW	0x0

gpo

Provides a low-latency, low-performance, and simple way to drive general-purpose signals to the FPGA fabric.

Module Instance	Base Address	Register Address
`fpgamgrregs`	0xFF706000	0xFF706010

Offset: `0x10`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

gpo Fields

Bit	Name	Description	Access	Reset
31:0	`value`	Drives h2f_gp[31:0] with specified value. When read, returns the current value being driven to the FPGA fabric.	RW	0x0

gpi

Provides a low-latency, low-performance, and simple way to read general-purpose signals driven from the FPGA fabric.

Module Instance	Base Address	Register Address
`fpgamgrregs`	0xFF706000	0xFF706014

Offset: `0x14`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

gpi Fields

Bit	Name	Description	Access	Reset
31:0	value	The value being driven from the FPGA fabric on f2h_gp[31:0]. If the FPGA is not in User Mode, the value of this field is undefined.	RO	0x0

misci

Provides a low-latency, low-performance, and simple way to read specific handshaking signals driven from the FPGA fabric.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706018

Offset: 0x18

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														bootFPGArdy RO 0x0	bootFPGAfail RO 0x0

misci Fields

Bit	Name	Description	Access	Reset
1	bootFPGArdy	The value of the f2h_boot_from_fpga_ready signal from the FPGA fabric. If the FPGA is not in User Mode, the value of this field is undefined. 1 = FPGA fabric is ready to accept AXI master requests from the HPS2FPGA bridge. 0 = FPGA fabric is not ready (probably still processing a reset).	RO	0x0

Bit	Name	Description	Access	Reset
0	bootFPGAfail	The value of the f2h_boot_from_fpga_on_failure signal from the FPGA fabric. If the FPGA is not in User Mode, the value of this field is undefined. 1 = Boot ROM will boot from FPGA if boot from normal boot device fails. 0 = Boot ROM will not boot from FPGA if boot from normal boot device fails.	RO	0x0

Configuration Monitor (MON) Registers Register Descriptions

The Configuration Monitor allows software to poll or be interrupted by changes in the FPGA state. The Configuration Monitor is an instantiation of a Synopsys GPIO.

Only registers relevant to the MON operation are shown. The GPIO inputs are connected to the following signals:

- nSTATUS - Driven to 0 by the FPGA in this device if the FPGA is in Reset Phase or if the FPGA detected an error during the Configuration Phase.
- CONF_DONE - Driven to 0 by the FPGA in this device during the Reset Phase and driven to 1 when the FPGA Configuration Phase is done.
- INIT_DONE - Driven to 0 by the FPGA in this device during the Configuration Phase and driven to 1 when the FPGA Initialization Phase is done.
- CRC_ERROR - CRC error indicator. A CRC_ERROR value of 1 indicates that the FPGA detected a CRC error while in User Mode.
- CVP_CONF_DONE - Configuration via PCIe done indicator. A CVP_CONF_DONE value of 1 indicates that CVP is done.
- PR_READY - Partial reconfiguration ready indicator. A PR_READY value of 1 indicates that the FPGA is ready to receive partial reconfiguration or external scrubbing data.
- PR_ERROR - Partial reconfiguration error indicator. A PR_ERROR value of 1 indicates that the FPGA detected an error during partial reconfiguration or external scrubbing.
- PR_DONE - Partial reconfiguration done indicator. A PR_DONE value of 1 indicates partial reconfiguration or external scrubbing is done.
- nCONFIG Pin - Value of the nCONFIG pin. This can be pulled-down by the FPGA in this device or logic external to this device connected to the nCONFIG pin. See the description of the nCONFIG field in this register to understand when the FPGA in this device pulls-down the nCONFIG pin. Logic external to this device pulls-down the nCONFIG pin to put the FPGA into the Reset Phase.

- nSTATUS Pin - Value of the nSTATUS pin. This can be pulled-down by the FPGA in this device or logic external to this device connected to the nSTATUS pin. See the description of the nSTATUS field in this register to understand when the FPGA in this device pulls-down the nSTATUS pin. Logic external to this device pulls-down the nSTATUS pin during Configuration Phase or Initialization Phase if it detected an error.
- CONF_DONE Pin - Value of the CONF_DONE pin. This can be pulled-down by the FPGA in this device or logic external to this device connected to the CONF_DONE pin. See the description of the CONF_DONE field in this register to understand when the FPGA in this device pulls-down the CONF_DONE pin. See FPGA documentation to determine how logic external to this device drives CONF_DONE.
- FPGA_POWER_ON - FPGA powered on indicator
  - 0 = FPGA portion of device is powered off.
  - 1 = FPGA portion of device is powered on.

Offset: `0x800`

gpio_inten on page 4-23
Allows each bit of Port A to be configured to generate an interrupt or not.

gpio_intmask on page 4-25
This register has 12 individual interrupt masks for the MON. Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. By default, all interrupts bits are unmasked. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. The unmasked status can be read as well as the resultant status after masking.

gpio_inttype_level on page 4-28
The interrupt level register defines the type of interrupt (edge or level) for each GPIO input.

gpio_int_polarity on page 4-31
Controls the polarity of interrupts that can occur on each GPIO input.

gpio_intstatus on page 4-34
Reports on interrupt status for each GPIO input. The interrupt status includes the effects of masking.

gpio_raw_intstatus on page 4-36
Reports on raw interrupt status for each GPIO input. The raw interrupt status excludes the effects of masking.

gpio_porta_eoi on page 4-39
This register is written by software to clear edge interrupts generated by each individual GPIO input. This register always reads back as zero.

gpio_ext_porta on page 4-41
Reading this register reads the values of the GPIO inputs.

gpio_ls_sync on page 4-42
The Synchronization level register is used to synchronize inputs to the l4_mp_clk. All MON interrupts are already synchronized before the GPIO instance so it is not necessary to setup this register to enable synchronization.

gpio_ver_id_code on page 4-43
GPIO Component Version

gpio_config_reg2 on page 4-44
Specifies the bit width of port A.

gpio_config_reg1 on page 4-45
Reports settings of various GPIO configuration parameters

gpio_inten

Allows each bit of Port A to be configured to generate an interrupt or not.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706830

Offset: 0x830

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RW 0x0	cdp RW 0x0	nsp RW 0x0	ncp RW 0x0	prd RW 0x0	pre RW 0x0	prr RW 0x0	ccd RW 0x0	crc RW 0x0	id RW 0x0	cd RW 0x0	ns RW 0x0

gpio_inten Fields

Bit	Name	Description	Access	Reset
11	fpo	Enables interrupt generation for FPGA_POWER_ON **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
10	cdp	Enables interrupt generation for CONF_DONE Pin **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
9	nsp	Enables interrupt generation for nSTATUS Pin **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
8	ncp	Enables interrupt generation for nCONFIG Pin **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
7	prd	Enables interrupt generation for PR_DONE **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
6	pre	Enables interrupt generation for PR_ERROR **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
5	prr	Enables interrupt generation for PR_READY **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
4	ccd	Enables interrupt generation for CVP_CONF_DONE **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0
3	crc	Enables interrupt generation for CRC_ERROR **Value** **Description** 0x0 Disable Interrupt 0x1 Enable Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
2	`id`	Enables interrupt generation for INIT_DONE **Value** — **Description** 0x0 — Disable Interrupt 0x1 — Enable Interrupt	`RW`	`0x0`
1	`cd`	Enables interrupt generation for CONF_DONE **Value** — **Description** 0x0 — Disable Interrupt 0x1 — Enable Interrupt	`RW`	`0x0`
0	`ns`	Enables interrupt generation for nSTATUS **Value** — **Description** 0x0 — Disable Interrupt 0x1 — Enable Interrupt	`RW`	`0x0`

gpio_intmask

This register has 12 individual interrupt masks for the MON. Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. By default, all interrupts bits are unmasked. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. The unmasked status can be read as well as the resultant status after masking.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF706834`

Offset: `0x834`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td>fpo
RW
0x0</td><td>cdp
RW
0x0</td><td>nsp
RW
0x0</td><td>ncp
RW
0x0</td><td>prd
RW
0x0</td><td>pre
RW
0x0</td><td>prr
RW
0x0</td><td>ccd
RW
0x0</td><td>crc
RW
0x0</td><td>id
RW
0x0</td><td>cd
RW
0x0</td><td>ns
RW 0x0</td></tr>
</table>

gpio_intmask Fields

Bit	Name	Description	Access	Reset
11	`fpo`	Controls whether an interrupt for FPGA_POWER_ ON can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	RW	0x0
10	`cdp`	Controls whether an interrupt for CONF_DONE Pin can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	RW	0x0
9	`nsp`	Controls whether an interrupt for nSTATUS Pin can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	RW	0x0
8	`ncp`	Controls whether an interrupt for nCONFIG Pin can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
7	`prd`	Controls whether an interrupt for PR_DONE can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** **Description** 0x0 Unmask Interrupt 0x1 Mask Interrupt	`RW`	`0x0`
6	`pre`	Controls whether an interrupt for PR_ERROR can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** **Description** 0x0 Unmask Interrupt 0x1 Mask Interrupt	`RW`	`0x0`
5	`prr`	Controls whether an interrupt for PR_READY can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** **Description** 0x0 Unmask Interrupt 0x1 Mask Interrupt	`RW`	`0x0`
4	`ccd`	Controls whether an interrupt for CVP_CONF_DONE can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** **Description** 0x0 Unmask Interrupt 0x1 Mask Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
3	`crc`	Controls whether an interrupt for CRC_ERROR can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	`RW`	`0x0`
2	`id`	Controls whether an interrupt for INIT_DONE can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	`RW`	`0x0`
1	`cd`	Controls whether an interrupt for CONF_DONE can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	`RW`	`0x0`
0	`ns`	Controls whether an interrupt for nSTATUS can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Unmask Interrupt 0x1 — Mask Interrupt	`RW`	`0x0`

gpio_inttype_level

The interrupt level register defines the type of interrupt (edge or level) for each GPIO input.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF706838`

Offset: `0x838`

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RW 0x0	cdp RW 0x0	nsp RW 0x0	ncp RW 0x0	prd RW 0x0	pre RW 0x0	prr RW 0x0	ccd RW 0x0	crc RW 0x0	id RW 0x0	cd RW 0x0	ns RW 0x0

gpio_inttype_level Fields

Bit	Name	Description	Access	Reset
11	fpo	Controls whether the level of FPGA_POWER_ON or an edge on FPGA_POWER_ON generates an interrupt. **Value** **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
10	cdp	Controls whether the level of CONF_DONE Pin or an edge on CONF_DONE Pin generates an interrupt. **Value** **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
9	nsp	Controls whether the level of nSTATUS Pin or an edge on nSTATUS Pin generates an interrupt. **Value** **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
8	ncp	Controls whether the level of nCONFIG Pin or an edge on nCONFIG Pin generates an interrupt. **Value** **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0

Bit	Name	Description	Access	Reset
7	prd	Controls whether the level of PR_DONE or an edge on PR_DONE generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
6	pre	Controls whether the level of PR_ERROR or an edge on PR_ERROR generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
5	prr	Controls whether the level of PR_READY or an edge on PR_READY generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
4	ccd	Controls whether the level of CVP_CONF_DONE or an edge on CVP_CONF_DONE generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
3	crc	Controls whether the level of CRC_ERROR or an edge on CRC_ERROR generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0

Bit	Name	Description	Access	Reset
2	id	Controls whether the level of INIT_DONE or an edge on INIT_DONE generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
1	cd	Controls whether the level of CONF_DONE or an edge on CONF_DONE generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0
0	ns	Controls whether the level of nSTATUS or an edge on nSTATUS generates an interrupt. **Value** / **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0

gpio_int_polarity

Controls the polarity of interrupts that can occur on each GPIO input.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF70683C

Offset: 0x83C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RW 0x0	cdp RW 0x0	nsp RW 0x0	ncp RW 0x0	prd RW 0x0	pre RW 0x0	prr RW 0x0	ccd RW 0x0	crc RW 0x0	id RW 0x0	cd RW 0x0	ns RW 0x0

gpio_int_polarity Fields

Bit	Name	Description	Access	Reset
11	`fpo`	Controls the polarity of edge or level sensitivity for FPGA_POWER_ON **Value** **Description** 0x0 Active low 0x1 Active high	RW	0x0
10	`cdp`	Controls the polarity of edge or level sensitivity for CONF_DONE Pin **Value** **Description** 0x0 Active low 0x1 Active high	RW	0x0
9	`nsp`	Controls the polarity of edge or level sensitivity for nSTATUS Pin **Value** **Description** 0x0 Active low 0x1 Active high	RW	0x0
8	`ncp`	Controls the polarity of edge or level sensitivity for nCONFIG Pin **Value** **Description** 0x0 Active low 0x1 Active high	RW	0x0
7	`prd`	Controls the polarity of edge or level sensitivity for PR_DONE **Value** **Description** 0x0 Active low 0x1 Active high	RW	0x0

Bit	Name	Description	Access	Reset
6	`pre`	Controls the polarity of edge or level sensitivity for PR_ERROR **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`
5	`prr`	Controls the polarity of edge or level sensitivity for PR_READY **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`
4	`ccd`	Controls the polarity of edge or level sensitivity for CVP_CONF_DONE **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`
3	`crc`	Controls the polarity of edge or level sensitivity for CRC_ERROR **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`
2	`id`	Controls the polarity of edge or level sensitivity for INIT_DONE **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`
1	`cd`	Controls the polarity of edge or level sensitivity for CONF_DONE **Value** **Description** 0x0 Active low 0x1 Active high	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	`ns`	Controls the polarity of edge or level sensitivity for nSTATUS **Value** — **Description** 0x0 — Active low 0x1 — Active high	`RW`	`0x0`

gpio_intstatus

Reports on interrupt status for each GPIO input. The interrupt status includes the effects of masking.

Module Instance	Base Address	Register Address
`fpgamgrregs`	`0xFF706000`	`0xFF706840`

Offset: `0x840`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RO 0x0	cdp RO 0x0	nsp RO 0x0	ncp RO 0x0	prd RO 0x0	pre RO 0x0	prr RO 0x0	ccd RO 0x0	crc RO 0x0	id RO 0x0	cd RO 0x0	ns RO 0x0

gpio_intstatus Fields

Bit	Name	Description	Access	Reset
11	`fpo`	Indicates whether FPGA_POWER_ON has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	`RO`	`0x0`
10	`cdp`	Indicates whether CONF_DONE Pin has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	`RO`	`0x0`

Bit	Name	Description	Access	Reset
9	nsp	Indicates whether nSTATUS Pin has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
8	ncp	Indicates whether nCONFIG Pin has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
7	prd	Indicates whether PR_DONE has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
6	pre	Indicates whether PR_ERROR has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
5	prr	Indicates whether PR_READY has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
4	ccd	Indicates whether CVP_CONF_DONE has an active interrupt or not (after masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0

Bit	Name	Description	Access	Reset
3	crc	Indicates whether CRC_ERROR has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
2	id	Indicates whether INIT_DONE has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
1	cd	Indicates whether CONF_DONE has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
0	ns	Indicates whether nSTATUS has an active interrupt or not (after masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0

gpio_raw_intstatus

Reports on raw interrupt status for each GPIO input. The raw interrupt status excludes the effects of masking.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706844

Offset: 0x844

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RO 0x0	cdp RO 0x0	nsp RO 0x0	ncp RO 0x0	prd RO 0x0	pre RO 0x0	prr RO 0x0	ccd RO 0x0	crc RO 0x0	id RO 0x0	cd RO 0x0	ns RO 0x0

gpio_raw_intstatus Fields

Bit	Name	Description	Access	Reset
11	fpo	Indicates whether FPGA_POWER_ON has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
10	cdp	Indicates whether CONF_DONE Pin has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
9	nsp	Indicates whether nSTATUS Pin has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
8	ncp	Indicates whether nCONFIG Pin has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0

Bit	Name	Description	Access	Reset
7	`prd`	Indicates whether PR_DONE has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
6	`pre`	Indicates whether PR_ERROR has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
5	`prr`	Indicates whether PR_READY has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
4	`ccd`	Indicates whether CVP_CONF_DONE has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
3	`crc`	Indicates whether CRC_ERROR has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0
2	`id`	Indicates whether INIT_DONE has an active interrupt or not (before masking). **Value** — **Description** 0x0 — Inactive 0x1 — Active	RO	0x0

Bit	Name	Description	Access	Reset
1	cd	Indicates whether CONF_DONE has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0
0	ns	Indicates whether nSTATUS has an active interrupt or not (before masking). **Value** **Description** 0x0 Inactive 0x1 Active	RO	0x0

gpio_porta_eoi

This register is written by software to clear edge interrupts generated by each individual GPIO input. This register always reads back as zero.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF70684C

Offset: 0x84C

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo WO 0x0	cdp WO 0x0	nsp WO 0x0	ncp WO 0x0	prd WO 0x0	pre WO 0x0	prr WO 0x0	ccd WO 0x0	crc WO 0x0	id WO 0x0	cd WO 0x0	ns WO 0x0

gpio_porta_eoi Fields

Bit	Name	Description	Access	Reset
11	fpo	Used by software to clear an FPGA_POWER_ON edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0

Bit	Name	Description	Access	Reset
10	`cdp`	Used by software to clear an CONF_DONE Pin edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
9	`nsp`	Used by software to clear an nSTATUS Pin edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
8	`ncp`	Used by software to clear an nCONFIG Pin edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
7	`prd`	Used by software to clear an PR_DONE edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
6	`pre`	Used by software to clear an PR_ERROR edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
5	`prr`	Used by software to clear an PR_READY edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0

Bit	Name	Description	Access	Reset
4	ccd	Used by software to clear an CVP_CONF_DONE edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
3	crc	Used by software to clear an CRC_ERROR edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
2	id	Used by software to clear an INIT_DONE edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
1	cd	Used by software to clear an CONF_DONE edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0
0	ns	Used by software to clear an nSTATUS edge interrupt. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0

gpio_ext_porta

Reading this register reads the values of the GPIO inputs.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706850

Offset: 0x850

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				fpo RO 0x0	cdp RO 0x0	nsp RO 0x0	ncp RO 0x0	prd RO 0x0	pre RO 0x0	prr RO 0x0	ccd RO 0x0	crc RO 0x0	id RO 0x0	cd RO 0x0	ns RO 0x0

gpio_ext_porta Fields

Bit	Name	Description	Access	Reset
11	fpo	Reading this provides the value of FPGA_POWER_ ON	RO	0x0
10	cdp	Reading this provides the value of CONF_DONE Pin	RO	0x0
9	nsp	Reading this provides the value of nSTATUS Pin	RO	0x0
8	ncp	Reading this provides the value of nCONFIG Pin	RO	0x0
7	prd	Reading this provides the value of PR_DONE	RO	0x0
6	pre	Reading this provides the value of PR_ERROR	RO	0x0
5	prr	Reading this provides the value of PR_READY	RO	0x0
4	ccd	Reading this provides the value of CVP_CONF_ DONE	RO	0x0
3	crc	Reading this provides the value of CRC_ERROR	RO	0x0
2	id	Reading this provides the value of INIT_DONE	RO	0x0
1	cd	Reading this provides the value of CONF_DONE	RO	0x0
0	ns	Reading this provides the value of nSTATUS	RO	0x0

gpio_ls_sync

The Synchronization level register is used to synchronize inputs to the l4_mp_clk. All MON interrupts are already synchronized before the GPIO instance so it is not necessary to setup this register to enable synchronization.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706860

Offset: 0x860

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															gpio_ls_sync RW 0x0

gpio_ls_sync Fields

Bit	Name	Description	Access	Reset
0	gpio_ls_sync	The level-sensitive interrupts is synchronized to l4_mp_clk. **Value** **Description** 0x0 No synchronization to l4_mp_clk 0x1 Synchronize to l4_mp_clk	RW	0x0

gpio_ver_id_code

GPIO Component Version

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF70686C

Offset: 0x86C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio_ver_id_code RO 0x3230382A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_ver_id_code RO 0x3230382A															

gpio_ver_id_code Fields

Bit	Name	Description	Access	Reset
31:0	gpio_ver_id_code	ASCII value for each number in the version, followed by *. For example. 32_30_31_2A represents the version 2.01	RO	0x32303 82A

gpio_config_reg2

Specifies the bit width of port A.

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706870

Offset: 0x870

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												encoded_id_pwidth_d RO 0x7			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
encoded_id_pwidth_d RO 0x7	encoded_id_pwidth_c RO 0x7					encoded_id_pwidth_b RO 0x7					encoded_id_pwidth_a RO 0xB				

gpio_config_reg2 Fields

Bit	Name	Description	Access	Reset
19:15	encoded_id_pwidth_d	Specifies the width of GPIO Port D. Ignored because there is no Port D in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0xb — Width (less 1) of 12 bits	RO	0x7
14:10	encoded_id_pwidth_c	Specifies the width of GPIO Port C. Ignored because there is no Port C in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0xb — Width (less 1) of 12 bits	RO	0x7
9:5	encoded_id_pwidth_b	Specifies the width of GPIO Port B. Ignored because there is no Port B in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0xb — Width (less 1) of 12 bits	RO	0x7

Bit	Name	Description	Access	Reset
4:0	encoded_id_pwidth_a	Specifies the width of GPIO Port A. The value 11 represents the 12-bit width less one. **Value** **Description** 0x7 Width (less 1) of 8 bits 0xb Width (less 1) of 12 bits	RO	0xB

gpio_config_reg1

Reports settings of various GPIO configuration parameters

Module Instance	Base Address	Register Address
fpgamgrregs	0xFF706000	0xFF706874

Offset: 0x874

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											encoded_id_width RO 0x1F				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_id RO 0x0	add_encoded_params RO 0x1	debounce RO 0x0	porta_intr RO 0x1	Reserved			hw_porta RO 0x0	portd_single_ctl RO 0x1	portc_single_ctl RO 0x1	portb_single_ctl RO 0x1	porta_single_ctl RO 0x1	num_ports RO 0x0		apb_data_width RO 0x2	

gpio_config_reg1 Fields

Bit	Name	Description	Access	Reset
20:16	encoded_id_width	This value is fixed at 32 bits. **Value** **Description** 0x1f Width of ID Field	RO	0x1F
15	gpio_id	Provides an ID code value **Value** **Description** 0x0 GPIO ID Code Register Excluded	RO	0x0

Bit	Name	Description	Access	Reset
14	`add_encoded_params`	Fixed to allow the indentification of the Designware IP component. **Value** — **Description** 0x1 — Enable IP indentification	RO	0x1
13	`debounce`	The value of this field is fixed to not allow debouncing of the Port A signals. **Value** — **Description** 0x0 — Debounce is Disabled	RO	0x0
12	`porta_intr`	The value of this field is fixed to allow interrupts on Port A. **Value** — **Description** 0x1 — Port A Interrupts Enabled	RO	0x1
8	`hw_porta`	The value is fixed to enable Port A configuration to be controlled by software only. **Value** — **Description** 0x0 — Software Configuration Control Enabled	RO	0x0
7	`portd_single_ctl`	Indicates the mode of operation of Port D to be software controlled only. Ignored because there is no Port D in the GPIO. **Value** — **Description** 0x1 — Software Enabled Individual Port Control	RO	0x1
6	`portc_single_ctl`	Indicates the mode of operation of Port C to be software controlled only. Ignored because there is no Port C in the GPIO. **Value** — **Description** 0x1 — Software Enabled Individual Port Control	RO	0x1

Bit	Name	Description	Access	Reset
5	`portb_single_ctl`	Indicates the mode of operation of Port B to be software controlled only. Ignored because there is no Port B in the GPIO. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1
4	`porta_single_ctl`	Indicates the mode of operation of Port A to be software controlled only. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1
3:2	`num_ports`	The value of this register is fixed at one port (Port A). **Value** **Description** 0x0 Number of GPIO Ports = 1	RO	0x0
1:0	`apb_data_width`	Fixed to support an APB data bus width of 32-bits. **Value** **Description** 0x2 APB Data Width = 32-bits	RO	0x2

Document Revision History

Table 4-2: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address maps and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor updates.
November 2012	1.3	Minor updates.
June 2012	1.2	Updated the FPGA configuration section.

Date	Version	Changes
May 2012	1.1	• Updated the configuration schemes table. • Updated the FPGA configuration section. • Added address map and register definitions section.
January 2012	1.0	Initial release.

System Manager 5

2014.06.30

cv_54014 Subscribe Send Feedback

The system manager in the hard processor system (HPS) contains memory-mapped control and status registers (CSRs) and logic to control system level functions as well as other modules in the HPS.

The system manager connects to the following modules in the HPS:

- Direct memory access (DMA) controller
- Ethernet media access controllers (EMAC0 and EMAC1)
- Microprocessor unit (MPU) subsystem
- NAND flash controller
- Secure Digital/MultiMediaCard (SD/MMC) controller
- Quad serial peripheral interface (SPI) flash controller
- USB 2.0 On-The-Go (OTG) controllers (USB0 and USB1)
- Watchdog timers

Features of the System Manager

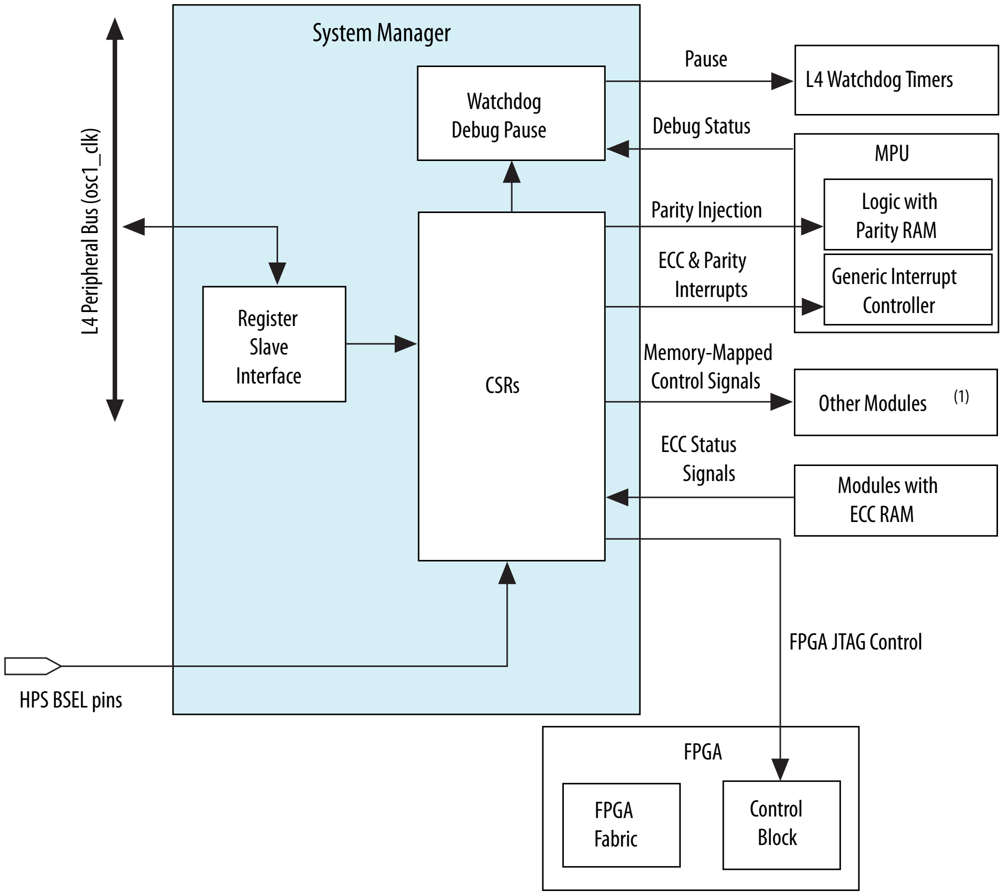
Software accesses the CSRs in the system manager to control and monitor various functions in other HPS modules that require external control signals. The system manager connects to these modules to perform the following functions:

- Sends pause signals to pause the watchdog timers when the processors in the MPU subsystem are in debug mode.
- Selects the EMAC level 3 (L3) master signal options .
- Freezes the I/O pins after the HPS comes out of cold reset and during serial configuration.
- Selects the SD/MMC controller clock options and L3 master signal options.
- Selects the NAND flash controller bootstrap options and L3 master signal options.
- Selects USB controller L3 master signal options.
- Provides control over the DMA security settings when the HPS exits from reset.
- Provides boot source and clock source information that can be read during the boot process.
- Provides the capability to enable/disable an interface of signals to the FPGA.
- Routes parity failure interrupts from the L1 caches to the Global Interrupt Controller.
- Sends error correction code (ECC) enable signals to all HPS modules with ECC protected RAM.
- Provides the capability to inject errors during testing in the MPU L2 ECC-protected RAM.

System Manager Block Diagram and System Integration

The system manager connects to the level 4 bus through a slave interface. The CSRs connect to signals in the FPGA and also to other HPS modules.

Figure 5-1: System Manager Block Diagram



The system manager consists of the following blocks:

- CSRs—Provide memory-mapped access to control signals and status for the following HPS modules:
  - EMACs
  - Debug core
  - SD/MMC controller
  - NAND controller
  - USB controllers
  - DMA controller
  - L3 Interconnect
  - ECC and parity interrupt routing to the MPU
  - Status information received from other HPS modules
- Register slave interface—provides connected masters access to the CSRs in the system manager.
- Watchdog debug pause—accepts the debug mode status from the MPU subsystem and pauses the L4 watchdog timers.

Functional Description of the System Manager

The system manager serves the following purposes:

- Provides software access to boot configuration and system information
- Provides software access to control and status signals in other HPS modules
- Provides combined ECC status and interrupt from other HPS modules with ECC-protected RAM
- Enables and disables HPS peripheral interfaces to the FPGA
- Provides eight registers that software can use to pass information between boot stages

Boot Configuration and System Information

The system manager provides boot configuration information through the `bootinfo` register. The following information is available to the Boot ROM software:

- Sampled value of the HPS boot select (`BSEL`) pins

The boot configuration information comes from the HPS `BSEL` pins. The `BSEL` signal informs Boot ROM software which flash device CPU0 booted from.

Related Information

Booting and Configuration

For boot and clock source values, refer to the *Booting and Configuration* appendix in the *Cyclone V Device Handbook, Volume 3*.

Additional Module Control

Each module in the HPS has its own CSRs, providing access to the internal state of the module. The system manager CSRs provide access to additional module state information, enabling additional control and monitoring. To fully control each module, you must manipulate both the module's own CSRs, and CSRs in the system manager. This section describes system manager CSR usage for each module.

DMA Controller

The security state of the DMA controller is controlled by the manager thread security `mgr_ns` and interrupt security (`irq_ns`) bits of the `dma` register.

Note: Register bits should be accessed by drivers or user application software only when the DMA master interface is guaranteed to be inactive.

Related Information

DMA Controller on page 16-1

NAND Flash Controller

The bootstrap control register (`nand_bootstrap`) modifies the default behavior of the NAND flash controller after reset. The NAND flash controller samples the register bits when it comes out of reset.

The following `bootstrap` register bits control configuration of the NAND flash controller:

- Bootstrap inhibit initialization bit (`noinit`)—inhibits the NAND flash controller from initializing when coming out of reset, and allows software to program all registers pertaining to device parameters such as page size and width.
- Bootstrap 512 byte device bit (`page512`)—informs the NAND flash controller that a NAND flash device of 512 byte page size is connected to the system.
- Bootstrap inhibit load block 0 page 0 bit (`noloadb0p0`)—inhibits the NAND flash controller from loading page 0 of block 0 of the NAND flash device as part of the initialization procedure.
- Bootstrap two row address cycles bit (`tworowaddr`)—informs the NAND flash controller that only two ROW address cycles are required instead of the default three row address cycles.

Registers in the system manager control the L3 master `ARCACHE` and `AWCACHE` signals. Set the NAND arcache (`arcache`) and NAND awcache (`awcache`) bits of the NAND L3 master AxCACHE register (`l3master`) to control these selections. These bits define the cache attributes for the master transactions of the DMA engine in the NAND controller.

Note: Register bits should be accessed only when the master interface is guaranteed to be in an inactive state.

Related Information

NAND Flash Controller on page 13-1

CAN Controller

The switching between the CAN controller and FPGA interfaces is controlled by the system manager. The DMA channels can be dedicated to the FPGA or to one of the four CAN interfaces.

Table 5-1: Peripheral Request Interface mapping

DMA Channel	Peripheral	CAN Controller
channel 0	FPGA 4	CAN0 interface 1
channel 1	FPGA 5	CAN0 interface 2
channel 2	FPGA 6	CAN1 interface 1
channel 3	FPGA 7	CAN1 interface 2

The ctrl register controls the mux that selects whether FPGA or CAN connects to one of the DMA peripherals request interfaces.

Table 5-2: Control Register (ctrl)

Interface	Value
FPGA	0x0
CAN	0x1

Related Information

- **Peripheral Request Interface** on page 16-9
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**
  For more information on the System Manger's registers, refer to the *Cyclone V SoC HPS Address Map and Register Definitions* page.

EMAC

The system manager allows software to select either emac_ptp_clk from the Clock Manager or f2s_ptp_ref_clk from the FPGA fabric as the source of the IEEE 1588 reference clock for each EMAC.

Registers in the system manager control the L3 master ARCACHE and AWCACHE signals. Set the EMAC arcache, awcache, and arprot, awprot bits to control there signals. These bits define the cache attributes for the master transactions of the DMA engine in the EMAC controllers.

The app_clk_sel bit determines the source of the clocks for the application clock. The ptp_ref_sel bit selects if the Timestamp reference is internally or externally generated. Note that EMAC0 must be set to Internal Timestamp. The phy_intf_sel bit determines if Out of Reset, GMII (or MII), RGMII or RMII would be used as the PHY interface.

Note: Register bits should be accessed only when the master interface is guaranteed to be in an inactive state.

Related Information

- **Clock Manager** on page 2-1
  For more information, refer to the Clock Manager chapter in the Cyclone V Device Handbook, Volume 3.
- **Ethernet Media Access Controller** on page 17-1

USB 2.0 OTG Controller

Registers in the system manager control the HPROT field of the USB master port of the USB 2.0 OTG Controller.

Note: Register bits should be accessed only when the master interface is guaranteed to be in an inactive state.

Related Information

USB 2.0 OTG Controller on page 18-1

SD/MMC Controller

Registers in the system manager control the HPROT field of the SD/MMC master port.

Note: Register bits should be accessed only when the master interface is guaranteed to be in an inactive state.

The system manager allows software to select the clock's phase shift for `cclk_in_drv` and `cclk_in_sample` by setting the drive clock phase shift select (`drvsel`) and sample clock phase shift select (`smplsel`) bits of the `sdmmc register`.

Related Information

SD/MMC Controller on page 14-1

Watchdog Timer

The system manager controls the watchdog timer behavior when the CPUs are in debug mode. The system manager sends a pause signal to the watchdog timers depending on the setting of the debug mode bits of the L4 watchdog debug register (`wddbg`). Each watchdog timer built into the MPU subsystem is automatically paused when its associated CPU enters debug mode.

Related Information

Watchdog Timer on page 24-1

Boot ROM Code

Registers in the system manager control whether the boot ROM code configures the pin multiplexing for boot pins after a warm reset. Set the warm-reset-configure-pin-multiplex for boot pins bit (`warmrstcfg-pinmux`) of the boot ROM code register to enable or disable this control.

Note: The boot ROM code always configures the pin multiplexing for boot pins after a cold reset.

Registers in the system manager also control whether the boot ROM code configures the I/O pins used during the boot process after a warm reset. Set the warm reset configure I/Os for boot pins bit (`warmrstcfgio`) of the `ctrl` register to enable or disable this control. By default, the boot ROM code always configures the I/O pins used by boot after a cold reset.

When CPU1 is released from reset and the boot ROM code is located at the CPU1 reset exception address (for a typical case), the boot ROM reset handler code reads the address stored in the CPU1 start address register (`cpu1startaddr`) and passes control to software at that address.

The preloader state register (`initswstate`) stores the magic number 0x49535756 written by the preloader to indicate there is a valid preloader software image in the on-chip RAM.

There can be up to four preloader images stored in flash memory. The (`initswlastld`) register contains the index of the preloader's last image that is loaded in the on-chip RAM.

The boot ROM software state register (`bootromswstate`) is a 32-bit general-purpose register reserved for the boot ROM.

The following `warmram` related registers are used to configure the warm reset from on-chip RAM feature.

Table 5-3: The warmram Registers

Register	Name	Purpose
`enable`	Enable	Controls whether the boot ROM attempts to boot from the contents of the on-chip RAM on a warm reset.

Register	Name	Purpose
`datastart`	Data start	Contains the byte offset of the warm boot CRC validation region in the on-chip RAM. The offset must be word-aligned to an integer multiple of four.
`length`	Length	Contains the length in bytes of the region in the on-chip RAM available for warm boot CRC validation.
`execution`	Execution offset	Contains the byte offset into the on-chip RAM that the boot code jumps to if the CRC validation succeeds.
`crc`	Expected CRC	Contains the expected CRC of the region in the on-chip RAM.

All the registers in the above table must be written by software prior to the warm reset occurring.

The number of wait states applied to the boot ROM's read operation is determined by the wait state bit (`waitstate`) of the `ctrl` register. After the boot process, software might require reading the code in the boot ROM. If software has changed the clock frequency of the `l3_main_clk` after reset, an additional wait state is necessary to access the boot ROM. Set the `waitstate` bit to add an additional wait state to the read access of the boot ROM. The enable safe mode warm reset update bit controls whether the wait state bit is updated during a warm reset.

L3 Interconnect

The System Manager provides remap bits to the L3 Interconnect. These bits can remap the Boot ROM and the On-chip RAM.

FPGA Interface Enables

The system manager can enable or disable interfaces between the FPGA and HPS. The interfaces must be disabled when not in use to avoid undefined behavior.

The global interface bit (`intf`) of the global disable register (`gbl`) disables all interfaces between the FPGA and HPS.

Note: Ensure that all interfaces between the FPGA and HPS are inactive before disabling them.

You can set the individual disable register (`indiv`) to disable the following interfaces between the FPGA and HPS:

- Reset request interface
- JTAG enable interface
- I/O configuration interface
- Boundary scan interface
- Debug interface
- Trace interface
- System Trace Macrocell (STM) interface
- Cross-trigger interface (CTI)
- NAND interface

- SD/MMC interface
- SPI Master interface
- EMAC interfaces

ECC and Parity Control

The system manager can enable or disable ECC for each of the following HPS modules with ECC-protected RAM:

- MPU L2 cache data RAM
- On-chip RAM
- USB 2.0 OTG controller (USB0 and USB1) RAM
- EMAC (EMAC0, EMAC1, and EMAC2) RAM
- DMA controller RAM
- CAN controller RAM
- NAND flash controller RAM
- Quad SPI flash controller RAM
- SD/MMC controller RAM
- DDR interfaces

The system manager can inject single-bit or double-bit errors into the MPU L2 ECC memories for testing purposes. Set the bits in the appropriate memory enable register to inject errors. For example, to inject a single bit EEC error, set the `injs` bit of the mpu_ctrl_l2_ecc register.

Note: The injection request is edge-sensitive, meaning that the request is latched on 0 to 1 transitions on the injection bit. The next time a write operation occurs, the data will be corrupted, containing either a single or double bit error as selected. When the data is read back, the ECC logic detects the single or double bit error appropriately. The injection request cannot be cancelled, and the number of injections is limited to once every 5 MPU cycles.

The system manager can also inject parity failures into the parity-protected RAM in the MPU L2 to test the parity failure interrupt handler. Set the bits of the parity fail injection register (`parityinj`) to inject parity failures.

Note: Injecting parity failures into the parity-protected RAM in the MPU L2 causes the interrupt to be raised immediately. There is no actual error injected and the data is not corrupted. Furthermore, there is no need for a memory operation to actually be performed for the interrupt to be raised.

Preloader Handoff Information

The system manager provides eight 32-bit registers to store handoff information between the preloader and the operating system. The preloader can store any information in these registers. These register contents have no impact on the state of the HPS hardware. When the operating system kernel boots, it retrieves the information by reading the preloader to OS handoff information register array. These registers are reset only by a cold reset.

Clocks

The system manager is driven by a clock generated by the clock manager.

Related Information

Clock Manager on page 2-1
For more information, refer to the Clock Manager chapter in the Cyclone V Device Handbook, Volume 3.

Resets

The system manager receives two reset signals from the reset manager. The `sys_manager_rst_n` signal is driven on a cold or warm reset and the `sys_manager_cold_rst_n` signal is driven only on a cold reset. This function allows the system manager to reset some CSR fields on either a cold or warm reset and others only on a cold reset.

Related Information

Reset Manager on page 3-1

System Manager Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- System Manager Module

Related Information

- **Introduction to the Hard Processor System**
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter in the *Cyclone V Device Handbook, Volume 3*.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

System Manager Module Address Map

Registers in the System Manager module

Base Address: `0xFFD08000`

System Manager Module

Register	Offset	Width	Access	Reset Value	Description
siliconid1 on page 5-22	0x0	32	RO	0x1	Silicon ID1 Register
siliconid2 on page 5-22	0x4	32	RO	0x0	Silicon ID2 Register
wddbg on page 5-23	0x10	32	RW	0xF	L4 Watchdog Debug Register
bootinfo on page 5-24	0x14	32	RO	0x0	Boot Info Register
hpsinfo on page 5-26	0x18	32	RO	0x0	HPS Info Register
parityinj on page 5-27	0x1C	32	RW	0x0	Parity Fail Injection Register

FPGA Interface Group

Register	Offset	Width	Acces s	Reset Value	Description
gbl on page 5-29	0x20	32	RW	0x1	Global Disable Register
indiv on page 5-30	0x24	32	RW	0xFF	Individual Disable Register
module on page 5-33	0x28	32	RW	0x0	Module Disable Register

Scan Manager Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-34	0x30	32	RW	0x0	Scan Manager Control Register

Freeze Control Group

Register	Offset	Width	Acces s	Reset Value	Description
vioctrl on page 5-36	0x40	32	RW	0x0	VIO Control Register
hioctrl on page 5-37	0x50	32	RW	0xE0	HIO Control Register
src on page 5-40	0x54	32	RW	0x0	Source Register
hwctrl on page 5-40	0x58	32	RW	0x5	Hardware Control Register

EMAC Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-42	0x60	32	RW	0xA	Control Register
l3master on page 5-43	0x64	32	RW	0x0	EMAC L3 Master AxCACHE Register

DMA Controller Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-48	0x70	32	RW	0x0	Control Register
persecurity on page 5-49	0x74	32	RW	0x0	Peripheral Security Register

Preloader (initial software) Group

Register	Offset	Width	Acces s	Reset Value	Description
handoff on page 5-50	0x80	32	RW	0x0	Preloader to OS Handoff Information

Boot ROM Code Register Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-51	0xC0	32	RW	0x0	Control Register
cpu1startaddr on page 5-53	0xC4	32	RW	0x0	CPU1 Start Address Register
initswstate on page 5-53	0xC8	32	RW	0x0	Preloader (initial software) State Register
initswlastld on page 5-54	0xCC	32	RW	0x0	Preloader (initial software) Last Image Loaded Register
bootromswstate on page 5-54	0xD0	32	RW	0x0	Boot ROM Software State Register

Warm Boot from On-Chip RAM Group

Register	Offset	Width	Acces s	Reset Value	Description
enable on page 5-55	0xE0	32	RW	0x0	Enable Register
datastart on page 5-56	0xE4	32	RW	0x0	Data Start Register
length on page 5-56	0xE8	32	RW	0x0	Length Register
execution on page 5-57	0xEC	32	RW	0x0	Execution Register
crc on page 5-58	0xF0	32	RW	0xE763552A	Expected CRC Register

Boot ROM Hardware Register Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-59	0x100	32	RW	0x2	Boot ROM Hardware Control Register

SDMMC Controller Group

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 5-60	0x108	32	RW	0x0	Control Register

Register	Offset	Width	Acces s	Reset Value	Description
l3master on page 5-61	0x10C	32	RW	0x3	SD/MMC L3 Master HPROT Register

NAND Flash Controller Register Group

Register	Offset	Width	Acces s	Reset Value	Description
bootstrap on page 5-63	0x110	32	RW	0x0	Bootstrap Control Register
l3master on page 5-63	0x114	32	RW	0x0	NAND L3 Master AxCACHE Register

USB Controller Group

Register	Offset	Width	Acces s	Reset Value	Description
l3master on page 5-65	0x118	32	RW	0xF	USB L3 Master HPROT Register

ECC Management Register Group

Register	Offset	Width	Acces s	Reset Value	Description
l2 on page 5-68	0x140	32	RW	0x0	L2 Data RAM ECC Enable Register
ocram on page 5-69	0x144	32	RW	0x0	On-chip RAM ECC Enable Register
usb0 on page 5-70	0x148	32	RW	0x0	USB0 RAM ECC Enable Register
usb1 on page 5-71	0x14C	32	RW	0x0	USB1 RAM ECC Enable Register
emac0 on page 5-72	0x150	32	RW	0x0	EMAC0 RAM ECC Enable Register
emac1 on page 5-73	0x154	32	RW	0x0	EMAC1 RAM ECC Enable Register
dma on page 5-75	0x158	32	RW	0x0	DMA RAM ECC Enable Register
can0 on page 5-76	0x15C	32	RW	0x0	CAN0 RAM ECC Enable Register
can1 on page 5-77	0x160	32	RW	0x0	CAN1 RAM ECC Enable Register
nand on page 5-78	0x164	32	RW	0x0	NAND RAM ECC Enable Register
qspi on page 5-80	0x168	32	RW	0x0	QSPI RAM ECC Enable Register
sdmmc on page 5-80	0x16C	32	RW	0x0	SDMMC RAM ECC Enable Register

Pin Mux Control Group

Register	Offset	Width	Acces s	Reset Value	Description
EMACIO0 on page 5-101	0x400	32	RW	0x0	emac0_tx_clk Mux Selection Register

Register	Offset	Width	Access	Reset Value	Description
EMACIO1 on page 5-102	0x404	32	RW	0x0	emac0_tx_d0 Mux Selection Register
EMACIO2 on page 5-103	0x408	32	RW	0x0	emac0_tx_d1 Mux Selection Register
EMACIO3 on page 5-103	0x40C	32	RW	0x0	emac0_tx_d2 Mux Selection Register
EMACIO4 on page 5-104	0x410	32	RW	0x0	emac0_tx_d3 Mux Selection Register
EMACIO5 on page 5-104	0x414	32	RW	0x0	emac0_rx_d0 Mux Selection Register
EMACIO6 on page 5-105	0x418	32	RW	0x0	emac0_mdio Mux Selection Register
EMACIO7 on page 5-105	0x41C	32	RW	0x0	emac0_mdc Mux Selection Register
EMACIO8 on page 5-106	0x420	32	RW	0x0	emac0_rx_ctl Mux Selection Register
EMACIO9 on page 5-107	0x424	32	RW	0x0	emac0_tx_ctl Mux Selection Register
EMACIO10 on page 5-107	0x428	32	RW	0x0	emac0_rx_clk Mux Selection Register
EMACIO11 on page 5-108	0x42C	32	RW	0x0	emac0_rx_d1 Mux Selection Register
EMACIO12 on page 5-109	0x430	32	RW	0x0	emac0_rx_d2 Mux Selection Register
EMACIO13 on page 5-109	0x434	32	RW	0x0	emac0_rx_d3 Mux Selection Register
EMACIO14 on page 5-110	0x438	32	RW	0x0	emac1_tx_clk Mux Selection Register
EMACIO15 on page 5-110	0x43C	32	RW	0x0	emac1_tx_d0 Mux Selection Register
EMACIO16 on page 5-111	0x440	32	RW	0x0	emac1_tx_d1 Mux Selection Register
EMACIO17 on page 5-112	0x444	32	RW	0x0	emac1_tx_ctl Mux Selection Register
EMACIO18 on page 5-112	0x448	32	RW	0x0	emac1_rx_d0 Mux Selection Register
EMACIO19 on page 5-113	0x44C	32	RW	0x0	emac1_rx_d1 Mux Selection Register
FLASHIO0 on page 5-113	0x450	32	RW	0x0	sdmmc_cmd Mux Selection Register
FLASHIO1 on page 5-114	0x454	32	RW	0x0	sdmmc_pwren Mux Selection Register
FLASHIO2 on page 5-115	0x458	32	RW	0x0	sdmmc_d0 Mux Selection Register
FLASHIO3 on page 5-115	0x45C	32	RW	0x0	sdmmc_d1 Mux Selection Register
FLASHIO4 on page 5-116	0x460	32	RW	0x0	sdmmc_d4 Mux Selection Register
FLASHIO5 on page 5-116	0x464	32	RW	0x0	sdmmc_d5 Mux Selection Register
FLASHIO6 on page 5-117	0x468	32	RW	0x0	sdmmc_d6 Mux Selection Register
FLASHIO7 on page 5-117	0x46C	32	RW	0x0	sdmmc_d7 Mux Selection Register
FLASHIO8 on page 5-118	0x470	32	RW	0x0	sdmmc_clk_in Mux Selection Register
FLASHIO9 on page 5-119	0x474	32	RW	0x0	sdmmc_clk Mux Selection Register
FLASHIO10 on page 5-119	0x478	32	RW	0x0	sdmmc_d2 Mux Selection Register
FLASHIO11 on page 5-120	0x47C	32	RW	0x0	sdmmc_d3 Mux Selection Register

Register	Offset	Width	Access	Reset Value	Description
GENERALIO0 on page 5-120	0x480	32	RW	0x0	trace_clk Mux Selection Register
GENERALIO1 on page 5-121	0x484	32	RW	0x0	trace_d0 Mux Selection Register
GENERALIO2 on page 5-122	0x488	32	RW	0x0	trace_d1 Mux Selection Register
GENERALIO3 on page 5-122	0x48C	32	RW	0x0	trace_d2 Mux Selection Register
GENERALIO4 on page 5-123	0x490	32	RW	0x0	trace_d3 Mux Selection Register
GENERALIO5 on page 5-123	0x494	32	RW	0x0	trace_d4 Mux Selection Register
GENERALIO6 on page 5-124	0x498	32	RW	0x0	trace_d5 Mux Selection Register
GENERALIO7 on page 5-125	0x49C	32	RW	0x0	trace_d6 Mux Selection Register
GENERALIO8 on page 5-125	0x4A0	32	RW	0x0	trace_d7 Mux Selection Register
GENERALIO9 on page 5-126	0x4A4	32	RW	0x0	spim0_clk Mux Selection Register
GENERALIO10 on page 5-126	0x4A8	32	RW	0x0	spim0_mosi Mux Selection Register
GENERALIO11 on page 5-127	0x4AC	32	RW	0x0	spim0_miso Mux Selection Register
GENERALIO12 on page 5-128	0x4B0	32	RW	0x0	spim0_ss0 Mux Selection Register
GENERALIO13 on page 5-128	0x4B4	32	RW	0x0	uart0_rx Mux Selection Register
GENERALIO14 on page 5-129	0x4B8	32	RW	0x0	uart0_tx Mux Selection Register
GENERALIO15 on page 5-129	0x4BC	32	RW	0x0	i2c0_sda Mux Selection Register
GENERALIO16 on page 5-130	0x4C0	32	RW	0x0	i2c0_scl Mux Selection Register
GENERALIO17 on page 5-131	0x4C4	32	RW	0x0	can0_rx Mux Selection Register
GENERALIO18 on page 5-131	0x4C8	32	RW	0x0	can0_tx Mux Selection Register

Register	Offset	Width	Acces s	Reset Value	Description
GENERALIO19 on page 5-132	0x4CC	32	RW	0x0	spis1_clk Mux Selection Register
GENERALIO20 on page 5-132	0x4D0	32	RW	0x0	spis1_mosi Mux Selection Register
GENERALIO21 on page 5-133	0x4D4	32	RW	0x0	spis1_miso Mux Selection Register
GENERALIO22 on page 5-134	0x4D8	32	RW	0x0	spis1_ss0 Mux Selection Register
GENERALIO23 on page 5-134	0x4DC	32	RW	0x0	uart1_rx Mux Selection Register
GENERALIO24 on page 5-135	0x4E0	32	RW	0x0	uart1_tx Mux Selection Register
GENERALIO25 on page 5-135	0x4E4	32	RW	0x0	i2c1_sda Mux Selection Register
GENERALIO26 on page 5-136	0x4E8	32	RW	0x0	i2c1_scl Mux Selection Register
GENERALIO27 on page 5-137	0x4EC	32	RW	0x0	spim0_ss0_alt Mux Selection Register
MIXED1IO0 on page 5-137	0x500	32	RW	0x0	nand_ale Mux Selection Register
MIXED1IO1 on page 5-138	0x504	32	RW	0x0	nand_ce Mux Selection Register
MIXED1IO2 on page 5-138	0x508	32	RW	0x0	nand_cle Mux Selection Register
MIXED1IO3 on page 5-139	0x50C	32	RW	0x0	nand_re Mux Selection Register
MIXED1IO4 on page 5-140	0x510	32	RW	0x0	nand_rb Mux Selection Register
MIXED1IO5 on page 5-140	0x514	32	RW	0x0	nand_dq0 Mux Selection Register
MIXED1IO6 on page 5-141	0x518	32	RW	0x0	nand_dq1 Mux Selection Register
MIXED1IO7 on page 5-141	0x51C	32	RW	0x0	nand_dq2 Mux Selection Register
MIXED1IO8 on page 5-142	0x520	32	RW	0x0	nand_dq3 Mux Selection Register
MIXED1IO9 on page 5-143	0x524	32	RW	0x0	nand_dq4 Mux Selection Register
MIXED1IO10 on page 5-143	0x528	32	RW	0x0	nand_dq5 Mux Selection Register
MIXED1IO11 on page 5-144	0x52C	32	RW	0x0	nand_dq6 Mux Selection Register
MIXED1IO12 on page 5-144	0x530	32	RW	0x0	nand_dq7 Mux Selection Register
MIXED1IO13 on page 5-145	0x534	32	RW	0x0	nand_wp Mux Selection Register

Register	Offset	Width	Access	Reset Value	Description
MIXED1IO14 on page 5-146	0x538	32	RW	0x0	nand_we Mux Selection Register
MIXED1IO15 on page 5-146	0x53C	32	RW	0x0	qspi_io0 Mux Selection Register
MIXED1IO16 on page 5-147	0x540	32	RW	0x0	qspi_io1 Mux Selection Register
MIXED1IO17 on page 5-147	0x544	32	RW	0x0	qspi_io2 Mux Selection Register
MIXED1IO18 on page 5-148	0x548	32	RW	0x0	qspi_io3 Mux Selection Register
MIXED1IO19 on page 5-149	0x54C	32	RW	0x0	qspi_ss0 Mux Selection Register
MIXED1IO20 on page 5-149	0x550	32	RW	0x0	qpsi_clk Mux Selection Register
MIXED1IO21 on page 5-150	0x554	32	RW	0x0	qspi_ss1 Mux Selection Register
MIXED2IO0 on page 5-150	0x558	32	RW	0x0	emac1_mdio Mux Selection Register
MIXED2IO1 on page 5-151	0x55C	32	RW	0x0	emac1_mdc Mux Selection Register
MIXED2IO2 on page 5-152	0x560	32	RW	0x0	emac1_tx_d2 Mux Selection Register
MIXED2IO3 on page 5-152	0x564	32	RW	0x0	emac1_tx_d3 Mux Selection Register
MIXED2IO4 on page 5-153	0x568	32	RW	0x0	emac1_rx_clk Mux Selection Register
MIXED2IO5 on page 5-153	0x56C	32	RW	0x0	emac1_rx_ctl Mux Selection Register
MIXED2IO6 on page 5-154	0x570	32	RW	0x0	emac1_rx_d2 Mux Selection Register
MIXED2IO7 on page 5-155	0x574	32	RW	0x0	emac1_rx_d3 Mux Selection Register
GPLINMUX48 on page 5-155	0x578	32	RW	0x0	GPIO/LoanIO 48 Input Mux Selection Register
GPLINMUX49 on page 5-156	0x57C	32	RW	0x0	GPIO/LoanIO 49 Input Mux Selection Register
GPLINMUX50 on page 5-156	0x580	32	RW	0x0	GPIO/LoanIO 50 Input Mux Selection Register
GPLINMUX51 on page 5-157	0x584	32	RW	0x0	GPIO/LoanIO 51 Input Mux Selection Register
GPLINMUX52 on page 5-157	0x588	32	RW	0x0	GPIO/LoanIO 52 Input Mux Selection Register
GPLINMUX53 on page 5-158	0x58C	32	RW	0x0	GPIO/LoanIO 53 Input Mux Selection Register
GPLINMUX54 on page 5-159	0x590	32	RW	0x0	GPIO/LoanIO 54 Input Mux Selection Register

Register	Offset	Width	Access	Reset Value	Description
GPLINMUX55 on page 5-159	0x594	32	RW	0x0	GPIO/LoanIO 55 Input Mux Selection Register
GPLINMUX56 on page 5-160	0x598	32	RW	0x0	GPIO/LoanIO 56 Input Mux Selection Register
GPLINMUX57 on page 5-160	0x59C	32	RW	0x0	GPIO/LoanIO 57 Input Mux Selection Register
GPLINMUX58 on page 5-161	0x5A0	32	RW	0x0	GPIO/LoanIO 58 Input Mux Selection Register
GPLINMUX59 on page 5-161	0x5A4	32	RW	0x0	GPIO/LoanIO 59 Input Mux Selection Register
GPLINMUX60 on page 5-162	0x5A8	32	RW	0x0	GPIO/LoanIO 60 Input Mux Selection Register
GPLINMUX61 on page 5-162	0x5AC	32	RW	0x0	GPIO/LoanIO 61 Input Mux Selection Register
GPLINMUX62 on page 5-163	0x5B0	32	RW	0x0	GPIO/LoanIO 62 Input Mux Selection Register
GPLINMUX63 on page 5-164	0x5B4	32	RW	0x0	GPIO/LoanIO 63 Input Mux Selection Register
GPLINMUX64 on page 5-164	0x5B8	32	RW	0x0	GPIO/LoanIO 64 Input Mux Selection Register
GPLINMUX65 on page 5-165	0x5BC	32	RW	0x0	GPIO/LoanIO 65 Input Mux Selection Register
GPLINMUX66 on page 5-165	0x5C0	32	RW	0x0	GPIO/LoanIO 66 Input Mux Selection Register
GPLINMUX67 on page 5-166	0x5C4	32	RW	0x0	GPIO/LoanIO 67 Input Mux Selection Register
GPLINMUX68 on page 5-166	0x5C8	32	RW	0x0	GPIO/LoanIO 68 Input Mux Selection Register
GPLINMUX69 on page 5-167	0x5CC	32	RW	0x0	GPIO/LoanIO 69 Input Mux Selection Register
GPLINMUX70 on page 5-167	0x5D0	32	RW	0x0	GPIO/LoanIO 70 Input Mux Selection Register
GPLMUX0 on page 5-168	0x5D4	32	RW	0x0	GPIO/LoanIO 0 Output/Output Enable Mux Selection Register
GPLMUX1 on page 5-169	0x5D8	32	RW	0x0	GPIO/LoanIO 1 Output/Output Enable Mux Selection Register
GPLMUX2 on page 5-169	0x5DC	32	RW	0x0	GPIO/LoanIO 2 Output/Output Enable Mux Selection Register

Register	Offset	Width	Acces s	Reset Value	Description
GPLMUX3 on page 5-170	0x5E0	32	RW	0x0	GPIO/LoanIO 3 Output/Output Enable Mux Selection Register
GPLMUX4 on page 5-170	0x5E4	32	RW	0x0	GPIO/LoanIO 4 Output/Output Enable Mux Selection Register
GPLMUX5 on page 5-171	0x5E8	32	RW	0x0	GPIO/LoanIO 5 Output/Output Enable Mux Selection Register
GPLMUX6 on page 5-171	0x5EC	32	RW	0x0	GPIO/LoanIO 6 Output/Output Enable Mux Selection Register
GPLMUX7 on page 5-172	0x5F0	32	RW	0x0	GPIO/LoanIO 7 Output/Output Enable Mux Selection Register
GPLMUX8 on page 5-173	0x5F4	32	RW	0x0	GPIO/LoanIO 8 Output/Output Enable Mux Selection Register
GPLMUX9 on page 5-173	0x5F8	32	RW	0x0	GPIO/LoanIO 9 Output/Output Enable Mux Selection Register
GPLMUX10 on page 5-174	0x5FC	32	RW	0x0	GPIO/LoanIO 10 Output/Output Enable Mux Selection Register
GPLMUX11 on page 5-174	0x600	32	RW	0x0	GPIO/LoanIO 11 Output/Output Enable Mux Selection Register
GPLMUX12 on page 5-175	0x604	32	RW	0x0	GPIO/LoanIO 12 Output/Output Enable Mux Selection Register
GPLMUX13 on page 5-176	0x608	32	RW	0x0	GPIO/LoanIO 13 Output/Output Enable Mux Selection Register
GPLMUX14 on page 5-176	0x60C	32	RW	0x0	GPIO/LoanIO 14 Output/Output Enable Mux Selection Register
GPLMUX15 on page 5-177	0x610	32	RW	0x0	GPIO/LoanIO 15 Output/Output Enable Mux Selection Register
GPLMUX16 on page 5-177	0x614	32	RW	0x0	GPIO/LoanIO 16 Output/Output Enable Mux Selection Register
GPLMUX17 on page 5-178	0x618	32	RW	0x0	GPIO/LoanIO 17 Output/Output Enable Mux Selection Register
GPLMUX18 on page 5-179	0x61C	32	RW	0x0	GPIO/LoanIO 18 Output/Output Enable Mux Selection Register
GPLMUX19 on page 5-179	0x620	32	RW	0x0	GPIO/LoanIO 19 Output/Output Enable Mux Selection Register
GPLMUX20 on page 5-180	0x624	32	RW	0x0	GPIO/LoanIO 20 Output/Output Enable Mux Selection Register
GPLMUX21 on page 5-180	0x628	32	RW	0x0	GPIO/LoanIO 21 Output/Output Enable Mux Selection Register

Register	Offset	Width	Acces s	Reset Value	Description
GPLMUX22 on page 5-181	0x62C	32	RW	0x0	GPIO/LoanIO 22 Output/Output Enable Mux Selection Register
GPLMUX23 on page 5-182	0x630	32	RW	0x0	GPIO/LoanIO 23 Output/Output Enable Mux Selection Register
GPLMUX24 on page 5-182	0x634	32	RW	0x0	GPIO/LoanIO 24 Output/Output Enable Mux Selection Register
GPLMUX25 on page 5-183	0x638	32	RW	0x0	GPIO/LoanIO 25 Output/Output Enable Mux Selection Register
GPLMUX26 on page 5-183	0x63C	32	RW	0x0	GPIO/LoanIO 26 Output/Output Enable Mux Selection Register
GPLMUX27 on page 5-184	0x640	32	RW	0x0	GPIO/LoanIO 27 Output/Output Enable Mux Selection Register
GPLMUX28 on page 5-185	0x644	32	RW	0x0	GPIO/LoanIO 28 Output/Output Enable Mux Selection Register
GPLMUX29 on page 5-185	0x648	32	RW	0x0	GPIO/LoanIO 29 Output/Output Enable Mux Selection Register
GPLMUX30 on page 5-186	0x64C	32	RW	0x0	GPIO/LoanIO 30 Output/Output Enable Mux Selection Register
GPLMUX31 on page 5-186	0x650	32	RW	0x0	GPIO/LoanIO 31 Output/Output Enable Mux Selection Register
GPLMUX32 on page 5-187	0x654	32	RW	0x0	GPIO/LoanIO 32 Output/Output Enable Mux Selection Register
GPLMUX33 on page 5-188	0x658	32	RW	0x0	GPIO/LoanIO 33 Output/Output Enable Mux Selection Register
GPLMUX34 on page 5-188	0x65C	32	RW	0x0	GPIO/LoanIO 34 Output/Output Enable Mux Selection Register
GPLMUX35 on page 5-189	0x660	32	RW	0x0	GPIO/LoanIO 35 Output/Output Enable Mux Selection Register
GPLMUX36 on page 5-189	0x664	32	RW	0x0	GPIO/LoanIO 36 Output/Output Enable Mux Selection Register
GPLMUX37 on page 5-190	0x668	32	RW	0x0	GPIO/LoanIO 37 Output/Output Enable Mux Selection Register
GPLMUX38 on page 5-191	0x66C	32	RW	0x0	GPIO/LoanIO 38 Output/Output Enable Mux Selection Register
GPLMUX39 on page 5-191	0x670	32	RW	0x0	GPIO/LoanIO 39 Output/Output Enable Mux Selection Register
GPLMUX40 on page 5-192	0x674	32	RW	0x0	GPIO/LoanIO 40 Output/Output Enable Mux Selection Register

Register	Offset	Width	Acces s	Reset Value	Description
GPLMUX41 on page 5-192	0x678	32	RW	0x0	GPIO/LoanIO 41 Output/Output Enable Mux Selection Register
GPLMUX42 on page 5-193	0x67C	32	RW	0x0	GPIO/LoanIO 42 Output/Output Enable Mux Selection Register
GPLMUX43 on page 5-194	0x680	32	RW	0x0	GPIO/LoanIO 43 Output/Output Enable Mux Selection Register
GPLMUX44 on page 5-194	0x684	32	RW	0x0	GPIO/LoanIO 44 Output/Output Enable Mux Selection Register
GPLMUX45 on page 5-195	0x688	32	RW	0x0	GPIO/LoanIO 45 Output/Output Enable Mux Selection Register
GPLMUX46 on page 5-195	0x68C	32	RW	0x0	GPIO/LoanIO 46 Output/Output Enable Mux Selection Register
GPLMUX47 on page 5-196	0x690	32	RW	0x0	GPIO/LoanIO 47 Output/Output Enable Mux Selection Register
GPLMUX48 on page 5-197	0x694	32	RW	0x0	GPIO/LoanIO 48 Output/Output Enable Mux Selection Register
GPLMUX49 on page 5-197	0x698	32	RW	0x0	GPIO/LoanIO 49 Output/Output Enable Mux Selection Register
GPLMUX50 on page 5-198	0x69C	32	RW	0x0	GPIO/LoanIO 50 Output/Output Enable Mux Selection Register
GPLMUX51 on page 5-198	0x6A0	32	RW	0x0	GPIO/LoanIO 51 Output/Output Enable Mux Selection Register
GPLMUX52 on page 5-199	0x6A4	32	RW	0x0	GPIO/LoanIO 52 Output/Output Enable Mux Selection Register
GPLMUX53 on page 5-200	0x6A8	32	RW	0x0	GPIO/LoanIO 53 Output/Output Enable Mux Selection Register
GPLMUX54 on page 5-200	0x6AC	32	RW	0x0	GPIO/LoanIO 54 Output/Output Enable Mux Selection Register
GPLMUX55 on page 5-201	0x6B0	32	RW	0x0	GPIO/LoanIO 55 Output/Output Enable Mux Selection Register
GPLMUX56 on page 5-201	0x6B4	32	RW	0x0	GPIO/LoanIO 56 Output/Output Enable Mux Selection Register
GPLMUX57 on page 5-202	0x6B8	32	RW	0x0	GPIO/LoanIO 57 Output/Output Enable Mux Selection Register
GPLMUX58 on page 5-203	0x6BC	32	RW	0x0	GPIO/LoanIO 58 Output/Output Enable Mux Selection Register
GPLMUX59 on page 5-203	0x6C0	32	RW	0x0	GPIO/LoanIO 59 Output/Output Enable Mux Selection Register

Register	Offset	Width	Acces s	Reset Value	Description
GPLMUX60 on page 5-204	0x6C4	32	RW	0x0	GPIO/LoanIO 60 Output/Output Enable Mux Selection Register
GPLMUX61 on page 5-204	0x6C8	32	RW	0x0	GPIO/LoanIO 61 Output/Output Enable Mux Selection Register
GPLMUX62 on page 5-205	0x6CC	32	RW	0x0	GPIO/LoanIO 62 Output/Output Enable Mux Selection Register
GPLMUX63 on page 5-206	0x6D0	32	RW	0x0	GPIO/LoanIO 63 Output/Output Enable Mux Selection Register
GPLMUX64 on page 5-206	0x6D4	32	RW	0x0	GPIO/LoanIO 64 Output/Output Enable Mux Selection Register
GPLMUX65 on page 5-207	0x6D8	32	RW	0x0	GPIO/LoanIO 65 Output/Output Enable Mux Selection Register
GPLMUX66 on page 5-207	0x6DC	32	RW	0x0	GPIO/LoanIO 66 Output/Output Enable Mux Selection Register
GPLMUX67 on page 5-208	0x6E0	32	RW	0x0	GPIO/LoanIO 67 Output/Output Enable Mux Selection Register
GPLMUX68 on page 5-209	0x6E4	32	RW	0x0	GPIO/LoanIO 68 Output/Output Enable Mux Selection Register
GPLMUX69 on page 5-209	0x6E8	32	RW	0x0	GPIO/LoanIO 69 Output/Output Enable Mux Selection Register
GPLMUX70 on page 5-210	0x6EC	32	RW	0x0	GPIO/LoanIO 70 Output/Output Enable Mux Selection Register
NANDUSEFPGA on page 5-210	0x6F0	32	RW	0x0	Select source for NAND signals (HPS Pins or FPGA Interface)
RGMII1USEFPGA on page 5-211	0x6F8	32	RW	0x0	Select source for RGMII1 signals (HPS Pins or FPGA Interface)
I2C0USEFPGA on page 5-212	0x704	32	RW	0x0	Select source for I2C0 signals (HPS Pins or FPGA Interface)
RGMII0USEFPGA on page 5-212	0x714	32	RW	0x0	Select source for RGMII0 signals (HPS Pins or FPGA Interface)
I2C3USEFPGA on page 5-213	0x724	32	RW	0x0	Select source for I2C3 signals (HPS Pins or FPGA Interface)
I2C2USEFPGA on page 5-213	0x728	32	RW	0x0	Select source for I2C2 signals (HPS Pins or FPGA Interface)
I2C1USEFPGA on page 5-214	0x72C	32	RW	0x0	Select source for I2C1 signals (HPS Pins or FPGA Interface)
SPIM1USEFPGA on page 5-214	0x730	32	RW	0x0	Select source for SPIM1 signals (HPS Pins or FPGA Interface)

Register	Offset	Width	Acces s	Reset Value	Description
SPIM0USEFPGA on page 5-215	0x738	32	RW	0x0	Select source for SPIM0 signals (HPS Pins or FPGA Interface)

siliconid1

Specifies Silicon ID and revision number.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08000

Offset: 0x0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
id RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rev RO 0x1															

siliconid1 Fields

Bit	Name	Description	Access	Reset
31:16	id	Silicon ID **Value** **Description** 0x0 HPS in Cyclone V and Arria V SoC FPGA devices	RO	0x0
15:0	rev	Silicon revision number. **Value** **Description** 0x1 Revision 1	RO	0x1

siliconid2

Reserved for future use.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08004

Offset: 0x4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rsv RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rsv RO 0x0															

siliconid2 Fields

Bit	Name	Description	Access	Reset
31:0	rsv	Reserved for future use.	RO	0x0

wddbg

Controls the behavior of the L4 watchdogs when the CPUs are in debug mode. These control registers are used to drive the pause input signal of the L4 watchdogs. Note that the watchdogs built into the MPU automatically are paused when their associated CPU enters debug mode. Only reset by a cold reset.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												mode_1 RW 0x3		mode_0 RW 0x3	

wddbg Fields

Bit	Name	Description	Access	Reset
3:2	`mode_1`	Controls behavior of L4 watchdog when CPUs in debug mode. Field array index matches L4 watchdog index. **Value** **Description** 0x0 Continue normal operation ignoring debug mode of CPUs 0x1 Pause normal operation only if CPU0 is in debug mode 0x2 Pause normal operation only if CPU1 is in debug mode 0x3 Pause normal operation if CPU0 or CPU1 is in debug mode	`RW`	`0x3`
1:0	`mode_0`	Controls behavior of L4 watchdog when CPUs in debug mode. Field array index matches L4 watchdog index. **Value** **Description** 0x0 Continue normal operation ignoring debug mode of CPUs 0x1 Pause normal operation only if CPU0 is in debug mode 0x2 Pause normal operation only if CPU1 is in debug mode 0x3 Pause normal operation if CPU0 or CPU1 is in debug mode	`RW`	`0x3`

bootinfo

Provides access to boot configuration information.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08014`

Offset: `0x14`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						pincsel RO 0x0		pinbsel RO 0x0			csel RO 0x0		bsel RO 0x0		

bootinfo Fields

Bit	Name	Description	Access	Reset
9:8	pincsel	Specifies the sampled value of the HPS CSEL pins. The value of HPS CSEL pins are sampled upon deassertion of cold reset.	RO	0x0
7:5	pinbsel	Specifies the sampled value of the HPS BSEL pins. The value of HPS BSEL pins are sampled upon deassertion of cold reset.	RO	0x0
4:3	csel	The clock select field specifies clock information for booting. The clock select encoding is a function of the CSEL value. The clock select field is read by the Boot ROM code on a cold or warm reset when booting from a flash device to get information about how to setup the HPS clocking to boot from the specified clock device. The encoding of the clock select field is specified by the enum associated with this field. The HPS CSEL pins value are sampled upon deassertion of cold reset. **Value** — **Description** 0x0 — QSPI device clock is osc1_clk divided by 4, SD/MMC device clock is osc1_clk divided by 4, NAND device operation is osc1_clk divided by 25 0x1 — QSPI device clock is osc1_clk divided by 2, SD/MMC device clock is osc1_clk divided by 1, NAND device operation is osc1_clk multiplied by 20/25 0x2 — QSPI device clock is osc1_clk divided by 1, SD/MMC device clock is osc1_clk divided by 2, NAND device operation is osc1_clk multiplied by 10/25 0x3 — QSPI device clock is osc1_clk multiplied by 2, SD/MMC device clock is osc1_clk divided by 4, NAND device operation is osc1_clk multiplied by 5/25	RO	0x0

Bit	Name	Description	Access	Reset
2:0	`bsel`	The boot select field specifies the boot source. It is read by the Boot ROM code on a cold or warm reset to determine the boot source. The HPS BSEL pins value are sampled upon deassertion of cold reset. **Value** **Description** 0x0 Reserved 0x1 FPGA (HPS2FPGA Bridge) 0x2 NAND Flash (1.8v) 0x3 NAND Flash (3.0v) 0x4 SD/MMC External Transceiver (1.8v) 0x5 SD/MMC Internal Transceiver (3.0v) 0x6 QSPI Flash (1.8v) 0x7 QSPI Flash (3.0v)	RO	0x0

hpsinfo

Provides information about the HPS capabilities.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08018`

Offset: `0x18`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														can RO 0x0	dualcore RO 0x0

hpsinfo Fields

Bit	Name	Description	Access	Reset
1	`can`	Indicates if CAN0 and CAN1 controllers are available or not. **Value** **Description** 0x0 CAN0 and CAN1 are not available. 0x1 CAN0 and CAN1 are available.	RO	0x0
0	`dualcore`	Indicates if CPU1 is available in MPU or not. **Value** **Description** 0x0 Not dual-core (only CPU0 available). 0x1 Is dual-core (CPU0 and CPU1 both available).	RO	0x0

parityinj

Inject parity failures into the parity-protected RAMs in the MPU. Allows software to test the parity failure interrupt handler. The field array index corresponds to the CPU index. All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0801C`

Offset: `0x1C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
btac_1 RW 0x0	btac_0 RW 0x0	ghb_1 RW 0x0	ghb_0 RW 0x0	ictag_1 RW 0x0	ictag_0 RW 0x0	icdata_1 RW 0x0	icdata_0 RW 0x0	maintlb_1 RW 0x0	maintlb_0 RW 0x0	dcouter_1 RW 0x0	dcouter_0 RW 0x0	dctag_1 RW 0x0	dctag_0 RW 0x0	dcdata_1 RW 0x0	dcdata_0 RW 0x0

parityinj Fields

Bit	Name	Description	Access	Reset
15	`btac_1`	If 1, injecting parity error to BTAC RAM.The field array index corresponds to the CPU index.	RW	0x0

Bit	Name	Description	Access	Reset
14	btac_0	If 1, injecting parity error to BTAC RAM.The field array index corresponds to the CPU index.	RW	0x0
13	ghb_1	If 1, injecting parity error to GHB RAM.The field array index corresponds to the CPU index.	RW	0x0
12	ghb_0	If 1, injecting parity error to GHB RAM.The field array index corresponds to the CPU index.	RW	0x0
11	ictag_1	If 1, injecting parity error to Instruction Cache Tag RAM.The field array index corresponds to the CPU index.	RW	0x0
10	ictag_0	If 1, injecting parity error to Instruction Cache Tag RAM.The field array index corresponds to the CPU index.	RW	0x0
9	icdata_1	If 1, injecting parity error to Instruction Cache Data RAM.The field array index corresponds to the CPU index.	RW	0x0
8	icdata_0	If 1, injecting parity error to Instruction Cache Data RAM.The field array index corresponds to the CPU index.	RW	0x0
7	maintlb_1	If 1, injecting parity error to Main TLB RAM.The field array index corresponds to the CPU index.	RW	0x0
6	maintlb_0	If 1, injecting parity error to Main TLB RAM.The field array index corresponds to the CPU index.	RW	0x0
5	dcouter_1	If 1, injecting parity error to Data Cache Outer RAM.The field array index corresponds to the CPU index.	RW	0x0
4	dcouter_0	If 1, injecting parity error to Data Cache Outer RAM.The field array index corresponds to the CPU index.	RW	0x0
3	dctag_1	If 1, injecting parity error to Data Cache Tag RAM.The field array index corresponds to the CPU index.	RW	0x0
2	dctag_0	If 1, injecting parity error to Data Cache Tag RAM.The field array index corresponds to the CPU index.	RW	0x0

Bit	Name	Description	Access	Reset
1	dcdata_1	If 1, injecting parity error to Data Cache Data RAM.The field array index corresponds to the CPU index.	RW	0x0
0	dcdata_0	If 1, injecting parity error to Data Cache Data RAM.The field array index corresponds to the CPU index.	RW	0x0

FPGA Interface Group Register Descriptions

Registers used to enable/disable interfaces between the FPGA and HPS. Required for either of the following situations:[list][*]Interfaces that cannot be disabled by putting an HPS module associated with the interface into reset.[*]HPS modules that accept signals from the FPGA fabric and those signals might interfere with the normal operation of the module.[/list]. All registers are only reset by a cold reset (ignore warm reset).

Offset: 0x20

gbl on page 5-29
Used to disable all interfaces between the FPGA and HPS.

indiv on page 5-30
Used to disable individual interfaces between the FPGA and HPS.

module on page 5-33
Used to disable signals from the FPGA fabric to individual HPS modules.

gbl

Used to disable all interfaces between the FPGA and HPS.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															intf RW 0x1

gbl Fields

Bit	Name	Description	Access	Reset
0	intf	Used to disable all interfaces between the FPGA and HPS. Software must ensure that all interfaces between the FPGA and HPS are inactive before disabling them. **Value** — **Description** 0x0 — All interfaces between FPGA and HPS are disabled. 0x1 — Interfaces between FPGA and HPS are not all disabled. Interfaces can be indivdually disabled by putting the HPS module associated with the interface in reset using registers in the Reset Manager or by using registers in this register group of the System Manager for interfaces without an associated module.	RW	0x1

indiv

Used to disable individual interfaces between the FPGA and HPS.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08024

Offset: 0x24

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td>crosstrigintf
RW 0x1</td><td>stmeventintf
RW 0x1</td><td>Reserved</td><td>traceintf
RW 0x1</td><td>bscanintf
RW 0x1</td><td>configiointf
RW 0x1</td><td>jtagenintf
RW 0x1</td><td>rstreqintf
RW 0x1</td></tr>
</table>

indiv Fields

Bit	Name	Description	Access	Reset
7	`crosstrigintf`	Used to disable the FPGA Fabric from sending triggers to HPS debug logic. Note that this doesn't prevent the HPS debug logic from sending triggers to the FPGA Fabric. **Value** **Description** 0x0 FPGA Fabric cannot send triggers. 0x1 FPGA Fabric can send triggers.	`RW`	`0x1`
6	`stmeventintf`	Used to disable the STM event interface. This interface allows logic in the FPGA fabric to trigger events to the STM debug module in the HPS. **Value** **Description** 0x0 STM event interface is disabled. Logic in the FPGA fabric cannot trigger STM events. 0x1 STM event interface is enabled. Logic in the FPGA fabric can trigger STM events.	`RW`	`0x1`
4	`traceintf`	Used to disable the trace interface. This interface allows the HPS debug logic to send trace data to logic in the FPGA fabric. **Value** **Description** 0x0 Trace interface is disabled. HPS debug logic cannot send trace data to the FPGA fabric. 0x1 Trace interface is enabled. Other registers in the HPS debug logic must be programmmed to actually send trace data to the FPGA fabric.	`RW`	`0x1`

Bit	Name	Description	Access	Reset
3	`bscanintf`	Used to disable the boundary-scan interface. This interface allows the FPGA JTAG TAP controller to execute boundary-scan instructions such as SAMPLE/PRELOAD, EXTEST, and HIGHZ. The boundary-scan interface must be enabled before attempting to send the boundary-scan instructions to the FPGA JTAG TAP controller. **Value** — **Description** 0x0 — Boundary-scan interface is disabled. Execution of boundary-scan instructions in the FPGA JTAG TAP controller is unsupported and produces undefined results. 0x1 — Boundary-scan interface is enabled. Execution of the boundary-scan instructions in the FPGA JTAG TAP controller is supported.	RW	0x1
2	`configiointf`	Used to disable the CONFIG_IO interface. This interface allows the FPGA JTAG TAP controller to execute the CONFIG_IO instruction and configure all device I/Os (FPGA and HPS). This is typically done before executing boundary-scan instructions. The CONFIG_IO interface must be enabled before attempting to send the CONFIG_IO instruction to the FPGA JTAG TAP controller. **Value** — **Description** 0x0 — CONFIG_IO interface is disabled. Execution of the CONFIG_IO instruction in the FPGA JTAG TAP controller is unsupported and produces undefined results. 0x1 — CONFIG_IO interface is enabled. Execution of the CONFIG_IO instruction in the FPGA JTAG TAP controller is supported.	RW	0x1
1	`jtagenintf`	Used to disable the JTAG enable interface. This interface allows logic in the FPGA fabric to disable the HPS JTAG operation. **Value** — **Description** 0x0 — JTAG enable interface is disabled. Logic in the FPGA fabric cannot disable the HPS JTAG. 0x1 — JTAG enable interface is enabled. Logic in the FPGA fabric can disable the HPS JTAG.	RW	0x1

Bit	Name	Description	Access	Reset
0	`rstreqintf`	Used to disable the reset request interface. This interface allows logic in the FPGA fabric to request HPS resets. This field disables the following reset request signals from the FPGA fabric to HPS:[list][*]f2h_cold_rst_req_n - Triggers a cold reset of the HPS[*]f2h_warm_rst_req_n - Triggers a warm reset of the HPS[*]f2h_dbg_rst_req_n - Triggers a debug reset of the HPS[/list] **Value** **Description** 0x0 Reset request interface is disabled. Logic in the FPGA fabric cannot reset the HPS. 0x1 Reset request interface is enabled. Logic in the FPGA fabric can reset the HPS.	RW	0x1

module

Used to disable signals from the FPGA fabric to individual HPS modules.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08028`

Offset: `0x28`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												emac_1 RW 0x0	emac_0 RW 0x0	Reserved	

module Fields

Bit	Name	Description	Access	Reset
3	emac_1	Used to disable signals from the FPGA fabric to the EMAC modules that could potentially interfere with their normal operation. The array index corresponds to the EMAC module instance. **Value** **Description** 0x0 Signals from FPGA fabric cannot affect operation of the EMAC module. 0x1 Signals from FPGA fabric can potentially affect operation of the EMAC module.	RW	0x0
2	emac_0	Used to disable signals from the FPGA fabric to the EMAC modules that could potentially interfere with their normal operation. The array index corresponds to the EMAC module instance. **Value** **Description** 0x0 Signals from FPGA fabric cannot affect operation of the EMAC module. 0x1 Signals from FPGA fabric can potentially affect operation of the EMAC module.	RW	0x0

Scan Manager Group Register Descriptions

Registers related to the Scan Manager that aren't located inside the Scan Manager itself.

Offset: 0x30

ctrl on page 5-34
Controls behaviors of Scan Manager not controlled by registers in the Scan Manager itself.

ctrl

Controls behaviors of Scan Manager not controlled by registers in the Scan Manager itself.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08030

Offset: 0x30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															fpgajtagen RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
0	fpgajtagen	Controls whether FPGA JTAG pins or Scan Manager drives JTAG signals to the FPGA. Only reset by a cold reset (ignores warm reset). **Value** **Description** 0x0 FPGA JTAG pins drive JTAG signals to FPGA 0x1 Scan Manager drives JTAG signals to FPGA	RW	0x0

Freeze Control Group Register Descriptions

Registers used to generate HPS IO freeze signals. All registers are only reset by a cold reset (ignore warm reset).

Offset: 0x40

vioctrl on page 5-36
Used to drive freeze signals to HPS VIO banks. The register array index corresponds to the freeze channel. Freeze channel 0 provides freeze signals to VIO bank 0 and 1. Freeze channel 1 provides freeze signals to VIO bank 2 and 3. Only drives freeze signals when SRC.VIO1 is set to SW. Freeze channel 2 provides freeze signals to VIO bank 4. All fields are only reset by a cold reset (ignore warm reset). The following equation determines when the weak pullup resistor is enabled: enabled = ~wkpullup | (CFF & cfg & tristate) where CFF is the value of weak pullup as set by IO configuration

hioctrl on page 5-37
Used to drive freeze signals to HPS HIO bank (DDR SDRAM). All fields are only reset by a cold reset (ignore warm reset). The following equation determines when the weak pullup resistor is enabled: enabled = ~wkpullup | (CFF & cfg & tristate) where CFF is the value of weak pullup as set by IO configuration

src on page 5-40
Contains register field to choose between software state machine (vioctrl array index [1] register) or hardware state machine in the Freeze Controller as the freeze signal source for VIO channel 1. All fields are only reset by a cold reset (ignore warm reset).

hwctrl on page 5-40

Activate freeze or thaw operations on VIO channel 1 (HPS IO bank 2 and bank 3) and monitor for completeness and the current state. These fields interact with the hardware state machine in the Freeze Controller. These fields can be accessed independent of the value of SRC1.VIO1 although they only have an effect on the VIO channel 1 freeze signals when SRC1.VIO1 is setup to have the hardware state machine be the freeze signal source. All fields are only reset by a cold reset (ignore warm reset).

vioctrl

Used to drive freeze signals to HPS VIO banks. The register array index corresponds to the freeze channel. Freeze channel 0 provides freeze signals to VIO bank 0 and 1. Freeze channel 1 provides freeze signals to VIO bank 2 and 3. Only drives freeze signals when SRC.VIO1 is set to SW. Freeze channel 2 provides freeze signals to VIO bank 4. All fields are only reset by a cold reset (ignore warm reset). The following equation determines when the weak pullup resistor is enabled: enabled = ~wkpullup | (CFF & cfg & tristate) where CFF is the value of weak pullup as set by IO configuration

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08040`

Offset: `0x40`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											slew RW 0x0	wkpullup RW 0x0	tristate RW 0x0	bushold RW 0x0	cfg RW 0x0

vioctrl Fields

Bit	Name	Description	Access	Reset
4	`slew`	Controls IO slew-rate **Value** **Description** 0x0 Slew-rate forced to slow. 0x1 Slew-rate controlled by IO configuration.	RW	0x0

Bit	Name	Description	Access	Reset
3	wkpullup	Controls weak pullup resistor **Value** **Description** 0x0 Weak pullup resistor enabled. 0x1 Weak pullup resistor enable controlled by IO configuration.	RW	0x0
2	tristate	Controls IO tri-state **Value** **Description** 0x0 IO tri-state enabled. 0x1 IO tri-state controlled by IO configuration.	RW	0x0
1	bushold	Controls bus hold circuit **Value** **Description** 0x0 Disable bus hold circuit. 0x1 Bus hold circuit controlled by IO configuration.	RW	0x0
0	cfg	Controls IO configuration **Value** **Description** 0x0 Disable IO configuration (forced to a safe value). 0x1 Enables IO configuration as previously configured by software using the Scan Manager.	RW	0x0

hioctrl

Used to drive freeze signals to HPS HIO bank (DDR SDRAM). All fields are only reset by a cold reset (ignore warm reset). The following equation determines when the weak pullup resistor is enabled: enabled = ~wkpullup | (CFF & cfg & tristate) where CFF is the value of weak pullup as set by IO configuration

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							oct_cfgen_calstart RW 0x0	regrst RW 0x1	octrst RW 0x1	dllrst RW 0x1	slew RW 0x0	wkpullup RW 0x0	tristate RW 0x0	bushold RW 0x0	cfg RW 0x0

hioctrl Fields

Bit	Name	Description	Access	Reset
8	oct_cfgen_calstart	Controls OCT calibration and OCT IO configuration enable. **Value** **Description** 0x0 Disables IO configuration (forced to a safe value) in OCT calibration block. 0x1 Starts OCT calibration state machine and enables IO configuration in OCT calibration block.	RW	0x0
7	regrst	Controls IO and DQS reset. **Value** **Description** 0x0 No reset. 0x1 Resets all IO registers and DQS registers.	RW	0x1
6	octrst	Controls OCT reset. **Value** **Description** 0x0 No reset. 0x1 Resets registers in the OCT.	RW	0x1

Bit	Name	Description	Access	Reset
5	`dllrst`	Controls DLL (Delay-Locked Loop) reset. **Value** **Description** 0x0 No reset or clock gating. 0x1 Resets registers in the DLL and gates off DLL clock.	RW	0x1
4	`slew`	Controls IO slew-rate **Value** **Description** 0x0 Slew-rate forced to slow. 0x1 Slew-rate controlled by IO configuration.	RW	0x0
3	`wkpullup`	Controls weak pullup resistor **Value** **Description** 0x0 Weak pullup resistor enabled. 0x1 Weak pullup resistor enable controlled by IO configuration.	RW	0x0
2	`tristate`	Controls IO tri-state **Value** **Description** 0x0 IO tri-state enabled. 0x1 IO tri-state controlled by IO configuration.	RW	0x0
1	`bushold`	Controls bus hold circuit **Value** **Description** 0x0 Disable bus hold circuit. 0x1 Bus hold circuit controlled by IO configuration.	RW	0x0
0	`cfg`	Controls IO configuration **Value** **Description** 0x0 Disable IO configuration (forced to a safe value). 0x1 Enables IO configuration as previously configured by software using the Scan Manager.	RW	0x0

src

Contains register field to choose between software state machine (vioctrl array index [1] register) or hardware state machine in the Freeze Controller as the freeze signal source for VIO channel 1. All fields are only reset by a cold reset (ignore warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08054

Offset: 0x54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															vio1 RW 0x0

src Fields

Bit	Name	Description	Access	Reset
0	vio1	The freeze signal source for VIO channel 1 (VIO bank 2 and bank 3). **Value** **Description** 0x0 VIO1 freeze signals are driven by software writing to the VIOCTRL[1] register. The VIO1-related fields in the hwctrl register are active but don't effect the VIO1 freeze signals. 0x1 VIO1 freeze signals are driven by the hardware state machine in the Freeze Controller. The VIO1-related fields in the hwctrl register are active and effect the VIO1 freeze signals.	RW	0x0

hwctrl

Activate freeze or thaw operations on VIO channel 1 (HPS IO bank 2 and bank 3) and monitor for completeness and the current state. These fields interact with the hardware state machine in the Freeze Controller. These fields can be accessed independent of the value of SRC1.VIO1 although they only have an effect on the VIO channel 1 freeze signals when SRC1.VIO1 is setup to have the hardware state machine be the freeze signal source. All fields are only reset by a cold reset (ignore warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08058

Offset: 0x58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													vio1state RO 0x2		vio1req RW 0x1

hwctrl Fields

Bit	Name	Description	Access	Reset
2:1	vio1state	Software reads this field to determine the current frozen/thawed state of the VIO channel 1 or to determine when a freeze/thaw request is made by writing the corresponding *REQ field in this register has completed. Reset by a cold reset (ignores warm reset). **Value** **Description** 0x0 Transitioning from thawed state to frozen state. 0x1 Thawed state. I/Os behave as configured. I/Os must be configured by the Scan Manager before entering this state. 0x2 Frozen state. I/O configuration is ignored. Instead, I/Os are in tri-state mode with a weak pull-up. Scan Manager can be used to configure the I/Os while they are frozen. 0x3 Transitioning from frozen state to thawed state.	RO	0x2
0	vio1req	Requests hardware state machine to generate freeze signal sequence to transition between frozen and thawed states. If this field is read by software, it contains the value previously written by software (i.e. this field is not written by hardware). **Value** **Description** 0x0 Requests a thaw (unfreeze) operation. 0x1 Requests a freeze operation.	RW	0x1

EMAC Group Register Descriptions

External control registers for the EMACs

Offset: 0x60

ctrl on page 5-42
Registers used by the EMACs. All fields are reset by a cold or warm reset.

l3master on page 5-43
Controls the L3 master ARCACHE and AWCACHE AXI signals. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

ctrl

Registers used by the EMACs. All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08060

Offset: 0x60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										ptpclksel_1 RW 0x0	ptpclksel_0 RW 0x0	physel_1 RW 0x2		physel_0 RW 0x2	

ctrl Fields

Bit	Name	Description	Access	Reset
5	ptpclksel_1	Selects the source of the 1588 PTP reference clock. This is sampled by an EMAC module when it exits from reset. The field array index corresponds to the EMAC index. **Value** **Description** 0x0 Selects osc1_clk 0x1 Selects fpga_ptp_ref_clk	RW	0x0

Bit	Name	Description	Access	Reset
4	ptpclksel_0	Selects the source of the 1588 PTP reference clock. This is sampled by an EMAC module when it exits from reset. The field array index corresponds to the EMAC index. **Value** — **Description** 0x0 — Selects osc1_clk 0x1 — Selects fpga_ptp_ref_clk	RW	0x0
3:2	physel_1	Controls the PHY interface selection of the EMACs. This is sampled by an EMAC module when it exits from reset. The associated enum defines the allowed values. The field array index corresponds to the EMAC index. **Value** — **Description** 0x0 — Select GMII/MII PHY interface 0x1 — Select RGMII PHY interface 0x2 — Select RMII PHY interface	RW	0x2
1:0	physel_0	Controls the PHY interface selection of the EMACs. This is sampled by an EMAC module when it exits from reset. The associated enum defines the allowed values. The field array index corresponds to the EMAC index. **Value** — **Description** 0x0 — Select GMII/MII PHY interface 0x1 — Select RGMII PHY interface 0x2 — Select RMII PHY interface	RW	0x2

l3master

Controls the L3 master ARCACHE and AWCACHE AXI signals. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08064

Offset: 0x64

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
awcache_1 RW 0x0				awcache_0 RW 0x0				arcache_1 RW 0x0				arcache_0 RW 0x0			

l3master Fields

Bit	Name	Description	Access	Reset
15:12	awcache_1	Specifies the values of the 2 EMAC AWCACHE signals. The field array index corresponds to the EMAC index. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	RW	0x0

Bit	Name	Description	Access	Reset
11:8	`awcache_0`	Specifies the values of the 2 EMAC AWCACHE signals. The field array index corresponds to the EMAC index. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
7:4	`arcache_1`	Specifies the values of the 2 EMAC ARCACHE signals. The field array index corresponds to the EMAC index. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
3:0	`arcache_0`	Specifies the values of the 2 EMAC ARCACHE signals. The field array index corresponds to the EMAC index. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	`RW`	`0x0`

DMA Controller Group Register Descriptions

Registers used by the DMA Controller to enable secured system support and select DMA channels.

Offset: `0x70`

ctrl on page 5-48
Registers used by the DMA Controller. All fields are reset by a cold or warm reset. These register bits should be updated during system initialization prior to removing the DMA controller from reset. They may not be changed dynamically during DMA operation.

persecurity on page 5-49
Controls the security state of a peripheral request interface. Sampled by the DMA controller when it exits from reset. These register bits should be updated during system initialization prior to removing the DMA controller from reset. They may not be changed dynamically during DMA operation.

ctrl

Registers used by the DMA Controller. All fields are reset by a cold or warm reset. These register bits should be updated during system initialization prior to removing the DMA controller from reset. They may not be changed dynamically during DMA operation.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08070

Offset: 0x70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			irqnonsecure RW 0x0								mgrnonsecure RW 0x0	chansel_3 RW 0x0	chansel_2 RW 0x0	chansel_1 RW 0x0	chansel_0 RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
12:5	irqnonsecure	Specifies the security state of an event-interrupt resource. If bit index [x] is 0, the DMAC assigns event<x> or irq[x] to the Secure state. If bit index [x] is 1, the DMAC assigns event<x> or irq[x] to the Non-secure state.	RW	0x0
4	mgrnonsecure	Specifies the security state of the DMA manager thread. 0 = assigns DMA manager to the Secure state. 1 = assigns DMA manager to the Non-secure state. Sampled by the DMA controller when it exits from reset.	RW	0x0
3	chansel_3	Controls mux that selects whether FPGA or CAN connects to one of the DMA peripheral request interfaces.The peripheral request interface index equals the array index + 4. For example, array index 0 is for peripheral request index 4. **Value** **Description** 0x0 FPGA drives peripheral request interface 0x1 CAN drives peripheral request interface	RW	0x0

Bit	Name	Description	Access	Reset
2	chansel_2	Controls mux that selects whether FPGA or CAN connects to one of the DMA peripheral request interfaces.The peripheral request interface index equals the array index + 4. For example, array index 0 is for peripheral request index 4. **Value** / **Description** 0x0 FPGA drives peripheral request interface 0x1 CAN drives peripheral request interface	RW	0x0
1	chansel_1	Controls mux that selects whether FPGA or CAN connects to one of the DMA peripheral request interfaces.The peripheral request interface index equals the array index + 4. For example, array index 0 is for peripheral request index 4. **Value** / **Description** 0x0 FPGA drives peripheral request interface 0x1 CAN drives peripheral request interface	RW	0x0
0	chansel_0	Controls mux that selects whether FPGA or CAN connects to one of the DMA peripheral request interfaces.The peripheral request interface index equals the array index + 4. For example, array index 0 is for peripheral request index 4. **Value** / **Description** 0x0 FPGA drives peripheral request interface 0x1 CAN drives peripheral request interface	RW	0x0

persecurity

Controls the security state of a peripheral request interface. Sampled by the DMA controller when it exits from reset. These register bits should be updated during system initialization prior to removing the DMA controller from reset. They may not be changed dynamically during DMA operation.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08074

Offset: 0x74

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
nonsecure RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nonsecure RW 0x0															

persecurity Fields

Bit	Name	Description	Access	Reset
31:0	nonsecure	If bit index [x] is 0, the DMA controller assigns peripheral request interface x to the Secure state. If bit index [x] is 1, the DMA controller assigns peripheral request interface x to the Non-secure state. Reset by a cold or warm reset.	RW	0x0

Preloader (initial software) Group Register Descriptions

Registers used by preloader code and the OS. All registers are only reset by a cold reset (ignore warm reset).

Offset: 0x80

handoff on page 5-50
These registers are used to store handoff infomation between the preloader and the OS. These 8 registers can be used to store any information. The contents of these registers have no impact on the state of the HPS hardware.

handoff

These registers are used to store handoff infomation between the preloader and the OS. These 8 registers can be used to store any information. The contents of these registers have no impact on the state of the HPS hardware.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08080

Offset: 0x80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

handoff Fields

Bit	Name	Description	Access	Reset
31:0	value	Preloader Handoff Information.	RW	0x0

Boot ROM Code Register Group Register Descriptions

Registers used by the Boot ROM code. All fields are only reset by a cold reset (ignore warm reset).

Offset: 0xc0

ctrl on page 5-51
Contains information used to control Boot ROM code.

cpu1startaddr on page 5-53
When CPU1 is released from reset and the Boot ROM is located at the CPU1 reset exception address (the typical case), the Boot ROM reset handler code reads the address stored in this register and jumps it to hand off execution to user software.

initswstate on page 5-53
The preloader software (loaded by the Boot ROM) writes the magic value 0x49535756 (ISWV in ASCII) to this register when it has reached a valid state.

initswlastld on page 5-54
Contains the index of the last preloader software image loaded by the Boot ROM from the boot device.

bootromswstate on page 5-54
32-bits general purpose register used by the Boot ROM code. Actual usage is defined in the Boot ROM source code.

Warm Boot from On-Chip RAM Group Register Descriptions on page 5-55
Registers used by the Boot ROM code to support booting from the On-chip RAM on a warm reset. All these registers must be written by user software before a warm reset occurs to make use of this feature.

ctrl

Contains information used to control Boot ROM code.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080C0

Offset: 0xC0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														warmrstcfgio RW 0x0	warmrstcfgpinmux RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
1	`warmrstcfgio`	Specifies whether the Boot ROM code configures the IOs used by boot after a warm reset. Note that the Boot ROM code always configures the IOs used by boot after a cold reset. After the Boot ROM code configures the IOs used by boot, it always disables this field. It is up to user software to enable this field if it wants a different behavior. **Value** **Description** 0x0 Boot ROM code will not configure IOs used by boot after a warm reset 0x1 Boot ROM code will configure IOs used by boot after a warm reset	RW	0x0
0	`warmrstcfgpinmux`	Specifies whether the Boot ROM code configures the pin mux for boot pins after a warm reset. Note that the Boot ROM code always configures the pin mux for boot pins after a cold reset. After the Boot ROM code configures the pin mux for boot pins, it always disables this field. It is up to user software to enable this field if it wants a different behavior. **Value** **Description** 0x0 Boot ROM code will not configure pin mux for boot pins after a warm reset 0x1 Boot ROM code will configure pin mux for boot pins after a warm reset	RW	0x0

cpu1startaddr

When CPU1 is released from reset and the Boot ROM is located at the CPU1 reset exception address (the typical case), the Boot ROM reset handler code reads the address stored in this register and jumps it to hand off execution to user software.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080C4

Offset: 0xC4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

cpu1startaddr Fields

Bit	Name	Description	Access	Reset
31:0	value	Address for CPU1 to start executing at after coming out of reset.	RW	0x0

initswstate

The preloader software (loaded by the Boot ROM) writes the magic value 0x49535756 (ISWV in ASCII) to this register when it has reached a valid state.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080C8

Offset: 0xC8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

initswstate Fields

Bit	Name	Description	Access	Reset
31:0	value	Written with magic value. **Value** **Description** 0x0 0x49535756	RW	0x0

initswlastld

Contains the index of the last preloader software image loaded by the Boot ROM from the boot device.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080CC

Offset: 0xCC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														index RW 0x0	

initswlastld Fields

Bit	Name	Description	Access	Reset
1:0	index	Index of last image loaded.	RW	0x0

bootromswstate

32-bits general purpose register used by the Boot ROM code. Actual usage is defined in the Boot ROM source code.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080D0

Offset: 0xD0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

bootromswstate Fields

Bit	Name	Description	Access	Reset
31:0	value	Reserved for Boot ROM use.	RW	0x0

Warm Boot from On-Chip RAM Group Register Descriptions

Registers used by the Boot ROM code to support booting from the On-chip RAM on a warm reset. All these registers must be written by user software before a warm reset occurs to make use of this feature.

Offset: 0x20

enable on page 5-55
Enables or disables the warm reset from On-chip RAM feature.

datastart on page 5-56
Offset into On-chip RAM of the start of the region for CRC validation

length on page 5-56
Length of region in On-chip RAM for CRC validation.

execution on page 5-57
Offset into On-chip RAM to enter to on a warm boot.

crc on page 5-58
Length of region in On-chip RAM for CRC validation.

enable

Enables or disables the warm reset from On-chip RAM feature.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080E0

Offset: 0xE0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
magic RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
magic RW 0x0															

enable Fields

Bit	Name	Description	Access	Reset
31:0	`magic`	Controls whether Boot ROM will attempt to boot from the contents of the On-chip RAM on a warm reset. When this feature is enabled, the Boot ROM code will not configure boot IOs, the pin mux, or clocks. Note that the enable value is a 32-bit magic value (provided by the enum). **Value** — **Description** 0x0 — Boot ROM code will not attempt to boot from On-chip RAM on a warm reset 0xae9efebc — Boot ROM code will attempt to boot from On-chip RAM on a warm reset	`RW`	`0x0`

datastart

Offset into On-chip RAM of the start of the region for CRC validation

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD080E4`

Offset: `0xE4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
offset RW 0x0															

datastart Fields

Bit	Name	Description	Access	Reset
15:0	`offset`	Contains the byte offset into the On-chip RAM of the start of the On-chip RAM region for the warm boot CRC validation. The offset must be an integer multiple of 4 (i.e. aligned to a word). The Boot ROM code will set the top 16 bits to 0xFFFF and clear the bottom 2 bits.	`RW`	`0x0`

length

Length of region in On-chip RAM for CRC validation.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080E8

Offset: 0xE8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
size RW 0x0															

length Fields

Bit	Name	Description	Access	Reset
15:0	size	Contains the length (in bytes) of the region in the On-chip RAM for the warm boot CRC validation. If the length is 0, the Boot ROM won't perform CRC calculation and CRC check to avoid overhead caused by CRC validation. If the START + LENGTH exceeds the maximum offset into the On-chip RAM, the Boot ROM won't boot from the On-chip RAM. The length must be an integer multiple of 4. The Boot ROM code will clear the top 16 bits and the bottom 2 bits.	RW	0x0

execution

Offset into On-chip RAM to enter to on a warm boot.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD080EC

Offset: 0xEC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
offset RW 0x0															

execution Fields

Bit	Name	Description	Access	Reset
15:0	`offset`	Contains the byte offset into the On-chip RAM that the Boot ROM will jump to if the CRC validation succeeds. The Boot ROM code will set the top 16 bits to 0xFFFF.	`RW`	`0x0`

crc

Length of region in On-chip RAM for CRC validation.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD080F0`

Offset: `0xF0`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">expected
RW 0xE763552A</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">expected
RW 0xE763552A</td></tr>
</table>

crc Fields

Bit	Name	Description	Access	Reset
31:0	`expected`	Contains the expected CRC of the region in the On-chip RAM.The Boot ROM code calculates the actual CRC for all bytes in the region specified by the DATA START an LENGTH registers. The contents of the EXECUTION register (after it has been read and modified by the Boot ROM code) is also included in the CRC calculation. The contents of the EXECUTION register is added to the CRC accumulator a byte at a time starting with the least significant byte. If the actual CRC doesn't match the expected CRC value in this register, the Boot ROM won't boot from the On-chip RAM. The CRC is a standard CRC32 with the polynomial: x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 There is no reflection of the bits and the initial value of the remainder is 0xFFFFFFFF and the final value is exclusive ORed with 0xFFFFFFFF.	`RW`	`0xE763552A`

Boot ROM Hardware Register Group Register Descriptions

Registers used by the Boot ROM hardware, not the code within it.

Offset: `0x100`

ctrl on page 5-59
Controls behavior of Boot ROM hardware. All fields are only reset by a cold reset (ignore warm reset).

ctrl

Controls behavior of Boot ROM hardware. All fields are only reset by a cold reset (ignore warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08100

Offset: `0x100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ensfmdwru RW 0x1	waitstate RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
1	ensfmdwru	Controls whether the wait state bit is updated upon deassertion of warm reset. This field is set on a cold reset. **Value** — **Description** 0x0 — Wait state bit is not updated upon deassertion of warm reset. 0x1 — Wait state bit is updated upon deassertion of warm reset. It's value is updated based on the control bit from clock manager which specifies whether clock manager will be in safe mode or not after warm reset.	RW	0x1

Bit	Name	Description	Access	Reset
0	`waitstate`	Controls the number of wait states applied to the Boot ROM's read operation. This field is cleared on a cold reset and optionally updated by hardware upon deassertion of warm reset. **Value** **Description** 0x0 No wait states are applied to the Boom ROM's read operation. 0x1 A single wait state is applied to the Boot ROM's read operation.	`RW`	`0x0`

SDMMC Controller Group Register Descriptions

Registers related to SDMMC Controller which aren't located inside the SDMMC itself.

Offset: `0x108`

ctrl on page 5-60
Registers used by the SDMMC Controller. All fields are reset by a cold or warm reset.

l3master on page 5-61
Controls the L3 master HPROT AHB-Lite signal. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

ctrl

Registers used by the SDMMC Controller. All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08108`

Offset: `0x108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`									`fbclksel` `RW 0x0`	`smplsel` `RW 0x0`			`drvsel` `RW 0x0`		

ctrl Fields

Bit	Name	Description	Access	Reset
6	`fbclksel`	Select which fb_clk to be used as cclk_in_sample. If 0, cclk_in_sample is driven by internal phase shifted cclk_in. If 1, cclk_in_sample is driven by fb_clk_in. No phase shifting is provided internally on cclk_in_sample. Note: Using the feedback clock (setting this bit to 1) is not a supported use model.	RW	0x0
5:3	`smplsel`	Select which phase shift of the clock for cclk_in_sample. **Value** **Description** 0x0 0 degrees phase shifted clock is selected 0x1 45 degrees phase shifted clock is selected 0x2 90 degrees phase shifted clock is selected 0x3 135 degrees phase shifted clock is selected 0x4 180 degrees phase shifted clock is selected 0x5 225 degrees phase shifted clock is selected 0x6 270 degrees phase shifted clock is selected 0x7 315 degrees phase shifted clock is selected	RW	0x0
2:0	`drvsel`	Select which phase shift of the clock for cclk_in_drv. **Value** **Description** 0x0 0 degrees phase shifted clock is selected 0x1 45 degrees phase shifted clock is selected 0x2 90 degrees phase shifted clock is selected 0x3 135 degrees phase shifted clock is selected 0x4 180 degrees phase shifted clock is selected 0x5 225 degrees phase shifted clock is selected 0x6 270 degrees phase shifted clock is selected 0x7 315 degrees phase shifted clock is selected	RW	0x0

l3master

Controls the L3 master HPROT AHB-Lite signal. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0810C`

Offset: `0x10C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												hprotcache_0 RW 0x0	hprotbuff_0 RW 0x0	hprotpriv_0 RW 0x1	hprotdata_0 RW 0x1

l3master Fields

Bit	Name	Description	Access	Reset
3	`hprotcache_0`	If 1, L3 master accesses for the SD/MMC module are cacheable.	`RW`	`0x0`
2	`hprotbuff_0`	If 1, L3 master accesses for the SD/MMC module are bufferable.	`RW`	`0x0`
1	`hprotpriv_0`	If 1, L3 master accesses for the SD/MMC module are privileged.	`RW`	`0x1`
0	`hprotdata_0`	Specifies if the L3 master access is for data or opcode for the SD/MMC module. **Value** / **Description** 0x0 — Opcode fetch 0x1 — Data access	`RW`	`0x1`

NAND Flash Controller Register Group Register Descriptions

Registers related to NAND Flash Controller which aren't located in the NAND Flash Controller itself.

Offset: `0x110`

bootstrap on page 5-63
Bootstrap fields sampled by NAND Flash Controller when released from reset. All fields are reset by a cold or warm reset.

l3master on page 5-63
Controls the L3 master ARCACHE and AWCACHE AXI signals. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

bootstrap

Bootstrap fields sampled by NAND Flash Controller when released from reset. All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08110

Offset: 0x110

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												tworowaddr RW 0x0	noloadb0p0 RW 0x0	page512 RW 0x0	noinit RW 0x0

bootstrap Fields

Bit	Name	Description	Access	Reset
3	tworowaddr	If 1, NAND device requires only 2 row address cycles instead of the normal 3 row address cycles.	RW	0x0
2	noloadb0p0	If 1, inhibits NAND Flash Controller from loading page 0 of block 0 of the NAND device as part of the initialization procedure.	RW	0x0
1	page512	If 1, NAND device has a 512 byte page size.	RW	0x0
0	noinit	If 1, inhibits NAND Flash Controller from performing initialization when coming out of reset. Instead, software must program all registers pertaining to device parameters like page size, width, etc.	RW	0x0

l3master

Controls the L3 master ARCACHE and AWCACHE AXI signals. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08114

Offset: 0x114

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								awcache_0 RW 0x0				arcache_0 RW 0x0			

l3master Fields

Bit	Name	Description	Access	Reset
7:4	awcache_0	Specifies the value of the module AWCACHE signal. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	RW	0x0

Bit	Name	Description	Access	Reset
3:0	`arcache_0`	Specifies the value of the module ARCACHE signal. **Value** **Description** 0x0 Noncacheable and nonbufferable. 0x1 Bufferable only. 0x2 Cacheable, but do not allocate. 0x3 Cacheable and bufferable, but do not allocate. 0x4 Reserved. 0x5 Reserved. 0x6 Cacheable write-through, allocate on reads only. 0x7 Cacheable write-back, allocate on reads only. 0x8 Reserved. 0x9 Reserved. 0xa Cacheable write-through, allocate on writes only. 0xb Cacheable write-back, allocate on writes only. 0xc Reserved. 0xd Reserved. 0xe Cacheable write-through, allocate on both reads and writes. 0xf Cacheable write-back, allocate on both reads and writes.	`RW`	`0x0`

USB Controller Group Register Descriptions

Registers related to USB Controllers which aren't located inside the USB controllers themselves.

Offset: `0x118`

l3master on page 5-65
Controls the L3 master HPROT AHB-Lite signal. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

l3master

Controls the L3 master HPROT AHB-Lite signal. These register bits should be updated only during system initialization prior to removing the peripheral from reset. They may not be changed dynamically during peripheral operation All fields are reset by a cold or warm reset.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08118`

Offset: `0x118`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								hprotcache_1 RW 0x0	hprotcache_0 RW 0x0	hprotbuff_1 RW 0x0	hprotbuff_0 RW 0x0	hprotpriv_1 RW 0x1	hprotpriv_0 RW 0x1	hprotdata_1 RW 0x1	hprotdata_0 RW 0x1

l3master Fields

Bit	Name	Description	Access	Reset
7	hprotcache_1	If 1, L3 master accesses for the USB modules are cacheable. The field array index corresponds to the USB index.	RW	0x0
6	hprotcache_0	If 1, L3 master accesses for the USB modules are cacheable. The field array index corresponds to the USB index.	RW	0x0
5	hprotbuff_1	If 1, L3 master accesses for the USB modules are bufferable. The field array index corresponds to the USB index.	RW	0x0
4	hprotbuff_0	If 1, L3 master accesses for the USB modules are bufferable. The field array index corresponds to the USB index.	RW	0x0
3	hprotpriv_1	If 1, L3 master accesses for the USB modules are privileged. The field array index corresponds to the USB index.	RW	0x1
2	hprotpriv_0	If 1, L3 master accesses for the USB modules are privileged. The field array index corresponds to the USB index.	RW	0x1

Bit	Name	Description	Access	Reset
1	`hprotdata_1`	Specifies if the L3 master access is for data or opcode for the USB modules. The field array index corresponds to the USB index. **Value** **Description** 0x0 Opcode fetch 0x1 Data access	RW	0x1
0	`hprotdata_0`	Specifies if the L3 master access is for data or opcode for the USB modules. The field array index corresponds to the USB index. **Value** **Description** 0x0 Opcode fetch 0x1 Data access	RW	0x1

ECC Management Register Group Register Descriptions

ECC error status and control for all ECC-protected HPS RAM blocks.

Offset: `0x140`

l2 on page 5-68
This register is used to enable ECC on the L2 Data RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

ocram on page 5-69
This register is used to enable ECC on the On-chip RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

usb0 on page 5-70
This register is used to enable ECC on the USB0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

usb1 on page 5-71
This register is used to enable ECC on the USB1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

emac0 on page 5-72
This register is used to enable ECC on the EMAC0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

emac1 on page 5-73
This register is used to enable ECC on the EMAC1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

dma on page 5-75
This register is used to enable ECC on the DMA RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

can0 on page 5-76
This register is used to enable ECC on the CAN0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

can1 on page 5-77
This register is used to enable ECC on the CAN1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

nand on page 5-78
This register is used to enable ECC on the NAND RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

qspi on page 5-80
This register is used to enable ECC on the QSPI RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

sdmmc on page 5-80
This register is used to enable ECC on the SDMMC RAM.ECC errors can be injected into the write path using bits in this register. Only reset by a cold reset (ignores warm reset).

l2

This register is used to enable ECC on the L2 Data RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08140

Offset: 0x140

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													injd RW 0x0	injs RW 0x0	en RW 0x0

l2 Fields

Bit	Name	Description	Access	Reset
2	`injd`	Changing this bit from zero to one injects a double, non-correctable error into the L2 Data RAM. This only injects one double bit error into the L2 Data RAM.	`RW`	`0x0`
1	`injs`	Changing this bit from zero to one injects a single, correctable error into the L2 Data RAM. This only injects one error into the L2 Data RAM.	`RW`	`0x0`
0	`en`	Enable ECC for L2 Data RAM	`RW`	`0x0`

ocram

This register is used to enable ECC on the On-chip RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08144`

Offset: `0x144`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											derr RW 0x0	serr RW 0x0	injd RW 0x0	injs RW 0x0	en RW 0x0

ocram Fields

Bit	Name	Description	Access	Reset
4	`derr`	This bit is an interrupt status bit for On-chip RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in On-chip RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`
3	`serr`	This bit is an interrupt status bit for On-chip RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in On-chip RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
2	injd	Changing this bit from zero to one injects a double, non-correctable error into the On-chip RAM. This only injects one double bit error into the On-chip RAM.	RW	0x0
1	injs	Changing this bit from zero to one injects a single, correctable error into the On-chip RAM. This only injects one error into the On-chip RAM.	RW	0x0
0	en	Enable ECC for On-chip RAM	RW	0x0

usb0

This register is used to enable ECC on the USB0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08148

Offset: 0x148

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="11">Reserved</td><td>derr
RW
0x0</td><td>serr
RW
0x0</td><td>injd
RW
0x0</td><td>injs
RW
0x0</td><td>en
RW 0x0</td></tr>
</table>

usb0 Fields

Bit	Name	Description	Access	Reset
4	derr	This bit is an interrupt status bit for USB0 RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in USB0 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
3	serr	This bit is an interrupt status bit for USB0 RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in USB0 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0

Bit	Name	Description	Access	Reset
2	`injd`	Changing this bit from zero to one injects a double, non-correctable error into the USB0 RAM. This only injects one double bit error into the USB0 RAM.	`RW`	`0x0`
1	`injs`	Changing this bit from zero to one injects a single, correctable error into the USB0 RAM. This only injects one error into the USB0 RAM.	`RW`	`0x0`
0	`en`	Enable ECC for USB0 RAM	`RW`	`0x0`

usb1

This register is used to enable ECC on the USB1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0814C`

Offset: `0x14C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`											`derr` `RW` `0x0`	`serr` `RW` `0x0`	`injd` `RW` `0x0`	`injs` `RW` `0x0`	`en` `RW 0x0`

usb1 Fields

Bit	Name	Description	Access	Reset
4	`derr`	This bit is an interrupt status bit for USB1 RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in USB1 RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`
3	`serr`	This bit is an interrupt status bit for USB1 RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in USB1 RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`
2	`injd`	Changing this bit from zero to one injects a double, non-correctable error into the USB1 RAM. This only injects one double bit error into the USB1 RAM.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
1	injs	Changing this bit from zero to one injects a single, correctable error into the USB1 RAM. This only injects one error into the USB1 RAM.	RW	0x0
0	en	Enable ECC for USB1 RAM	RW	0x0

emac0

This register is used to enable ECC on the EMAC0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08150

Offset: 0x150

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rxfifoderr RW 0x0	rxfifoserr RW 0x0	txfifoderr RW 0x0	txfifoserr RW 0x0	rxfifoinjd RW 0x0	rxfifoinjs RW 0x0	txfifoinjd RW 0x0	txfifoinjs RW 0x0	en RW 0x0

emac0 Fields

Bit	Name	Description	Access	Reset
8	rxfifoderr	This bit is an interrupt status bit for EMAC0 RXFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in EMAC0 RXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
7	rxfifoserr	This bit is an interrupt status bit for EMAC0 RXFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in EMAC0 RXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0

Bit	Name	Description	Access	Reset
6	`txfifoderr`	This bit is an interrupt status bit for EMAC0 TXFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in EMAC0 TXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
5	`txfifoserr`	This bit is an interrupt status bit for EMAC0 TXFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in EMAC0 TXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
4	`rxfifoinjd`	Changing this bit from zero to one injects a double, non-correctable error into the EMAC0 RXFIFO RAM. This only injects one double bit error into the EMAC0 RXFIFO RAM.	RW	0x0
3	`rxfifoinjs`	Changing this bit from zero to one injects a single, correctable error into the EMAC0 RXFIFO RAM. This only injects one error into the EMAC0 RXFIFO RAM.	RW	0x0
2	`txfifoinjd`	Changing this bit from zero to one injects a double, non-correctable error into the EMAC0 TXFIFO RAM. This only injects one double bit error into the EMAC0 TXFIFO RAM.	RW	0x0
1	`txfifoinjs`	Changing this bit from zero to one injects a single, correctable error into the EMAC0 TXFIFO RAM. This only injects one error into the EMAC0 TXFIFO RAM.	RW	0x0
0	`en`	Enable ECC for EMAC0 RAM	RW	0x0

emac1

This register is used to enable ECC on the EMAC1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08154`

Offset: `0x154`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rxfifoderr RW 0x0	rxfifoserr RW 0x0	txfifoderr RW 0x0	txfifoserr RW 0x0	rxfifoinjd RW 0x0	rxfifoinjs RW 0x0	txfifoinjd RW 0x0	txfifoinjs RW 0x0	en RW 0x0

emac1 Fields

Bit	Name	Description	Access	Reset
8	rxfifoderr	This bit is an interrupt status bit for EMAC1 RXFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in EMAC1 RXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
7	rxfifoserr	This bit is an interrupt status bit for EMAC1 RXFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in EMAC1 RXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
6	txfifoderr	This bit is an interrupt status bit for EMAC1 TXFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in EMAC1 TXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
5	txfifoserr	This bit is an interrupt status bit for EMAC1 TXFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in EMAC1 TXFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
4	rxfifoinjd	Changing this bit from zero to one injects a double, non-correctable error into the EMAC1 RXFIFO RAM. This only injects one double bit error into the EMAC1 RXFIFO RAM.	RW	0x0
3	rxfifoinjs	Changing this bit from zero to one injects a single, correctable error into the EMAC1 RXFIFO RAM. This only injects one error into the EMAC1 RXFIFO RAM.	RW	0x0

Bit	Name	Description	Access	Reset
2	`txfifoinjd`	Changing this bit from zero to one injects a double, non-correctable error into the EMAC1 TXFIFO RAM. This only injects one double bit error into the EMAC1 TXFIFO RAM.	`RW`	`0x0`
1	`txfifoinjs`	Changing this bit from zero to one injects a single, correctable error into the EMAC1 TXFIFO RAM. This only injects one error into the EMAC1 TXFIFO RAM.	`RW`	`0x0`
0	`en`	Enable ECC for EMAC1 RAM	`RW`	`0x0`

dma

This register is used to enable ECC on the DMA RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08158`

Offset: `0x158`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`											`derr` `RW` `0x0`	`serr` `RW` `0x0`	`injd` `RW` `0x0`	`injs` `RW` `0x0`	`en` `RW 0x0`

dma Fields

Bit	Name	Description	Access	Reset
4	`derr`	This bit is an interrupt status bit for DMA RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in DMA RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`
3	`serr`	This bit is an interrupt status bit for DMA RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in DMA RAM. Software needs to write 1 into this bit to clear the interrupt status.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
2	injd	Changing this bit from zero to one injects a double, non-correctable error into the DMA RAM. This only injects one double bit error into the DMA RAM.	RW	0x0
1	injs	Changing this bit from zero to one injects a single, correctable error into the DMA RAM. This only injects one error into the DMA RAM.	RW	0x0
0	en	Enable ECC for DMA RAM	RW	0x0

can0

This register is used to enable ECC on the CAN0 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0815C

Offset: 0x15C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											derr RW 0x0	serr RW 0x0	injd RW 0x0	injs RW 0x0	en RW 0x0

can0 Fields

Bit	Name	Description	Access	Reset
4	derr	This bit is an interrupt status bit for CAN0 RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in CAN0 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
3	serr	This bit is an interrupt status bit for CAN0 RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in CAN0 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
2	injd	Changing this bit from zero to one injects a double, non-correctable error into the CAN0 RAM. This only injects one double bit error into the CAN0 RAM.	RW	0x0

Bit	Name	Description	Access	Reset
1	injs	Changing this bit from zero to one injects a single, correctable error into the CAN0 RAM. This only injects one error into the CAN0 RAM.	RW	0x0
0	en	Enable ECC for CAN0 RAM	RW	0x0

can1

This register is used to enable ECC on the CAN1 RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08160

Offset: 0x160

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											derr RW 0x0	serr RW 0x0	injd RW 0x0	injs RW 0x0	en RW 0x0

can1 Fields

Bit	Name	Description	Access	Reset
4	derr	This bit is an interrupt status bit for CAN1 RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in CAN1 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
3	serr	This bit is an interrupt status bit for CAN1 RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in CAN1 RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
2	injd	Changing this bit from zero to one injects a double, non-correctable error into the CAN1 RAM. This only injects one double bit error into the CAN1 RAM.	RW	0x0
1	injs	Changing this bit from zero to one injects a single, correctable error into the CAN1 RAM. This only injects one error into the CAN1 RAM.	RW	0x0

Bit	Name	Description	Access	Reset
0	en	Enable ECC for CAN1 RAM	RW	0x0

nand

This register is used to enable ECC on the NAND RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08164

Offset: 0x164

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			rdfifoderr RW 0x0	rdfifoserr RW 0x0	wrfifoderr RW 0x0	wrfifoserr RW 0x0	eccbufderr RW 0x0	eccbufserr RW 0x0	rdfifoinjd RW 0x0	rdfifoinjs RW 0x0	wrfifoinjd RW 0x0	wrfifoinjs RW 0x0	eccbufinjd RW 0x0	eccbufinjs RW 0x0	en RW 0x0

nand Fields

Bit	Name	Description	Access	Reset
12	rdfifoderr	This bit is an interrupt status bit for NAND RDFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in NAND RDFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
11	rdfifoserr	This bit is an interrupt status bit for NAND RDFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in NAND RDFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
10	wrfifoderr	This bit is an interrupt status bit for NAND WRFIFO RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in NAND WRFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0

Bit	Name	Description	Access	Reset
9	wrfifoserr	This bit is an interrupt status bit for NAND WRFIFO RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in NAND WRFIFO RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
8	eccbufderr	This bit is an interrupt status bit for NAND ECCBUFFER RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in NAND ECCBUFFER RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
7	eccbufserr	This bit is an interrupt status bit for NAND ECCBUFFER RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in NAND ECCBUFFER RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
6	rdfifoinjd	Changing this bit from zero to one injects a double, non-correctable error into the NAND RDFIFO RAM. This only injects one double bit error into the NAND RDFIFO RAM.	RW	0x0
5	rdfifoinjs	Changing this bit from zero to one injects a single, correctable error into the NAND RDFIFO RAM. This only injects one error into the NAND RDFIFO RAM.	RW	0x0
4	wrfifoinjd	Changing this bit from zero to one injects a double, non-correctable error into the NAND WRFIFO RAM. This only injects one double bit error into the NAND WRFIFO RAM.	RW	0x0
3	wrfifoinjs	Changing this bit from zero to one injects a single, correctable error into the NAND WRFIFO RAM. This only injects one error into the NAND WRFIFO RAM.	RW	0x0
2	eccbufinjd	Changing this bit from zero to one injects a double, non-correctable error into the NAND ECCBUFFER RAM. This only injects one double bit error into the NAND ECCBUFFER RAM.	RW	0x0
1	eccbufinjs	Changing this bit from zero to one injects a single, correctable error into the NAND ECCBUFFER RAM. This only injects one error into the NAND ECCBUFFER RAM.	RW	0x0
0	en	Enable ECC for NAND RAM	RW	0x0

qspi

This register is used to enable ECC on the QSPI RAM. ECC errors can be injected into the write path using bits in this register. This register contains interrupt status of the ECC single/double bit error. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08168

Offset: 0x168

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											derr RW 0x0	serr RW 0x0	injd RW 0x0	injs RW 0x0	en RW 0x0

qspi Fields

Bit	Name	Description	Access	Reset
4	derr	This bit is an interrupt status bit for QSPI RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in QSPI RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
3	serr	This bit is an interrupt status bit for QSPI RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in QSPI RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
2	injd	Changing this bit from zero to one injects a double, non-correctable error into the QSPI RAM. This only injects one double bit error into the QSPI RAM.	RW	0x0
1	injs	Changing this bit from zero to one injects a single, correctable error into the QSPI RAM. This only injects one error into the QSPI RAM.	RW	0x0
0	en	Enable ECC for QSPI RAM	RW	0x0

sdmmc

This register is used to enable ECC on the SDMMC RAM.ECC errors can be injected into the write path using bits in this register. Only reset by a cold reset (ignores warm reset).

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0816C`

Offset: `0x16C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							derrportb RW 0x0	serrportb RW 0x0	derrporta RW 0x0	serrporta RW 0x0	injdportb RW 0x0	injsportb RW 0x0	injdporta RW 0x0	injsporta RW 0x0	en RW 0x0

sdmmc Fields

Bit	Name	Description	Access	Reset
8	`derrportb`	This bit is an interrupt status bit for SDMMC Port B RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in SDMMC Port B RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
7	`serrportb`	This bit is an interrupt status bit for SDMMC Port B RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in SDMMC Port B RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
6	`derrporta`	This bit is an interrupt status bit for SDMMC Port A RAM ECC double bit, non-correctable error. It is set by hardware when double bit, non-correctable error occurs in SDMMC Port A RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
5	`serrporta`	This bit is an interrupt status bit for SDMMC Port A RAM ECC single, correctable error. It is set by hardware when single, correctable error occurs in SDMMC Port A RAM. Software needs to write 1 into this bit to clear the interrupt status.	RW	0x0
4	`injdportb`	Changing this bit from zero to one injects a double, non-correctable error into the SDMMC RAM at Port B. This only injects one double bit error into the SDMMC RAM at Port B.	RW	0x0

Bit	Name	Description	Access	Reset
3	`injsportb`	Changing this bit from zero to one injects a single, correctable error into the SDMMC RAM at Port B. This only injects one error into the SDMMC RAM at Port B.	RW	0x0
2	`injdporta`	Changing this bit from zero to one injects a double, non-correctable error into the SDMMC RAM at Port A. This only injects one double bit error into the SDMMC RAM at Port A.	RW	0x0
1	`injsporta`	Changing this bit from zero to one injects a single, correctable error into the SDMMC RAM at Port A. This only injects one error into the SDMMC RAM at Port A.	RW	0x0
0	`en`	Enable ECC for SDMMC RAM	RW	0x0

Pin Mux Control Group Register Descriptions

Controls Pin Mux selections NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Offset: `0x400`

EMACIO0 on page 5-101
This register is used to control the peripherals connected to emac0_tx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO1 on page 5-102
This register is used to control the peripherals connected to emac0_tx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO2 on page 5-103
This register is used to control the peripherals connected to emac0_tx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO3 on page 5-103
This register is used to control the peripherals connected to emac0_tx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO4 on page 5-104
This register is used to control the peripherals connected to emac0_tx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO5 on page 5-104
This register is used to control the peripherals connected to emac0_rx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO6 on page 5-105
This register is used to control the peripherals connected to emac0_mdio Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO7 on page 5-105
This register is used to control the peripherals connected to emac0_mdc Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO8 on page 5-106
This register is used to control the peripherals connected to emac0_rx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO9 on page 5-107
This register is used to control the peripherals connected to emac0_tx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO10 on page 5-107
This register is used to control the peripherals connected to emac0_rx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO11 on page 5-108
This register is used to control the peripherals connected to emac0_rx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO12 on page 5-109
This register is used to control the peripherals connected to emac0_rx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO13 on page 5-109
This register is used to control the peripherals connected to emac0_rx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO14 on page 5-110
This register is used to control the peripherals connected to emac1_tx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO15 on page 5-110
This register is used to control the peripherals connected to emac1_tx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO16 on page 5-111
This register is used to control the peripherals connected to emac1_tx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO17 on page 5-112
This register is used to control the peripherals connected to emac1_tx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO18 on page 5-112
This register is used to control the peripherals connected to emac1_rx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO19 on page 5-113
This register is used to control the peripherals connected to emac1_rx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO0 on page 5-113
This register is used to control the peripherals connected to sdmmc_cmd Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO1 on page 5-114
This register is used to control the peripherals connected to sdmmc_pwren Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO2 on page 5-115
This register is used to control the peripherals connected to sdmmc_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO3 on page 5-115
This register is used to control the peripherals connected to sdmmc_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO4 on page 5-116
This register is used to control the peripherals connected to sdmmc_d4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO5 on page 5-116
This register is used to control the peripherals connected to sdmmc_d5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO6 on page 5-117
This register is used to control the peripherals connected to sdmmc_d6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO7 on page 5-117
This register is used to control the peripherals connected to sdmmc_d7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO8 on page 5-118
This register is used to control the peripherals connected to sdmmc_clk_in Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO9 on page 5-119
This register is used to control the peripherals connected to sdmmc_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO10 on page 5-119
This register is used to control the peripherals connected to sdmmc_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

FLASHIO11 on page 5-120
This register is used to control the peripherals connected to sdmmc_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO0 on page 5-120
This register is used to control the peripherals connected to trace_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO1 on page 5-121
This register is used to control the peripherals connected to trace_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO2 on page 5-122
This register is used to control the peripherals connected to trace_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO3 on page 5-122
This register is used to control the peripherals connected to trace_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO4 on page 5-123
This register is used to control the peripherals connected to trace_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO5 on page 5-123
This register is used to control the peripherals connected to trace_d4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO6 on page 5-124
This register is used to control the peripherals connected to trace_d5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO7 on page 5-125
This register is used to control the peripherals connected to trace_d6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO8 on page 5-125
This register is used to control the peripherals connected to trace_d7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO9 on page 5-126
This register is used to control the peripherals connected to spim0_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO10 on page 5-126
This register is used to control the peripherals connected to spim0_mosi Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO11 on page 5-127
This register is used to control the peripherals connected to spim0_miso Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO12 on page 5-128
This register is used to control the peripherals connected to spim0_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO13 on page 5-128
This register is used to control the peripherals connected to uart0_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO14 on page 5-129
This register is used to control the peripherals connected to uart0_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO15 on page 5-129
This register is used to control the peripherals connected to i2c0_sda Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO16 on page 5-130
This register is used to control the peripherals connected to i2c0_scl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO17 on page 5-131
This register is used to control the peripherals connected to can0_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO18 on page 5-131
This register is used to control the peripherals connected to can0_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO19 on page 5-132
This register is used to control the peripherals connected to spis1_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO20 on page 5-132
This register is used to control the peripherals connected to spis1_mosi Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO21 on page 5-133
This register is used to control the peripherals connected to spis1_miso Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO22 on page 5-134
This register is used to control the peripherals connected to spis1_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO23 on page 5-134
This register is used to control the peripherals connected to uart1_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO24 on page 5-135
This register is used to control the peripherals connected to uart1_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO25 on page 5-135
This register is used to control the peripherals connected to i2c1_sda Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO26 on page 5-136
This register is used to control the peripherals connected to i2c1_scl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GENERALIO27 on page 5-137
This register is used to control the peripherals connected to spim0_ss0_alt Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO0 on page 5-137
This register is used to control the peripherals connected to nand_ale Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO1 on page 5-138
This register is used to control the peripherals connected to nand_ce Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO2 on page 5-138
This register is used to control the peripherals connected to nand_cle Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO3 on page 5-139
This register is used to control the peripherals connected to nand_re Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO4 on page 5-140
This register is used to control the peripherals connected to nand_rb Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO5 on page 5-140
This register is used to control the peripherals connected to nand_dq0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO6 on page 5-141
This register is used to control the peripherals connected to nand_dq1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO7 on page 5-141
This register is used to control the peripherals connected to nand_dq2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO8 on page 5-142
This register is used to control the peripherals connected to nand_dq3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO9 on page 5-143
This register is used to control the peripherals connected to nand_dq4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO10 on page 5-143
This register is used to control the peripherals connected to nand_dq5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO11 on page 5-144
This register is used to control the peripherals connected to nand_dq6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO12 on page 5-144
This register is used to control the peripherals connected to nand_dq7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO13 on page 5-145
This register is used to control the peripherals connected to nand_wp Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO14 on page 5-146
This register is used to control the peripherals connected to nand_we Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO15 on page 5-146
This register is used to control the peripherals connected to qspi_io0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO16 on page 5-147
This register is used to control the peripherals connected to qspi_io1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO17 on page 5-147
This register is used to control the peripherals connected to qspi_io2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO18 on page 5-148
This register is used to control the peripherals connected to qspi_io3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO19 on page 5-149
This register is used to control the peripherals connected to qspi_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO20 on page 5-149
This register is used to control the peripherals connected to qpsi_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED1IO21 on page 5-150
This register is used to control the peripherals connected to qspi_ss1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO0 on page 5-150
This register is used to control the peripherals connected to emac1_mdio Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO1 on page 5-151
This register is used to control the peripherals connected to emac1_mdc Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO2 on page 5-152
This register is used to control the peripherals connected to emac1_tx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO3 on page 5-152
This register is used to control the peripherals connected to emac1_tx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO4 on page 5-153
This register is used to control the peripherals connected to emac1_rx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO5 on page 5-153
This register is used to control the peripherals connected to emac1_rx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO6 on page 5-154
This register is used to control the peripherals connected to emac1_rx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

MIXED2IO7 on page 5-155
This register is used to control the peripherals connected to emac1_rx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX48 on page 5-155
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 48. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX49 on page 5-156
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 49. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX50 on page 5-156
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 50. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX51 on page 5-157
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 51. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX52 on page 5-157
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 52. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX53 on page 5-158
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 53. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX54 on page 5-159
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 54. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX55 on page 5-159
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 55. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX56 on page 5-160
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 56. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX57 on page 5-160
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 57. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX58 on page 5-161
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 58. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX59 on page 5-161
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 59. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX60 on page 5-162
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 60. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX61 on page 5-162
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 61. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX62 on page 5-163
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 62. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX63 on page 5-164
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 63. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX64 on page 5-164
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 64. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX65 on page 5-165
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 65. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX66 on page 5-165
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 66. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX67 on page 5-166
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 67. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX68 on page 5-166
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 68. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX69 on page 5-167
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 69. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLINMUX70 on page 5-167
Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 70. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX0 on page 5-168
Selection between GPIO and LoanIO output and output enable for GPIO0 and LoanIO0. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX1 on page 5-169
Selection between GPIO and LoanIO output and output enable for GPIO1 and LoanIO1. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX2 on page 5-169
Selection between GPIO and LoanIO output and output enable for GPIO2 and LoanIO2. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX3 on page 5-170
Selection between GPIO and LoanIO output and output enable for GPIO3 and LoanIO3. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX4 on page 5-170
Selection between GPIO and LoanIO output and output enable for GPIO4 and LoanIO4. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX5 on page 5-171
Selection between GPIO and LoanIO output and output enable for GPIO5 and LoanIO5. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX6 on page 5-171
Selection between GPIO and LoanIO output and output enable for GPIO6 and LoanIO6. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX7 on page 5-172
Selection between GPIO and LoanIO output and output enable for GPIO7 and LoanIO7. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX8 on page 5-173
Selection between GPIO and LoanIO output and output enable for GPIO8 and LoanIO8. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX9 on page 5-173
Selection between GPIO and LoanIO output and output enable for GPIO9 and LoanIO9. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX10 on page 5-174
Selection between GPIO and LoanIO output and output enable for GPIO10 and LoanIO10. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX11 on page 5-174
Selection between GPIO and LoanIO output and output enable for GPIO11 and LoanIO11. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX12 on page 5-175
Selection between GPIO and LoanIO output and output enable for GPIO12 and LoanIO12. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX13 on page 5-176
Selection between GPIO and LoanIO output and output enable for GPIO13 and LoanIO13. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX14 on page 5-176
Selection between GPIO and LoanIO output and output enable for GPIO14 and LoanIO14. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX15 on page 5-177
Selection between GPIO and LoanIO output and output enable for GPIO15 and LoanIO15. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX16 on page 5-177
Selection between GPIO and LoanIO output and output enable for GPIO16 and LoanIO16. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX17 on page 5-178
Selection between GPIO and LoanIO output and output enable for GPIO17 and LoanIO17. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX18 on page 5-179
Selection between GPIO and LoanIO output and output enable for GPIO18 and LoanIO18. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX19 on page 5-179
Selection between GPIO and LoanIO output and output enable for GPIO19 and LoanIO19. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX20 on page 5-180
Selection between GPIO and LoanIO output and output enable for GPIO20 and LoanIO20. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX21 on page 5-180
Selection between GPIO and LoanIO output and output enable for GPIO21 and LoanIO21. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX22 on page 5-181
Selection between GPIO and LoanIO output and output enable for GPIO22 and LoanIO22. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX23 on page 5-182
Selection between GPIO and LoanIO output and output enable for GPIO23 and LoanIO23. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX24 on page 5-182
Selection between GPIO and LoanIO output and output enable for GPIO24 and LoanIO24. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX25 on page 5-183
Selection between GPIO and LoanIO output and output enable for GPIO25 and LoanIO25. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX26 on page 5-183
Selection between GPIO and LoanIO output and output enable for GPIO26 and LoanIO26. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX27 on page 5-184
Selection between GPIO and LoanIO output and output enable for GPIO27 and LoanIO27. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX28 on page 5-185
Selection between GPIO and LoanIO output and output enable for GPIO28 and LoanIO28. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX29 on page 5-185
Selection between GPIO and LoanIO output and output enable for GPIO29 and LoanIO29. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX30 on page 5-186
Selection between GPIO and LoanIO output and output enable for GPIO30 and LoanIO30. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX31 on page 5-186
Selection between GPIO and LoanIO output and output enable for GPIO31 and LoanIO31. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX32 on page 5-187
Selection between GPIO and LoanIO output and output enable for GPIO32 and LoanIO32. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX33 on page 5-188
Selection between GPIO and LoanIO output and output enable for GPIO33 and LoanIO33. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX34 on page 5-188
Selection between GPIO and LoanIO output and output enable for GPIO34 and LoanIO34. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX35 on page 5-189
Selection between GPIO and LoanIO output and output enable for GPIO35 and LoanIO35. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX36 on page 5-189
Selection between GPIO and LoanIO output and output enable for GPIO36 and LoanIO36. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX37 on page 5-190
Selection between GPIO and LoanIO output and output enable for GPIO37 and LoanIO37. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX38 on page 5-191
Selection between GPIO and LoanIO output and output enable for GPIO38 and LoanIO38. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX39 on page 5-191
Selection between GPIO and LoanIO output and output enable for GPIO39 and LoanIO39. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX40 on page 5-192
Selection between GPIO and LoanIO output and output enable for GPIO40 and LoanIO40. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX41 on page 5-192
Selection between GPIO and LoanIO output and output enable for GPIO41 and LoanIO41. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX42 on page 5-193
Selection between GPIO and LoanIO output and output enable for GPIO42 and LoanIO42. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX43 on page 5-194
Selection between GPIO and LoanIO output and output enable for GPIO43 and LoanIO43. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX44 on page 5-194
Selection between GPIO and LoanIO output and output enable for GPIO44 and LoanIO44. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX45 on page 5-195
Selection between GPIO and LoanIO output and output enable for GPIO45 and LoanIO45. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX46 on page 5-195
Selection between GPIO and LoanIO output and output enable for GPIO46 and LoanIO46. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX47 on page 5-196
Selection between GPIO and LoanIO output and output enable for GPIO47 and LoanIO47. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX48 on page 5-197
Selection between GPIO and LoanIO output and output enable for GPIO48 and LoanIO48. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX49 on page 5-197
Selection between GPIO and LoanIO output and output enable for GPIO49 and LoanIO49. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX50 on page 5-198
Selection between GPIO and LoanIO output and output enable for GPIO50 and LoanIO50. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX51 on page 5-198
Selection between GPIO and LoanIO output and output enable for GPIO51 and LoanIO51. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX52 on page 5-199
Selection between GPIO and LoanIO output and output enable for GPIO52 and LoanIO52. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX53 on page 5-200
Selection between GPIO and LoanIO output and output enable for GPIO53 and LoanIO53. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX54 on page 5-200
Selection between GPIO and LoanIO output and output enable for GPIO54 and LoanIO54. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX55 on page 5-201
Selection between GPIO and LoanIO output and output enable for GPIO55 and LoanIO55. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX56 on page 5-201
Selection between GPIO and LoanIO output and output enable for GPIO56 and LoanIO56. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX57 on page 5-202
Selection between GPIO and LoanIO output and output enable for GPIO57 and LoanIO57. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX58 on page 5-203
Selection between GPIO and LoanIO output and output enable for GPIO58 and LoanIO58. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX59 on page 5-203
Selection between GPIO and LoanIO output and output enable for GPIO59 and LoanIO59. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX60 on page 5-204
Selection between GPIO and LoanIO output and output enable for GPIO60 and LoanIO60. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX61 on page 5-204
Selection between GPIO and LoanIO output and output enable for GPIO61 and LoanIO61. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX62 on page 5-205
Selection between GPIO and LoanIO output and output enable for GPIO62 and LoanIO62. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX63 on page 5-206
Selection between GPIO and LoanIO output and output enable for GPIO63 and LoanIO63. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX64 on page 5-206
Selection between GPIO and LoanIO output and output enable for GPIO64 and LoanIO64. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX65 on page 5-207
Selection between GPIO and LoanIO output and output enable for GPIO65 and LoanIO65. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX66 on page 5-207
Selection between GPIO and LoanIO output and output enable for GPIO66 and LoanIO66. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX67 on page 5-208
Selection between GPIO and LoanIO output and output enable for GPIO67 and LoanIO67. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX68 on page 5-209
Selection between GPIO and LoanIO output and output enable for GPIO68 and LoanIO68. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX69 on page 5-209
Selection between GPIO and LoanIO output and output enable for GPIO69 and LoanIO69. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

GPLMUX70 on page 5-210
Selection between GPIO and LoanIO output and output enable for GPIO70 and LoanIO70. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

NANDUSEFPGA on page 5-210
Selection between HPS Pins and FPGA Interface for NAND signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

RGMII1USEFPGA on page 5-211
Selection between HPS Pins and FPGA Interface for RGMII1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

I2C0USEFPGA on page 5-212
Selection between HPS Pins and FPGA Interface for I2C0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

RGMII0USEFPGA on page 5-212
Selection between HPS Pins and FPGA Interface for RGMII0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

I2C3USEFPGA on page 5-213
Selection between HPS Pins and FPGA Interface for I2C3 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

I2C2USEFPGA on page 5-213
Selection between HPS Pins and FPGA Interface for I2C2 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

I2C1USEFPGA on page 5-214
Selection between HPS Pins and FPGA Interface for I2C1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

SPIM1USEFPGA on page 5-214
Selection between HPS Pins and FPGA Interface for SPIM1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

SPIM0USEFPGA on page 5-215
Selection between HPS Pins and FPGA Interface for SPIM0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

EMACIO0

This register is used to control the peripherals connected to emac0_tx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08400

Offset: 0x400

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO0 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_tx_clk. 0 : Pin is connected to GPIO/LoanIO number 0. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII0.TX_CLK.	RW	0x0

EMACIO1

This register is used to control the peripherals connected to emac0_tx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08404

Offset: 0x404

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO1 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_tx_d0. 0 : Pin is connected to GPIO/LoanIO number 1. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D0. 3 : Pin is connected to Peripheral signal RGMII0.TXD0.	RW	0x0

EMACIO2

This register is used to control the peripherals connected to emac0_tx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08408

Offset: 0x408

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO2 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_tx_d1. 0 : Pin is connected to GPIO/LoanIO number 2. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D1. 3 : Pin is connected to Peripheral signal RGMII0.TXD1.	RW	0x0

EMACIO3

This register is used to control the peripherals connected to emac0_tx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0840C

Offset: 0x40C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO3 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac0_tx_d2. 0 : Pin is connected to GPIO/LoanIO number 3. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D2. 3 : Pin is connected to Peripheral signal RGMII0.TXD2.	`RW`	`0x0`

EMACIO4

This register is used to control the peripherals connected to emac0_tx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08410`

Offset: `0x410`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td colspan="2">sel
RW 0x0</td></tr>
</table>

EMACIO4 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac0_tx_d3. 0 : Pin is connected to GPIO/LoanIO number 4. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D3. 3 : Pin is connected to Peripheral signal RGMII0.TXD3.	`RW`	`0x0`

EMACIO5

This register is used to control the peripherals connected to emac0_rx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08414`

Offset: `0x414`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO5 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac0_rx_d0. 0 : Pin is connected to GPIO/LoanIO number 5. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D4. 3 : Pin is connected to Peripheral signal RGMII0.RXD0.	RW	0x0

EMACIO6

This register is used to control the peripherals connected to emac0_mdio Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08418`

Offset: `0x418`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO6 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac0_mdio. 0 : Pin is connected to GPIO/LoanIO number 6. 1 : Pin is connected to Peripheral signal I2C2.SDA. 2 : Pin is connected to Peripheral signal USB1.D5. 3 : Pin is connected to Peripheral signal RGMII0.MDIO.	RW	0x0

EMACIO7

This register is used to control the peripherals connected to emac0_mdc Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0841C

Offset: 0x41C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_mdc. 0 : Pin is connected to GPIO/LoanIO number 7. 1 : Pin is connected to Peripheral signal I2C2.SCL. 2 : Pin is connected to Peripheral signal USB1.D6. 3 : Pin is connected to Peripheral signal RGMII0.MDC.	RW	0x0

EMACIO8

This register is used to control the peripherals connected to emac0_rx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08420

Offset: 0x420

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO8 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_rx_ctl. 0 : Pin is connected to GPIO/LoanIO number 8. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.D7. 3 : Pin is connected to Peripheral signal RGMII0.RX_CTL.	RW	0x0

EMACIO9

This register is used to control the peripherals connected to emac0_tx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08424

Offset: 0x424

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO9 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_tx_ctl. 0 : Pin is connected to GPIO/LoanIO number 9. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII0.TX_CTL.	RW	0x0

EMACIO10

This register is used to control the peripherals connected to emac0_rx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08428

Offset: 0x428

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO10 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_rx_clk. 0 : Pin is connected to GPIO/LoanIO number 10. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.CLK. 3 : Pin is connected to Peripheral signal RGMII0.RX_CLK.	RW	0x0

EMACIO11

This register is used to control the peripherals connected to emac0_rx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0842C

Offset: 0x42C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO11 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_rx_d1. 0 : Pin is connected to GPIO/LoanIO number 11. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.STP. 3 : Pin is connected to Peripheral signal RGMII0.RXD1.	RW	0x0

EMACIO12

This register is used to control the peripherals connected to emac0_rx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08430

Offset: 0x430

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO12 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac0_rx_d2. 0 : Pin is connected to GPIO/LoanIO number 12. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.DIR. 3 : Pin is connected to Peripheral signal RGMII0.RXD2.	RW	0x0

EMACIO13

This register is used to control the peripherals connected to emac0_rx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08434

Offset: 0x434

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO13 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac0_rx_d3. 0 : Pin is connected to GPIO/LoanIO number 13. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB1.NXT. 3 : Pin is connected to Peripheral signal RGMII0.RXD3.	`RW`	`0x0`

EMACIO14

This register is used to control the peripherals connected to emac1_tx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08438`

Offset: `0x438`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

EMACIO14 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac1_tx_clk. 0 : Pin is connected to GPIO/LoanIO number 48. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.TX_CLK.	`RW`	`0x0`

EMACIO15

This register is used to control the peripherals connected to emac1_tx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0843C`

Offset: `0x43C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO15 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_tx_d0. 0 : Pin is connected to GPIO/LoanIO number 49. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.TXD0.	RW	0x0

EMACIO16

This register is used to control the peripherals connected to emac1_tx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08440

Offset: 0x440

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO16 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_tx_d1. 0 : Pin is connected to GPIO/LoanIO number 50. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.TXD1.	RW	0x0

EMACIO17

This register is used to control the peripherals connected to emac1_tx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08444

Offset: 0x444

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO17 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_tx_ctl. 0 : Pin is connected to GPIO/LoanIO number 51. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.TX_CTL.	RW	0x0

EMACIO18

This register is used to control the peripherals connected to emac1_rx_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08448

Offset: 0x448

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO18 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_rx_d0. 0 : Pin is connected to GPIO/LoanIO number 52. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.RXD0.	RW	0x0

EMACIO19

This register is used to control the peripherals connected to emac1_rx_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0844C

Offset: 0x44C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

EMACIO19 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_rx_d1. 0 : Pin is connected to GPIO/LoanIO number 53. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal RGMII1.RXD1.	RW	0x0

FLASHIO0

This register is used to control the peripherals connected to sdmmc_cmd Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08450

Offset: 0x450

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO0 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_cmd. 0 : Pin is connected to GPIO/LoanIO number 36. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D0. 3 : Pin is connected to Peripheral signal SDMMC.CMD.	RW	0x0

FLASHIO1

This register is used to control the peripherals connected to sdmmc_pwren Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08454

Offset: 0x454

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO1 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_pwren. 0 : Pin is connected to GPIO/LoanIO number 37. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D1. 3 : Pin is connected to Peripheral signal SDMMC.PWREN.	RW	0x0

FLASHIO2

This register is used to control the peripherals connected to sdmmc_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08458`

Offset: `0x458`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

FLASHIO2 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected sdmmc_d0. 0 : Pin is connected to GPIO/LoanIO number 38. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D2. 3 : Pin is connected to Peripheral signal SDMMC.D0.	`RW`	`0x0`

FLASHIO3

This register is used to control the peripherals connected to sdmmc_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0845C`

Offset: `0x45C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

FLASHIO3 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d1. 0 : Pin is connected to GPIO/LoanIO number 39. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D3. 3 : Pin is connected to Peripheral signal SDMMC.D1.	RW	0x0

FLASHIO4

This register is used to control the peripherals connected to sdmmc_d4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08460

Offset: 0x460

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td colspan="2">sel
RW 0x0</td></tr>
</table>

FLASHIO4 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d4. 0 : Pin is connected to GPIO/LoanIO number 40. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D4. 3 : Pin is connected to Peripheral signal SDMMC.D4.	RW	0x0

FLASHIO5

This register is used to control the peripherals connected to sdmmc_d5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08464

Offset: 0x464

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO5 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d5. 0 : Pin is connected to GPIO/LoanIO number 41. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D5. 3 : Pin is connected to Peripheral signal SDMMC.D5.	RW	0x0

FLASHIO6

This register is used to control the peripherals connected to sdmmc_d6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08468

Offset: 0x468

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO6 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d6. 0 : Pin is connected to GPIO/LoanIO number 42. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D6. 3 : Pin is connected to Peripheral signal SDMMC.D6.	RW	0x0

FLASHIO7

This register is used to control the peripherals connected to sdmmc_d7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0846C

Offset: 0x46C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d7. 0 : Pin is connected to GPIO/LoanIO number 43. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.D7. 3 : Pin is connected to Peripheral signal SDMMC.D7.	RW	0x0

FLASHIO8

This register is used to control the peripherals connected to sdmmc_clk_in Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08470

Offset: 0x470

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO8 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_clk_in. 0 : Pin is connected to GPIO/LoanIO number 44. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.CLK. 3 : Pin is connected to Peripheral signal SDMMC.CLK_IN.	RW	0x0

FLASHIO9

This register is used to control the peripherals connected to sdmmc_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08474

Offset: 0x474

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO9 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_clk. 0 : Pin is connected to GPIO/LoanIO number 45. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.STP. 3 : Pin is connected to Peripheral signal SDMMC.CLK.	RW	0x0

FLASHIO10

This register is used to control the peripherals connected to sdmmc_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08478

Offset: 0x478

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO10 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d2. 0 : Pin is connected to GPIO/LoanIO number 46. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.DIR. 3 : Pin is connected to Peripheral signal SDMMC.D2.	RW	0x0

FLASHIO11

This register is used to control the peripherals connected to sdmmc_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0847C

Offset: 0x47C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

FLASHIO11 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected sdmmc_d3. 0 : Pin is connected to GPIO/LoanIO number 47. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal USB0.NXT. 3 : Pin is connected to Peripheral signal SDMMC.D3.	RW	0x0

GENERALIO0

This register is used to control the peripherals connected to trace_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08480

Offset: 0x480

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO0 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_clk. 0 : Pin is connected to GPIO/LoanIO number 48. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal TRACE.CLK.	RW	0x0

GENERALIO1

This register is used to control the peripherals connected to trace_d0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08484

Offset: 0x484

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO1 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected trace_d0. 0 : Pin is connected to GPIO/LoanIO number 49. 1 : Pin is connected to Peripheral signal UART0.RX. 2 : Pin is connected to Peripheral signal SPIS0.CLK. 3 : Pin is connected to Peripheral signal TRACE.D0.	`RW`	`0x0`

GENERALIO2

This register is used to control the peripherals connected to trace_d1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08488`

Offset: `0x488`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

GENERALIO2 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected trace_d1. 0 : Pin is connected to GPIO/LoanIO number 50. 1 : Pin is connected to Peripheral signal UART0.TX. 2 : Pin is connected to Peripheral signal SPIS0.MOSI. 3 : Pin is connected to Peripheral signal TRACE.D1.	`RW`	`0x0`

GENERALIO3

This register is used to control the peripherals connected to trace_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0848C`

Offset: `0x48C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO3 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d2. 0 : Pin is connected to GPIO/LoanIO number 51. 1 : Pin is connected to Peripheral signal I2C1.SDA. 2 : Pin is connected to Peripheral signal SPIS0.MISO. 3 : Pin is connected to Peripheral signal TRACE.D2.	RW	0x0

GENERALIO4

This register is used to control the peripherals connected to trace_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08490

Offset: 0x490

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO4 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d3. 0 : Pin is connected to GPIO/LoanIO number 52. 1 : Pin is connected to Peripheral signal I2C1.SCL. 2 : Pin is connected to Peripheral signal SPIS0.SS0. 3 : Pin is connected to Peripheral signal TRACE.D3.	RW	0x0

GENERALIO5

This register is used to control the peripherals connected to trace_d4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08494

Offset: 0x494

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO5 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d4. 0 : Pin is connected to GPIO/LoanIO number 53. 1 : Pin is connected to Peripheral signal CAN1.RX. 2 : Pin is connected to Peripheral signal SPIS1.CLK. 3 : Pin is connected to Peripheral signal TRACE.D4.	RW	0x0

GENERALIO6

This register is used to control the peripherals connected to trace_d5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08498

Offset: 0x498

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO6 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d5. 0 : Pin is connected to GPIO/LoanIO number 54. 1 : Pin is connected to Peripheral signal CAN1.TX. 2 : Pin is connected to Peripheral signal SPIS1.MOSI. 3 : Pin is connected to Peripheral signal TRACE.D5.	RW	0x0

GENERALIO7

This register is used to control the peripherals connected to trace_d6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0849C

Offset: 0x49C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d6. 0 : Pin is connected to GPIO/LoanIO number 55. 1 : Pin is connected to Peripheral signal I2C0.SDA. 2 : Pin is connected to Peripheral signal SPIS1.SS0. 3 : Pin is connected to Peripheral signal TRACE.D6.	RW	0x0

GENERALIO8

This register is used to control the peripherals connected to trace_d7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084A0

Offset: 0x4A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO8 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected trace_d7. 0 : Pin is connected to GPIO/LoanIO number 56. 1 : Pin is connected to Peripheral signal I2C0.SCL. 2 : Pin is connected to Peripheral signal SPIS1.MISO. 3 : Pin is connected to Peripheral signal TRACE.D7.	RW	0x0

GENERALIO9

This register is used to control the peripherals connected to spim0_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084A4

Offset: 0x4A4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO9 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spim0_clk. 0 : Pin is connected to GPIO/LoanIO number 57. 1 : Pin is connected to Peripheral signal UART0.CTS. 2 : Pin is connected to Peripheral signal I2C1.SDA. 3 : Pin is connected to Peripheral signal SPIM0.CLK.	RW	0x0

GENERALIO10

This register is used to control the peripherals connected to spim0_mosi Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084A8

Offset: 0x4A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO10 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spim0_mosi. 0 : Pin is connected to GPIO/LoanIO number 58. 1 : Pin is connected to Peripheral signal UART0.RTS. 2 : Pin is connected to Peripheral signal I2C1.SCL. 3 : Pin is connected to Peripheral signal SPIM0.MOSI.	RW	0x0

GENERALIO11

This register is used to control the peripherals connected to spim0_miso Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084AC

Offset: 0x4AC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO11 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spim0_miso. 0 : Pin is connected to GPIO/LoanIO number 59. 1 : Pin is connected to Peripheral signal UART1.CTS. 2 : Pin is connected to Peripheral signal CAN1.RX. 3 : Pin is connected to Peripheral signal SPIM0.MISO.	RW	0x0

GENERALIO12

This register is used to control the peripherals connected to spim0_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084B0

Offset: 0x4B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO12 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spim0_ss0. 0 : Pin is connected to GPIO/LoanIO number 60. 1 : Pin is connected to Peripheral signal UART1.RTS. 2 : Pin is connected to Peripheral signal CAN1.TX. 3 : Pin is connected to Peripheral signal SPIM0.SS0.	RW	0x0

GENERALIO13

This register is used to control the peripherals connected to uart0_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084B4

Offset: 0x4B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GENERALIO13 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected uart0_rx. 0 : Pin is connected to GPIO/LoanIO number 61. 1 : Pin is connected to Peripheral signal SPIM0.SS1. 2 : Pin is connected to Peripheral signal CAN0.RX. 3 : Pin is connected to Peripheral signal UART0.RX.	RW	0x0

GENERALIO14

This register is used to control the peripherals connected to uart0_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084B8

Offset: 0x4B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GENERALIO14 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected uart0_tx. 0 : Pin is connected to GPIO/LoanIO number 62. 1 : Pin is connected to Peripheral signal SPIM1.SS1. 2 : Pin is connected to Peripheral signal CAN0.TX. 3 : Pin is connected to Peripheral signal UART0.TX.	RW	0x0

GENERALIO15

This register is used to control the peripherals connected to i2c0_sda Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084BC`

Offset: `0x4BC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

GENERALIO15 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected i2c0_sda. 0 : Pin is connected to GPIO/LoanIO number 63. 1 : Pin is connected to Peripheral signal SPIM1.CLK. 2 : Pin is connected to Peripheral signal UART1.RX. 3 : Pin is connected to Peripheral signal I2C0.SDA.	`RW`	`0x0`

GENERALIO16

This register is used to control the peripherals connected to i2c0_scl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084C0`

Offset: `0x4C0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

GENERALIO16 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected i2c0_scl. 0 : Pin is connected to GPIO/LoanIO number 64. 1 : Pin is connected to Peripheral signal SPIM1.MOSI. 2 : Pin is connected to Peripheral signal UART1.TX. 3 : Pin is connected to Peripheral signal I2C0.SCL.	RW	0x0

GENERALIO17

This register is used to control the peripherals connected to can0_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084C4`

Offset: `0x4C4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO17 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected can0_rx. 0 : Pin is connected to GPIO/LoanIO number 65. 1 : Pin is connected to Peripheral signal SPIM1.MISO. 2 : Pin is connected to Peripheral signal UART0.RX. 3 : Pin is connected to Peripheral signal CAN0.RX.	RW	0x0

GENERALIO18

This register is used to control the peripherals connected to can0_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084C8`

Offset: `0x4C8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO18 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected can0_tx. 0 : Pin is connected to GPIO/LoanIO number 66. 1 : Pin is connected to Peripheral signal SPIM1.SS0. 2 : Pin is connected to Peripheral signal UART0.TX. 3 : Pin is connected to Peripheral signal CAN0.TX.	RW	0x0

GENERALIO19

This register is used to control the peripherals connected to spis1_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084CC`

Offset: `0x4CC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO19 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected spis1_clk. 0 : Pin is connected to GPIO/LoanIO number 67. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM1.CLK. 3 : Pin is connected to Peripheral signal SPIS1.CLK.	RW	0x0

GENERALIO20

This register is used to control the peripherals connected to spis1_mosi Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084D0

Offset: 0x4D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO20 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spis1_mosi. 0 : Pin is connected to GPIO/LoanIO number 68. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM1.MOSI. 3 : Pin is connected to Peripheral signal SPIS1.MOSI.	RW	0x0

GENERALIO21

This register is used to control the peripherals connected to spis1_miso Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084D4

Offset: 0x4D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO21 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spis1_miso. 0 : Pin is connected to GPIO/LoanIO number 69. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM1.MISO. 3 : Pin is connected to Peripheral signal SPIS1.MISO.	RW	0x0

GENERALIO22

This register is used to control the peripherals connected to spis1_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084D8

Offset: 0x4D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO22 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected spis1_ss0. 0 : Pin is connected to GPIO/LoanIO number 70. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM1.SS0. 3 : Pin is connected to Peripheral signal SPIS1.SS0.	RW	0x0

GENERALIO23

This register is used to control the peripherals connected to uart1_rx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084DC

Offset: 0x4DC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO23 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected uart1_rx. 0 : Pin is connected to GPIO/LoanIO number 62. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM1.SS1. 3 : Pin is connected to Peripheral signal UART1.RX.	RW	0x0

GENERALIO24

This register is used to control the peripherals connected to uart1_tx Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084E0

Offset: 0x4E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO24 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected uart1_tx. 0 : Pin is connected to GPIO/LoanIO number 63. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM0.CLK. 3 : Pin is connected to Peripheral signal UART1.TX.	RW	0x0

GENERALIO25

This register is used to control the peripherals connected to i2c1_sda Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084E4

Offset: 0x4E4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO25 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected i2c1_sda. 0 : Pin is connected to GPIO/LoanIO number 64. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM0.MOSI. 3 : Pin is connected to Peripheral signal I2C1.SDA.	RW	0x0

GENERALIO26

This register is used to control the peripherals connected to i2c1_scl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD084E8

Offset: 0x4E8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

GENERALIO26 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected i2c1_scl. 0 : Pin is connected to GPIO/LoanIO number 65. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM0.MISO. 3 : Pin is connected to Peripheral signal I2C1.SCL.	`RW`	`0x0`

GENERALIO27

This register is used to control the peripherals connected to spim0_ss0_alt Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD084EC`

Offset: `0x4EC`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td colspan="2">sel
RW 0x0</td></tr>
</table>

GENERALIO27 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected spim0_ss0_alt. 0 : Pin is connected to GPIO/LoanIO number 66. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal SPIM0.SS0. 3 : Pin is connected to Peripheral signal not applicable.	`RW`	`0x0`

MIXED1IO0

This register is used to control the peripherals connected to nand_ale Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08500`

Offset: `0x500`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO0 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_ale. 0 : Pin is connected to GPIO/LoanIO number 14. 1 : Pin is connected to Peripheral signal QSPI.SS3. 2 : Pin is connected to Peripheral signal RGMII1.TX_CLK. 3 : Pin is connected to Peripheral signal NAND.ale.	RW	0x0

MIXED1IO1

This register is used to control the peripherals connected to nand_ce Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08504

Offset: 0x504

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO1 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_ce. 0 : Pin is connected to GPIO/LoanIO number 15. 1 : Pin is connected to Peripheral signal USB1.D0. 2 : Pin is connected to Peripheral signal RGMII1.TXD0. 3 : Pin is connected to Peripheral signal NAND.ce.	RW	0x0

MIXED1IO2

This register is used to control the peripherals connected to nand_cle Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08508

Offset: 0x508

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO2 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_cle. 0 : Pin is connected to GPIO/LoanIO number 16. 1 : Pin is connected to Peripheral signal USB1.D1. 2 : Pin is connected to Peripheral signal RGMII1.TXD1. 3 : Pin is connected to Peripheral signal NAND.cle.	RW	0x0

MIXED1IO3

This register is used to control the peripherals connected to nand_re Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0850C

Offset: 0x50C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO3 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_re. 0 : Pin is connected to GPIO/LoanIO number 17. 1 : Pin is connected to Peripheral signal USB1.D2. 2 : Pin is connected to Peripheral signal RGMII1.TXD2. 3 : Pin is connected to Peripheral signal NAND.re.	RW	0x0

MIXED1IO4

This register is used to control the peripherals connected to nand_rb Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08510

Offset: 0x510

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO4 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_rb. 0 : Pin is connected to GPIO/LoanIO number 18. 1 : Pin is connected to Peripheral signal USB1.D3. 2 : Pin is connected to Peripheral signal RGMII1.TXD3. 3 : Pin is connected to Peripheral signal NAND.rb.	RW	0x0

MIXED1IO5

This register is used to control the peripherals connected to nand_dq0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08514

Offset: 0x514

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO5 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq0. 0 : Pin is connected to GPIO/LoanIO number 19. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal RGMII1.RXD0. 3 : Pin is connected to Peripheral signal NAND.dq0.	RW	0x0

MIXED1IO6

This register is used to control the peripherals connected to nand_dq1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08518

Offset: 0x518

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO6 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq1. 0 : Pin is connected to GPIO/LoanIO number 20. 1 : Pin is connected to Peripheral signal I2C3.SDA. 2 : Pin is connected to Peripheral signal RGMII1.MDIO. 3 : Pin is connected to Peripheral signal NAND.dq1.	RW	0x0

MIXED1IO7

This register is used to control the peripherals connected to nand_dq2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0851C

Offset: 0x51C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq2. 0 : Pin is connected to GPIO/LoanIO number 21. 1 : Pin is connected to Peripheral signal I2C3.SCL. 2 : Pin is connected to Peripheral signal RGMII1.MDC. 3 : Pin is connected to Peripheral signal NAND.dq2.	RW	0x0

MIXED1IO8

This register is used to control the peripherals connected to nand_dq3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08520

Offset: 0x520

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO8 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq3. 0 : Pin is connected to GPIO/LoanIO number 22. 1 : Pin is connected to Peripheral signal USB1.D4. 2 : Pin is connected to Peripheral signal RGMII1.RX_CTL. 3 : Pin is connected to Peripheral signal NAND.dq3.	RW	0x0

MIXED1IO9

This register is used to control the peripherals connected to nand_dq4 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08524

Offset: 0x524

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO9 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq4. 0 : Pin is connected to GPIO/LoanIO number 23. 1 : Pin is connected to Peripheral signal USB1.D5. 2 : Pin is connected to Peripheral signal RGMII1.TX_CTL. 3 : Pin is connected to Peripheral signal NAND.dq4.	RW	0x0

MIXED1IO10

This register is used to control the peripherals connected to nand_dq5 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08528

Offset: 0x528

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO10 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq5. 0 : Pin is connected to GPIO/LoanIO number 24. 1 : Pin is connected to Peripheral signal USB1.D6. 2 : Pin is connected to Peripheral signal RGMII1.RX_CLK. 3 : Pin is connected to Peripheral signal NAND.dq5.	RW	0x0

MIXED1IO11

This register is used to control the peripherals connected to nand_dq6 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0852C

Offset: 0x52C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO11 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected nand_dq6. 0 : Pin is connected to GPIO/LoanIO number 25. 1 : Pin is connected to Peripheral signal USB1.D7. 2 : Pin is connected to Peripheral signal RGMII1.RXD1. 3 : Pin is connected to Peripheral signal NAND.dq6.	RW	0x0

MIXED1IO12

This register is used to control the peripherals connected to nand_dq7 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08530`

Offset: `0x530`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO12 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected nand_dq7. 0 : Pin is connected to GPIO/LoanIO number 26. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal RGMII1.RXD2. 3 : Pin is connected to Peripheral signal NAND.dq7.	`RW`	`0x0`

MIXED1IO13

This register is used to control the peripherals connected to nand_wp Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08534`

Offset: `0x534`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO13 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected nand_wp. 0 : Pin is connected to GPIO/LoanIO number 27. 1 : Pin is connected to Peripheral signal QSPI.SS2. 2 : Pin is connected to Peripheral signal RGMII1.RXD3. 3 : Pin is connected to Peripheral signal NAND.wp.	`RW`	`0x0`

MIXED1IO14

This register is used to control the peripherals connected to nand_we Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08538`

Offset: `0x538`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO14 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected nand_we. 0 : Pin is connected to GPIO/LoanIO number 28. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal QSPI.SS1. 3 : Pin is connected to Peripheral signal NAND.we.	`RW`	`0x0`

MIXED1IO15

This register is used to control the peripherals connected to qspi_io0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0853C`

Offset: `0x53C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO15 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_io0. 0 : Pin is connected to GPIO/LoanIO number 29. 1 : Pin is connected to Peripheral signal USB1.CLK. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.IO0.	RW	0x0

MIXED1IO16

This register is used to control the peripherals connected to qspi_io1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08540

Offset: 0x540

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO16 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_io1. 0 : Pin is connected to GPIO/LoanIO number 30. 1 : Pin is connected to Peripheral signal USB1.STP. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.IO1.	RW	0x0

MIXED1IO17

This register is used to control the peripherals connected to qspi_io2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08544

Offset: 0x544

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO17 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_io2. 0 : Pin is connected to GPIO/LoanIO number 31. 1 : Pin is connected to Peripheral signal USB1.DIR. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.IO2.	RW	0x0

MIXED1IO18

This register is used to control the peripherals connected to qspi_io3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08548

Offset: 0x548

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO18 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_io3. 0 : Pin is connected to GPIO/LoanIO number 32. 1 : Pin is connected to Peripheral signal USB1.NXT. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.IO3.	RW	0x0

MIXED1IO19

This register is used to control the peripherals connected to qspi_ss0 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0854C

Offset: 0x54C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td colspan="2">sel
RW 0x0</td></tr>
</table>

MIXED1IO19 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_ss0. 0 : Pin is connected to GPIO/LoanIO number 33. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.SS0.	RW	0x0

MIXED1IO20

This register is used to control the peripherals connected to qpsi_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08550

Offset: 0x550

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO20 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qpsi_clk. 0 : Pin is connected to GPIO/LoanIO number 34. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.CLK.	RW	0x0

MIXED1IO21

This register is used to control the peripherals connected to qspi_ss1 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08554

Offset: 0x554

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED1IO21 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected qspi_ss1. 0 : Pin is connected to GPIO/LoanIO number 35. 1 : Pin is connected to Peripheral signal not applicable. 2 : Pin is connected to Peripheral signal not applicable. 3 : Pin is connected to Peripheral signal QSPI.SS1.	RW	0x0

MIXED2IO0

This register is used to control the peripherals connected to emac1_mdio Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08558

Offset: 0x558

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO0 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_mdio. 0 : Pin is connected to GPIO/LoanIO number 54. 1 : Pin is connected to Peripheral signal SPIS0.CLK. 2 : Pin is connected to Peripheral signal SPIM0.CLK. 3 : Pin is connected to Peripheral signal RGMII1.MDIO.	RW	0x0

MIXED2IO1

This register is used to control the peripherals connected to emac1_mdc Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0855C

Offset: 0x55C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO1 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac1_mdc. 0 : Pin is connected to GPIO/LoanIO number 55. 1 : Pin is connected to Peripheral signal SPIS0.MOSI. 2 : Pin is connected to Peripheral signal SPIM0.MOSI. 3 : Pin is connected to Peripheral signal RGMII1.MDC.	`RW`	`0x0`

MIXED2IO2

This register is used to control the peripherals connected to emac1_tx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08560`

Offset: `0x560`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`sel` `RW 0x0`	

MIXED2IO2 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac1_tx_d2. 0 : Pin is connected to GPIO/LoanIO number 56. 1 : Pin is connected to Peripheral signal SPIS0.MISO. 2 : Pin is connected to Peripheral signal SPIM0.MISO. 3 : Pin is connected to Peripheral signal RGMII1.TXD2.	`RW`	`0x0`

MIXED2IO3

This register is used to control the peripherals connected to emac1_tx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08564`

Offset: `0x564`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO3 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac1_tx_d3. 0 : Pin is connected to GPIO/LoanIO number 57. 1 : Pin is connected to Peripheral signal SPIS0.SS0. 2 : Pin is connected to Peripheral signal SPIM0.SS0. 3 : Pin is connected to Peripheral signal RGMII1.TXD3.	RW	0x0

MIXED2IO4

This register is used to control the peripherals connected to emac1_rx_clk Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08568`

Offset: `0x568`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO4 Fields

Bit	Name	Description	Access	Reset
1:0	`sel`	Select peripheral signals connected emac1_rx_clk. 0 : Pin is connected to GPIO/LoanIO number 58. 1 : Pin is connected to Peripheral signal SPIM1.CLK. 2 : Pin is connected to Peripheral signal SPIS1.CLK. 3 : Pin is connected to Peripheral signal RGMII1.RX_CLK.	RW	0x0

MIXED2IO5

This register is used to control the peripherals connected to emac1_rx_ctl Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0856C

Offset: 0x56C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO5 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_rx_ctl. 0 : Pin is connected to GPIO/LoanIO number 59. 1 : Pin is connected to Peripheral signal SPIM1.MOSI. 2 : Pin is connected to Peripheral signal SPIS1.MOSI. 3 : Pin is connected to Peripheral signal RGMII1.RX_CTL.	RW	0x0

MIXED2IO6

This register is used to control the peripherals connected to emac1_rx_d2 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08570

Offset: 0x570

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO6 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_rx_d2. 0 : Pin is connected to GPIO/LoanIO number 60. 1 : Pin is connected to Peripheral signal SPIM1.MISO. 2 : Pin is connected to Peripheral signal SPIS1.MISO. 3 : Pin is connected to Peripheral signal RGMII1.RXD2.	RW	0x0

MIXED2IO7

This register is used to control the peripherals connected to emac1_rx_d3 Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08574

Offset: 0x574

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														sel RW 0x0	

MIXED2IO7 Fields

Bit	Name	Description	Access	Reset
1:0	sel	Select peripheral signals connected emac1_rx_d3. 0 : Pin is connected to GPIO/LoanIO number 61. 1 : Pin is connected to Peripheral signal SPIM1.SS0. 2 : Pin is connected to Peripheral signal SPIS1.SS0. 3 : Pin is connected to Peripheral signal RGMII1.RXD3.	RW	0x0

GPLINMUX48

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 48. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08578

Offset: 0x578

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX48 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 48. 0 : Source for GPIO/LoanIO 48 is GENERALIO0. 1 : Source for GPIO/LoanIO 48 is EMACIO14.	RW	0x0

GPLINMUX49

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 49. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0857C

Offset: `0x57C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX49 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 49. 0 : Source for GPIO/LoanIO 49 is GENERALIO1. 1 : Source for GPIO/LoanIO 49 is EMACIO15.	RW	0x0

GPLINMUX50

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 50. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08580

Offset: `0x580`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX50 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 50. 0 : Source for GPIO/LoanIO 50 is GENERALIO2. 1 : Source for GPIO/LoanIO 50 is EMACIO16.	RW	0x0

GPLINMUX51

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 51. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08584

Offset: `0x584`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX51 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 51. 0 : Source for GPIO/LoanIO 51 is GENERALIO3. 1 : Source for GPIO/LoanIO 51 is EMACIO17.	RW	0x0

GPLINMUX52

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 52. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08588

Offset: 0x588

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX52 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 52. 0 : Source for GPIO/LoanIO 52 is GENERALIO4. 1 : Source for GPIO/LoanIO 52 is EMACIO18.	RW	0x0

GPLINMUX53

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 53. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0858C

Offset: 0x58C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX53 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 53. 0 : Source for GPIO/LoanIO 53 is GENERALIO5. 1 : Source for GPIO/LoanIO 53 is EMACIO19.	RW	0x0

GPLINMUX54

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 54. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08590`

Offset: `0x590`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`sel` `RW 0x0`

GPLINMUX54 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 54. 0 : Source for GPIO/LoanIO 54 is GENERALIO6. 1 : Source for GPIO/LoanIO 54 is MIXED2IO0.	`RW`	`0x0`

GPLINMUX55

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 55. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08594`

Offset: `0x594`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`sel` `RW 0x0`

GPLINMUX55 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 55. 0 : Source for GPIO/LoanIO 55 is GENERALIO7. 1 : Source for GPIO/LoanIO 55 is MIXED2IO1.	`RW`	`0x0`

GPLINMUX56

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 56. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08598`

Offset: `0x598`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`sel` `RW 0x0`

GPLINMUX56 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 56. 0 : Source for GPIO/LoanIO 56 is GENERALIO8. 1 : Source for GPIO/LoanIO 56 is MIXED2IO2.	`RW`	`0x0`

GPLINMUX57

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 57. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0859C`

Offset: `0x59C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX57 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 57. 0 : Source for GPIO/LoanIO 57 is GENERALIO9. 1 : Source for GPIO/LoanIO 57 is MIXED2IO3.	RW	0x0

GPLINMUX58

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 58. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085A0

Offset: 0x5A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX58 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 58. 0 : Source for GPIO/LoanIO 58 is GENERALIO10. 1 : Source for GPIO/LoanIO 58 is MIXED2IO4.	RW	0x0

GPLINMUX59

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 59. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085A4

Offset: 0x5A4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX59 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 59. 0 : Source for GPIO/LoanIO 59 is GENERALIO11. 1 : Source for GPIO/LoanIO 59 is MIXED2IO5.	RW	0x0

GPLINMUX60

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 60. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085A8

Offset: 0x5A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX60 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 60. 0 : Source for GPIO/LoanIO 60 is GENERALIO12. 1 : Source for GPIO/LoanIO 60 is MIXED2IO6.	RW	0x0

GPLINMUX61

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 61. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085AC

Offset: 0x5AC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX61 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 61. 0 : Source for GPIO/LoanIO 61 is GENERALIO13. 1 : Source for GPIO/LoanIO 61 is MIXED2IO7.	RW	0x0

GPLINMUX62

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/ LoanIO 62. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085B0

Offset: 0x5B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX62 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 62. 0 : Source for GPIO/LoanIO 62 is GENERALIO14. 1 : Source for GPIO/LoanIO 62 is GENERALIO23.	RW	0x0

GPLINMUX63

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 63. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085B4

Offset: 0x5B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX63 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 63. 0 : Source for GPIO/LoanIO 63 is GENERALIO15. 1 : Source for GPIO/LoanIO 63 is GENERALIO24.	RW	0x0

GPLINMUX64

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 64. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085B8

Offset: 0x5B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX64 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 64. 0 : Source for GPIO/LoanIO 64 is GENERALIO16. 1 : Source for GPIO/LoanIO 64 is GENERALIO25.	RW	0x0

GPLINMUX65

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 65. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085BC

Offset: `0x5BC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX65 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 65. 0 : Source for GPIO/LoanIO 65 is GENERALIO17. 1 : Source for GPIO/LoanIO 65 is GENERALIO26.	RW	0x0

GPLINMUX66

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 66. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085C0

Offset: `0x5C0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX66 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 66. 0 : Source for GPIO/LoanIO 66 is GENERALIO18. 1 : Source for GPIO/LoanIO 66 is GENERALIO27.	RW	0x0

GPLINMUX67

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 67. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085C4

Offset: 0x5C4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX67 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 67. 0 : Source for GPIO/LoanIO 67 is GENERALIO19. 1 : Source for GPIO/LoanIO 67 is GENERALIO28.	RW	0x0

GPLINMUX68

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 68. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085C8

Offset: `0x5C8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX68 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 68. 0 : Source for GPIO/LoanIO 68 is GENERALIO20. 1 : Source for GPIO/LoanIO 68 is GENERALIO29.	RW	0x0

GPLINMUX69

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 69. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD085CC`

Offset: `0x5CC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX69 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 69. 0 : Source for GPIO/LoanIO 69 is GENERALIO21. 1 : Source for GPIO/LoanIO 69 is GENERALIO30.	RW	0x0

GPLINMUX70

Some GPIO/LoanIO inputs can be driven by multiple pins. This register selects the input signal for GPIO/LoanIO 70. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085D0

Offset: 0x5D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLINMUX70 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 70. 0 : Source for GPIO/LoanIO 70 is GENERALIO22. 1 : Source for GPIO/LoanIO 70 is GENERALIO31.	RW	0x0

GPLMUX0

Selection between GPIO and LoanIO output and output enable for GPIO0 and LoanIO0. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085D4

Offset: 0x5D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX0 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 0. 0 : LoanIO 0 controls GPIO/LOANIO[0] output and output enable signals. 1 : GPIO 0 controls GPIO/LOANI[0] output and output enable signals.	RW	0x0

GPLMUX1

Selection between GPIO and LoanIO output and output enable for GPIO1 and LoanIO1. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085D8

Offset: 0x5D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX1 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 1. 0 : LoanIO 1 controls GPIO/LOANIO[1] output and output enable signals. 1 : GPIO 1 controls GPIO/LOANI[1] output and output enable signals.	RW	0x0

GPLMUX2

Selection between GPIO and LoanIO output and output enable for GPIO2 and LoanIO2. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085DC

Offset: 0x5DC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX2 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 2. 0 : LoanIO 2 controls GPIO/LOANIO[2] output and output enable signals. 1 : GPIO 2 controls GPIO/LOANI[2] output and output enable signals.	RW	0x0

GPLMUX3

Selection between GPIO and LoanIO output and output enable for GPIO3 and LoanIO3. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085E0

Offset: `0x5E0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX3 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 3. 0 : LoanIO 3 controls GPIO/LOANIO[3] output and output enable signals. 1 : GPIO 3 controls GPIO/LOANI[3] output and output enable signals.	RW	0x0

GPLMUX4

Selection between GPIO and LoanIO output and output enable for GPIO4 and LoanIO4. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085E4

Offset: `0x5E4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX4 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 4. 0 : LoanIO 4 controls GPIO/LOANIO[4] output and output enable signals. 1 : GPIO 4 controls GPIO/LOANI[4] output and output enable signals.	RW	0x0

GPLMUX5

Selection between GPIO and LoanIO output and output enable for GPIO5 and LoanIO5. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085E8

Offset: `0x5E8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX5 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 5. 0 : LoanIO 5 controls GPIO/LOANIO[5] output and output enable signals. 1 : GPIO 5 controls GPIO/LOANI[5] output and output enable signals.	RW	0x0

GPLMUX6

Selection between GPIO and LoanIO output and output enable for GPIO6 and LoanIO6. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085EC

Offset: 0x5EC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX6 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 6. 0 : LoanIO 6 controls GPIO/LOANIO[6] output and output enable signals. 1 : GPIO 6 controls GPIO/LOANI[6] output and output enable signals.	RW	0x0

GPLMUX7

Selection between GPIO and LoanIO output and output enable for GPIO7 and LoanIO7. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085F0

Offset: 0x5F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX7 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 7. 0 : LoanIO 7 controls GPIO/LOANIO[7] output and output enable signals. 1 : GPIO 7 controls GPIO/LOANI[7] output and output enable signals.	RW	0x0

GPLMUX8

Selection between GPIO and LoanIO output and output enable for GPIO8 and LoanIO8. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085F4

Offset: 0x5F4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX8 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 8. 0 : LoanIO 8 controls GPIO/LOANIO[8] output and output enable signals. 1 : GPIO 8 controls GPIO/LOANI[8] output and output enable signals.	RW	0x0

GPLMUX9

Selection between GPIO and LoanIO output and output enable for GPIO9 and LoanIO9. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085F8

Offset: 0x5F8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX9 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 9. 0 : LoanIO 9 controls GPIO/LOANIO[9] output and output enable signals. 1 : GPIO 9 controls GPIO/LOANI[9] output and output enable signals.	RW	0x0

GPLMUX10

Selection between GPIO and LoanIO output and output enable for GPIO10 and LoanIO10. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD085FC

Offset: 0x5FC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX10 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 10. 0 : LoanIO 10 controls GPIO/LOANIO[10] output and output enable signals. 1 : GPIO 10 controls GPIO/LOANI[10] output and output enable signals.	RW	0x0

GPLMUX11

Selection between GPIO and LoanIO output and output enable for GPIO11 and LoanIO11. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08600

Offset: 0x600

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX11 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 11. 0 : LoanIO 11 controls GPIO/LOANIO[11] output and output enable signals. 1 : GPIO 11 controls GPIO/ LOANI[11] output and output enable signals.	RW	0x0

GPLMUX12

Selection between GPIO and LoanIO output and output enable for GPIO12 and LoanIO12. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08604

Offset: 0x604

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX12 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 12. 0 : LoanIO 12 controls GPIO/LOANIO[12] output and output enable signals. 1 : GPIO 12 controls GPIO/ LOANI[12] output and output enable signals.	RW	0x0

GPLMUX13

Selection between GPIO and LoanIO output and output enable for GPIO13 and LoanIO13. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08608

Offset: 0x608

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX13 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 13. 0 : LoanIO 13 controls GPIO/LOANIO[13] output and output enable signals. 1 : GPIO 13 controls GPIO/ LOANI[13] output and output enable signals.	RW	0x0

GPLMUX14

Selection between GPIO and LoanIO output and output enable for GPIO14 and LoanIO14. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0860C

Offset: 0x60C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX14 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 14. 0 : LoanIO 14 controls GPIO/LOANIO[14] output and output enable signals. 1 : GPIO 14 controls GPIO/ LOANI[14] output and output enable signals.	RW	0x0

GPLMUX15

Selection between GPIO and LoanIO output and output enable for GPIO15 and LoanIO15. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08610

Offset: 0x610

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX15 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 15. 0 : LoanIO 15 controls GPIO/LOANIO[15] output and output enable signals. 1 : GPIO 15 controls GPIO/ LOANI[15] output and output enable signals.	RW	0x0

GPLMUX16

Selection between GPIO and LoanIO output and output enable for GPIO16 and LoanIO16. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08614

Offset: 0x614

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX16 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 16. 0 : LoanIO 16 controls GPIO/LOANIO[16] output and output enable signals. 1 : GPIO 16 controls GPIO/ LOANI[16] output and output enable signals.	RW	0x0

GPLMUX17

Selection between GPIO and LoanIO output and output enable for GPIO17 and LoanIO17. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08618

Offset: 0x618

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX17 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 17. 0 : LoanIO 17 controls GPIO/LOANIO[17] output and output enable signals. 1 : GPIO 17 controls GPIO/ LOANI[17] output and output enable signals.	RW	0x0

GPLMUX18

Selection between GPIO and LoanIO output and output enable for GPIO18 and LoanIO18. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0861C

Offset: 0x61C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX18 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 18. 0 : LoanIO 18 controls GPIO/LOANIO[18] output and output enable signals. 1 : GPIO 18 controls GPIO/ LOANI[18] output and output enable signals.	RW	0x0

GPLMUX19

Selection between GPIO and LoanIO output and output enable for GPIO19 and LoanIO19. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08620

Offset: 0x620

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX19 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 19. 0 : LoanIO 19 controls GPIO/LOANIO[19] output and output enable signals. 1 : GPIO 19 controls GPIO/ LOANI[19] output and output enable signals.	RW	0x0

GPLMUX20

Selection between GPIO and LoanIO output and output enable for GPIO20 and LoanIO20. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08624

Offset: 0x624

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX20 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 20. 0 : LoanIO 20 controls GPIO/LOANIO[20] output and output enable signals. 1 : GPIO 20 controls GPIO/ LOANI[20] output and output enable signals.	RW	0x0

GPLMUX21

Selection between GPIO and LoanIO output and output enable for GPIO21 and LoanIO21. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08628`

Offset: `0x628`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX21 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 21. 0 : LoanIO 21 controls GPIO/LOANIO[21] output and output enable signals. 1 : GPIO 21 controls GPIO/ LOANI[21] output and output enable signals.	`RW`	`0x0`

GPLMUX22

Selection between GPIO and LoanIO output and output enable for GPIO22 and LoanIO22. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0862C`

Offset: `0x62C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX22 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 22. 0 : LoanIO 22 controls GPIO/LOANIO[22] output and output enable signals. 1 : GPIO 22 controls GPIO/ LOANI[22] output and output enable signals.	RW	0x0

GPLMUX23

Selection between GPIO and LoanIO output and output enable for GPIO23 and LoanIO23. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08630

Offset: 0x630

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>sel
RW 0x0</td></tr>
</table>

GPLMUX23 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 23. 0 : LoanIO 23 controls GPIO/LOANIO[23] output and output enable signals. 1 : GPIO 23 controls GPIO/ LOANI[23] output and output enable signals.	RW	0x0

GPLMUX24

Selection between GPIO and LoanIO output and output enable for GPIO24 and LoanIO24. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08634

Offset: 0x634

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX24 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 24. 0 : LoanIO 24 controls GPIO/LOANIO[24] output and output enable signals. 1 : GPIO 24 controls GPIO/LOANI[24] output and output enable signals.	RW	0x0

GPLMUX25

Selection between GPIO and LoanIO output and output enable for GPIO25 and LoanIO25. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08638

Offset: `0x638`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX25 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 25. 0 : LoanIO 25 controls GPIO/LOANIO[25] output and output enable signals. 1 : GPIO 25 controls GPIO/LOANI[25] output and output enable signals.	RW	0x0

GPLMUX26

Selection between GPIO and LoanIO output and output enable for GPIO26 and LoanIO26. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0863C

Offset: 0x63C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX26 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 26. 0 : LoanIO 26 controls GPIO/LOANIO[26] output and output enable signals. 1 : GPIO 26 controls GPIO/LOANI[26] output and output enable signals.	RW	0x0

GPLMUX27

Selection between GPIO and LoanIO output and output enable for GPIO27 and LoanIO27. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08640

Offset: 0x640

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX27 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 27. 0 : LoanIO 27 controls GPIO/LOANIO[27] output and output enable signals. 1 : GPIO 27 controls GPIO/LOANI[27] output and output enable signals.	RW	0x0

GPLMUX28

Selection between GPIO and LoanIO output and output enable for GPIO28 and LoanIO28. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08644

Offset: 0x644

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX28 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 28. 0 : LoanIO 28 controls GPIO/LOANIO[28] output and output enable signals. 1 : GPIO 28 controls GPIO/LOANI[28] output and output enable signals.	RW	0x0

GPLMUX29

Selection between GPIO and LoanIO output and output enable for GPIO29 and LoanIO29. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08648

Offset: 0x648

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX29 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 29. 0 : LoanIO 29 controls GPIO/LOANIO[29] output and output enable signals. 1 : GPIO 29 controls GPIO/ LOANI[29] output and output enable signals.	RW	0x0

GPLMUX30

Selection between GPIO and LoanIO output and output enable for GPIO30 and LoanIO30. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0864C

Offset: 0x64C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX30 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 30. 0 : LoanIO 30 controls GPIO/LOANIO[30] output and output enable signals. 1 : GPIO 30 controls GPIO/ LOANI[30] output and output enable signals.	RW	0x0

GPLMUX31

Selection between GPIO and LoanIO output and output enable for GPIO31 and LoanIO31. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08650

Offset: 0x650

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX31 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 31. 0 : LoanIO 31 controls GPIO/LOANIO[31] output and output enable signals. 1 : GPIO 31 controls GPIO/LOANI[31] output and output enable signals.	RW	0x0

GPLMUX32

Selection between GPIO and LoanIO output and output enable for GPIO32 and LoanIO32. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08654

Offset: 0x654

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX32 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 32. 0 : LoanIO 32 controls GPIO/LOANIO[32] output and output enable signals. 1 : GPIO 32 controls GPIO/ LOANI[32] output and output enable signals.	RW	0x0

GPLMUX33

Selection between GPIO and LoanIO output and output enable for GPIO33 and LoanIO33. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08658

Offset: 0x658

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX33 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 33. 0 : LoanIO 33 controls GPIO/LOANIO[33] output and output enable signals. 1 : GPIO 33 controls GPIO/ LOANI[33] output and output enable signals.	RW	0x0

GPLMUX34

Selection between GPIO and LoanIO output and output enable for GPIO34 and LoanIO34. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0865C

Offset: 0x65C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX34 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 34. 0 : LoanIO 34 controls GPIO/LOANIO[34] output and output enable signals. 1 : GPIO 34 controls GPIO/ LOANI[34] output and output enable signals.	`RW`	`0x0`

GPLMUX35

Selection between GPIO and LoanIO output and output enable for GPIO35 and LoanIO35. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08660`

Offset: `0x660`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX35 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 35. 0 : LoanIO 35 controls GPIO/LOANIO[35] output and output enable signals. 1 : GPIO 35 controls GPIO/ LOANI[35] output and output enable signals.	`RW`	`0x0`

GPLMUX36

Selection between GPIO and LoanIO output and output enable for GPIO36 and LoanIO36. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08664

Offset: 0x664

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX36 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 36. 0 : LoanIO 36 controls GPIO/LOANIO[36] output and output enable signals. 1 : GPIO 36 controls GPIO/ LOANI[36] output and output enable signals.	RW	0x0

GPLMUX37

Selection between GPIO and LoanIO output and output enable for GPIO37 and LoanIO37. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08668

Offset: 0x668

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX37 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 37. 0 : LoanIO 37 controls GPIO/LOANIO[37] output and output enable signals. 1 : GPIO 37 controls GPIO/ LOANI[37] output and output enable signals.	RW	0x0

GPLMUX38

Selection between GPIO and LoanIO output and output enable for GPIO38 and LoanIO38. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0866C

Offset: 0x66C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX38 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 38. 0 : LoanIO 38 controls GPIO/LOANIO[38] output and output enable signals. 1 : GPIO 38 controls GPIO/ LOANI[38] output and output enable signals.	RW	0x0

GPLMUX39

Selection between GPIO and LoanIO output and output enable for GPIO39 and LoanIO39. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08670

Offset: 0x670

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX39 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 39. 0 : LoanIO 39 controls GPIO/LOANIO[39] output and output enable signals. 1 : GPIO 39 controls GPIO/LOANI[39] output and output enable signals.	RW	0x0

GPLMUX40

Selection between GPIO and LoanIO output and output enable for GPIO40 and LoanIO40. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08674`

Offset: `0x674`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX40 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 40. 0 : LoanIO 40 controls GPIO/LOANIO[40] output and output enable signals. 1 : GPIO 40 controls GPIO/LOANI[40] output and output enable signals.	RW	0x0

GPLMUX41

Selection between GPIO and LoanIO output and output enable for GPIO41 and LoanIO41. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08678

Offset: 0x678

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX41 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 41. 0 : LoanIO 41 controls GPIO/LOANIO[41] output and output enable signals. 1 : GPIO 41 controls GPIO/ LOANI[41] output and output enable signals.	RW	0x0

GPLMUX42

Selection between GPIO and LoanIO output and output enable for GPIO42 and LoanIO42. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0867C

Offset: 0x67C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX42 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 42. 0 : LoanIO 42 controls GPIO/LOANIO[42] output and output enable signals. 1 : GPIO 42 controls GPIO/ LOANI[42] output and output enable signals.	RW	0x0

GPLMUX43

Selection between GPIO and LoanIO output and output enable for GPIO43 and LoanIO43. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08680

Offset: 0x680

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX43 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 43. 0 : LoanIO 43 controls GPIO/LOANIO[43] output and output enable signals. 1 : GPIO 43 controls GPIO/ LOANI[43] output and output enable signals.	RW	0x0

GPLMUX44

Selection between GPIO and LoanIO output and output enable for GPIO44 and LoanIO44. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08684

Offset: 0x684

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX44 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 44. 0 : LoanIO 44 controls GPIO/LOANIO[44] output and output enable signals. 1 : GPIO 44 controls GPIO/ LOANI[44] output and output enable signals.	RW	0x0

GPLMUX45

Selection between GPIO and LoanIO output and output enable for GPIO45 and LoanIO45. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08688

Offset: 0x688

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX45 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 45. 0 : LoanIO 45 controls GPIO/LOANIO[45] output and output enable signals. 1 : GPIO 45 controls GPIO/ LOANI[45] output and output enable signals.	RW	0x0

GPLMUX46

Selection between GPIO and LoanIO output and output enable for GPIO46 and LoanIO46. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD0868C`

Offset: `0x68C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX46 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 46. 0 : LoanIO 46 controls GPIO/LOANIO[46] output and output enable signals. 1 : GPIO 46 controls GPIO/LOANI[46] output and output enable signals.	RW	0x0

GPLMUX47

Selection between GPIO and LoanIO output and output enable for GPIO47 and LoanIO47. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08690`

Offset: `0x690`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX47 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 47. 0 : LoanIO 47 controls GPIO/LOANIO[47] output and output enable signals. 1 : GPIO 47 controls GPIO/ LOANI[47] output and output enable signals.	RW	0x0

GPLMUX48

Selection between GPIO and LoanIO output and output enable for GPIO48 and LoanIO48. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08694

Offset: 0x694

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX48 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 48. 0 : LoanIO 48 controls GPIO/LOANIO[48] output and output enable signals. 1 : GPIO 48 controls GPIO/ LOANI[48] output and output enable signals.	RW	0x0

GPLMUX49

Selection between GPIO and LoanIO output and output enable for GPIO49 and LoanIO49. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08698

Offset: 0x698

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX49 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 49. 0 : LoanIO 49 controls GPIO/LOANIO[49] output and output enable signals. 1 : GPIO 49 controls GPIO/ LOANI[49] output and output enable signals.	RW	0x0

GPLMUX50

Selection between GPIO and LoanIO output and output enable for GPIO50 and LoanIO50. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0869C

Offset: 0x69C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX50 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 50. 0 : LoanIO 50 controls GPIO/LOANIO[50] output and output enable signals. 1 : GPIO 50 controls GPIO/ LOANI[50] output and output enable signals.	RW	0x0

GPLMUX51

Selection between GPIO and LoanIO output and output enable for GPIO51 and LoanIO51. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086A0`

Offset: `0x6A0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX51 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 51. 0 : LoanIO 51 controls GPIO/LOANIO[51] output and output enable signals. 1 : GPIO 51 controls GPIO/ LOANI[51] output and output enable signals.	`RW`	`0x0`

GPLMUX52

Selection between GPIO and LoanIO output and output enable for GPIO52 and LoanIO52. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086A4`

Offset: `0x6A4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX52 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 52. 0 : LoanIO 52 controls GPIO/LOANIO[52] output and output enable signals. 1 : GPIO 52 controls GPIO/LOANI[52] output and output enable signals.	RW	0x0

GPLMUX53

Selection between GPIO and LoanIO output and output enable for GPIO53 and LoanIO53. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086A8

Offset: 0x6A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX53 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 53. 0 : LoanIO 53 controls GPIO/LOANIO[53] output and output enable signals. 1 : GPIO 53 controls GPIO/LOANI[53] output and output enable signals.	RW	0x0

GPLMUX54

Selection between GPIO and LoanIO output and output enable for GPIO54 and LoanIO54. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086AC

Offset: 0x6AC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX54 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 54. 0 : LoanIO 54 controls GPIO/LOANIO[54] output and output enable signals. 1 : GPIO 54 controls GPIO/LOANI[54] output and output enable signals.	RW	0x0

GPLMUX55

Selection between GPIO and LoanIO output and output enable for GPIO55 and LoanIO55. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086B0`

Offset: `0x6B0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX55 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 55. 0 : LoanIO 55 controls GPIO/LOANIO[55] output and output enable signals. 1 : GPIO 55 controls GPIO/LOANI[55] output and output enable signals.	RW	0x0

GPLMUX56

Selection between GPIO and LoanIO output and output enable for GPIO56 and LoanIO56. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086B4

Offset: 0x6B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX56 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 56. 0 : LoanIO 56 controls GPIO/LOANIO[56] output and output enable signals. 1 : GPIO 56 controls GPIO/LOANI[56] output and output enable signals.	RW	0x0

GPLMUX57

Selection between GPIO and LoanIO output and output enable for GPIO57 and LoanIO57. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086B8

Offset: 0x6B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX57 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 57. 0 : LoanIO 57 controls GPIO/LOANIO[57] output and output enable signals. 1 : GPIO 57 controls GPIO/LOANI[57] output and output enable signals.	RW	0x0

GPLMUX58

Selection between GPIO and LoanIO output and output enable for GPIO58 and LoanIO58. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086BC

Offset: 0x6BC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX58 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 58. 0 : LoanIO 58 controls GPIO/LOANIO[58] output and output enable signals. 1 : GPIO 58 controls GPIO/LOANI[58] output and output enable signals.	RW	0x0

GPLMUX59

Selection between GPIO and LoanIO output and output enable for GPIO59 and LoanIO59. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086C0

Offset: 0x6C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX59 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 59. 0 : LoanIO 59 controls GPIO/LOANIO[59] output and output enable signals. 1 : GPIO 59 controls GPIO/ LOANI[59] output and output enable signals.	RW	0x0

GPLMUX60

Selection between GPIO and LoanIO output and output enable for GPIO60 and LoanIO60. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086C4

Offset: 0x6C4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX60 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 60. 0 : LoanIO 60 controls GPIO/LOANIO[60] output and output enable signals. 1 : GPIO 60 controls GPIO/ LOANI[60] output and output enable signals.	RW	0x0

GPLMUX61

Selection between GPIO and LoanIO output and output enable for GPIO61 and LoanIO61. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086C8

Offset: 0x6C8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX61 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 61. 0 : LoanIO 61 controls GPIO/LOANIO[61] output and output enable signals. 1 : GPIO 61 controls GPIO/ LOANI[61] output and output enable signals.	RW	0x0

GPLMUX62

Selection between GPIO and LoanIO output and output enable for GPIO62 and LoanIO62. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086CC

Offset: 0x6CC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX62 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 62. 0 : LoanIO 62 controls GPIO/LOANIO[62] output and output enable signals. 1 : GPIO 62 controls GPIO/ LOANI[62] output and output enable signals.	RW	0x0

GPLMUX63

Selection between GPIO and LoanIO output and output enable for GPIO63 and LoanIO63. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086D0

Offset: 0x6D0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>sel
RW 0x0</td></tr>
</table>

GPLMUX63 Fields

Bit	Name	Description	Access	Reset
0	sel	Select source for GPIO/LoanIO 63. 0 : LoanIO 63 controls GPIO/LOANIO[63] output and output enable signals. 1 : GPIO 63 controls GPIO/ LOANI[63] output and output enable signals.	RW	0x0

GPLMUX64

Selection between GPIO and LoanIO output and output enable for GPIO64 and LoanIO64. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086D4

Offset: 0x6D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX64 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 64. 0 : LoanIO 64 controls GPIO/LOANIO[64] output and output enable signals. 1 : GPIO 64 controls GPIO/LOANI[64] output and output enable signals.	`RW`	`0x0`

GPLMUX65

Selection between GPIO and LoanIO output and output enable for GPIO65 and LoanIO65. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086D8`

Offset: `0x6D8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX65 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 65. 0 : LoanIO 65 controls GPIO/LOANIO[65] output and output enable signals. 1 : GPIO 65 controls GPIO/LOANI[65] output and output enable signals.	`RW`	`0x0`

GPLMUX66

Selection between GPIO and LoanIO output and output enable for GPIO66 and LoanIO66. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings

Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086DC`

Offset: `0x6DC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX66 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 66. 0 : LoanIO 66 controls GPIO/LOANIO[66] output and output enable signals. 1 : GPIO 66 controls GPIO/LOANI[66] output and output enable signals.	RW	0x0

GPLMUX67

Selection between GPIO and LoanIO output and output enable for GPIO67 and LoanIO67. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086E0`

Offset: `0x6E0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX67 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 67. 0 : LoanIO 67 controls GPIO/LOANIO[67] output and output enable signals. 1 : GPIO 67 controls GPIO/ LOANI[67] output and output enable signals.	`RW`	`0x0`

GPLMUX68

Selection between GPIO and LoanIO output and output enable for GPIO68 and LoanIO68. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086E4`

Offset: `0x6E4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`sel` `RW 0x0`

GPLMUX68 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 68. 0 : LoanIO 68 controls GPIO/LOANIO[68] output and output enable signals. 1 : GPIO 68 controls GPIO/ LOANI[68] output and output enable signals.	`RW`	`0x0`

GPLMUX69

Selection between GPIO and LoanIO output and output enable for GPIO69 and LoanIO69. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086E8`

Offset: `0x6E8`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX69 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 69. 0 : LoanIO 69 controls GPIO/LOANIO[69] output and output enable signals. 1 : GPIO 69 controls GPIO/LOANI[69] output and output enable signals.	RW	0x0

GPLMUX70

Selection between GPIO and LoanIO output and output enable for GPIO70 and LoanIO70. These signals drive the Pin Mux. The Pin Mux must be configured to use GPIO/LoanIO in addition to these settings Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD086EC`

Offset: `0x6EC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

GPLMUX70 Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select source for GPIO/LoanIO 70. 0 : LoanIO 70 controls GPIO/LOANIO[70] output and output enable signals. 1 : GPIO 70 controls GPIO/LOANI[70] output and output enable signals.	RW	0x0

NANDUSEFPGA

Selection between HPS Pins and FPGA Interface for NAND signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086F0

Offset: 0x6F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

NANDUSEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for NAND. 0 : NAND uses HPS Pins. 1 : NAND uses the FPGA Inteface.	RW	0x0

RGMII1USEFPGA

Selection between HPS Pins and FPGA Interface for RGMII1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD086F8

Offset: 0x6F8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

RGMII1USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for RGMII1. 0 : RGMII1 uses HPS Pins. 1 : RGMII1 uses the FPGA Inteface.	RW	0x0

I2C0USEFPGA

Selection between HPS Pins and FPGA Interface for I2C0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08704`

Offset: `0x704`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

I2C0USEFPGA Fields

Bit	Name	Description	Access	Reset
0	`sel`	Select connection for I2C0. 0 : I2C0 uses HPS Pins. 1 : I2C0 uses the FPGA Inteface.	`RW`	`0x0`

RGMII0USEFPGA

Selection between HPS Pins and FPGA Interface for RGMII0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
`sysmgr`	`0xFFD08000`	`0xFFD08714`

Offset: `0x714`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

RGMII0USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for RGMII0. 0 : RGMII0 uses HPS Pins. 1 : RGMII0 uses the FPGA Inteface.	RW	0x0

I2C3USEFPGA

Selection between HPS Pins and FPGA Interface for I2C3 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08724

Offset: 0x724

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

I2C3USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for I2C3. 0 : I2C3 uses HPS Pins. 1 : I2C3 uses the FPGA Inteface.	RW	0x0

I2C2USEFPGA

Selection between HPS Pins and FPGA Interface for I2C2 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08728

Offset: 0x728

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

I2C2USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for I2C2. 0 : I2C2 uses HPS Pins. 1 : I2C2 uses the FPGA Inteface.	RW	0x0

I2C1USEFPGA

Selection between HPS Pins and FPGA Interface for I2C1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD0872C

Offset: 0x72C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

I2C1USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for I2C1. 0 : I2C1 uses HPS Pins. 1 : I2C1 uses the FPGA Inteface.	RW	0x0

SPIM1USEFPGA

Selection between HPS Pins and FPGA Interface for SPIM1 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08730

Offset: 0x730

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

SPIM1USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for SPIM1. 0 : SPIM1 uses HPS Pins. 1 : SPIM1 uses the FPGA Inteface.	RW	0x0

SPIM0USEFPGA

Selection between HPS Pins and FPGA Interface for SPIM0 signals. Only reset by a cold reset (ignores warm reset). NOTE: These registers should not be modified after IO configuration.There is no support for dynamically changing the Pin Mux selections.

Module Instance	Base Address	Register Address
sysmgr	0xFFD08000	0xFFD08738

Offset: 0x738

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sel RW 0x0

SPIM0USEFPGA Fields

Bit	Name	Description	Access	Reset
0	sel	Select connection for SPIM0. 0 : SPIM0 uses HPS Pins. 1 : SPIM0 uses the FPGA Inteface.	RW	0x0

Document Revision History

Table 5-4: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Added address map and register descriptions • Updated ECC Parity Control CAN controller section added
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release.
November 2012	1.2	Minor updates.
May 2012	1.1	Added functional description, address map and register definitions sections.
January 2012	1.0	Initial release.

Scan Manager 6

2014.06.30

cv_54015 Subscribe Send Feedback

The scan manager is used to configure and manage the HPS I/O pins, and communicate with the FPGA JTAG test access port (TAP) controller. The scan manager drives the HPS I/O scan chains to configure the I/O bank properties before the pins are used by the peripherals in HPS. The scan manager can also optionally communicate with the FPGA JTAG TAP controller to send commands for purposes such as managing cyclic redundancy check (CRC) errors detected by the FPGA control block. When the scan manager communicates with the FPGA JTAG TAP controller, input on the FPGA JTAG pins is ignored.

The scan manager contains an ARM® JTAG Access Port (JTAG-AP). The JTAG-AP implements a multiple scan-chain JTAG master interface. One scan chain connects to the FPGA JTAG and uses the standard JTAG signals. Four other scan chains connect to the HPS I/O banks, using the JTAG clock and data outputs as a parallel-to-serial converter.

Related Information

http://infocenter.arm.com/

For more information about the ARM JTAG-AP, refer to the *DAP Components* chapter of the *CoreSight SoC Technical Reference Manual*, which you can download from the ARM Infocenter website.

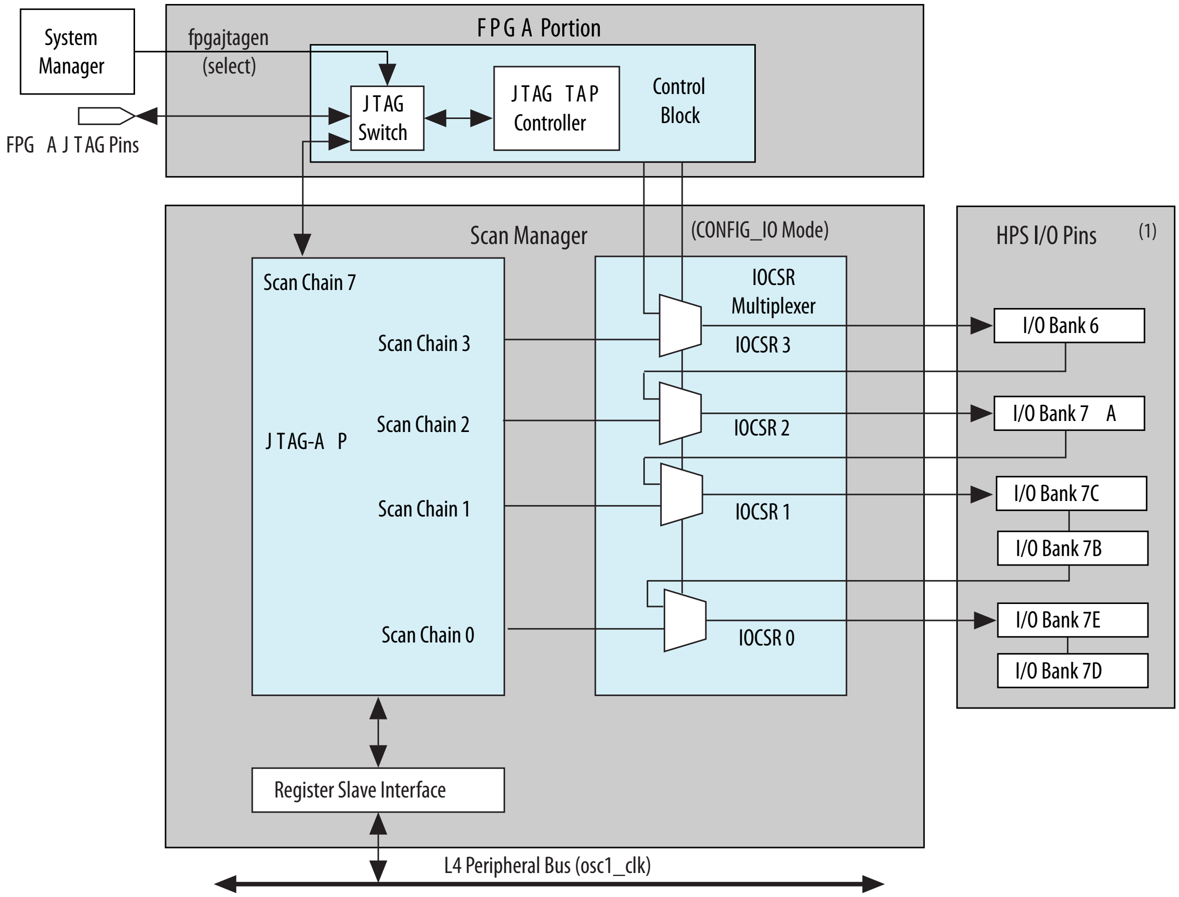
Features of the Scan Manager

- Drives all the I/O scan chains for HPS I/O banks
- Allows the HPS to access the FPGA JTAG TAP controller

Scan Manager Block Diagram and System Integration

Figure 6-1: Scan Manager Block Diagram

[1] Not all devices contain all the banks depicted.



The processor accesses the scan manager through the register slave interface connected to the level 4 (L4) peripheral bus.

ARM JTAG-AP Signal Use in the Scan Manager

The following table describes how the ARM JTAG-AP signals are connected in the scan manager. These signals are internal to the scan manager, are provided here for reference only, and are not shown in the preceding figure. The signal, register, and field names listed in the table match the names used in the *ARM Debug Interface v5 Architecture Specification*.

Signal	Direction	Implementation
`SRSTCONNECTED[7:0]`	Input	Tied to 0. The read-only `SRSTCONNECTED` field in the `CSW` register always reads as 0.

Signal	Direction	Implementation
PORTCONNECTED[7:0]	Input	Tied to 0x8F, which connects only ports 0-3 and 7. The read-only PORTCONNECTED field in the CSW register reads as 1 when the PORTSEL register is written with a value that enables one of the connected ports, and reads as 0 otherwise.
PORTENABLED[7:0]	Input	Tied to 0x8F, so all connected ports are always considered powered on. The ARM JTAG AP PSTA register is not supported. Software does not need to monitor the status of ports 0-3 because they are always on. For port 7, software can read the mode field of the stat register in the FPGA manager to determine the FPGA power status.
nSRSTOUT[7:0]	Output	Not connected. Writing to the SRST_OUT field of the CSW register has no effect.
nTRST*[7:0]	Output	nTRST*[7] is connected to the FPGA JTAG TAP controller and nTRST*[6:0] are not connected. Writing to the TRST_OUT field of the CSW register (the trst bit of the stat register in the scan manager) has an effect only when port 7 is enabled by software. For details, refer to *"Communicating with the JTAG TAP Controller"*.

Related Information

- stat on page 6-9
  Information about configuring the scan manager's stat register
- **Communicating with the JTAG TAP Controller** on page 6-5
- **http://infocenter.arm.com/**
  For detailed information about the ARM JTAG-AP CSW, PORTSEL, and PSTA registers, refer to the *DAP Components* chapter of the *CoreSight SoC Technical Reference Manual*, which you can download from the ARM Infocenter website.

ARM JTAG-AP Scan Chains

The ARM JTAG-AP supports up to eight scan chains. The scan manager uses only scan chains 0, 1, 2, 3, and 7.

Scan chain 7 of the JTAG-AP connects to FPGA JTAG TAP controller. When the system manager undergoes a cold reset, this connection is disabled and the FPGA JTAG pins are connected to the FPGA JTAG TAP controller. You can configure the system manager to enable the connection, which allows software running on the HPS to communicate with the FPGA JTAG TAP controller. In this case, software can send JTAG commands (such as the SHIFT_EDERROR_REG JTAG instruction) to the FPGA JTAG and get responses to determine details about CRC errors detected by the control block when the FPGA fabric is in user mode. Through the FPGA manager, software can determine that a CRC error was detected. For more information about the TAP controller, refer to the *Communicating with the JTAG TAP Controller* section of this chapter.

Scan chains 0 to 3 of the JTAG-AP connect to the configuration information in the HPS I/O scan chain banks through the I/O configuration shift register (IOCSR) multiplexer. For more information, refer to the *Configuring HPS I/O Scan Chains* section of this chapter.

Note: The I/O scan chains do not use the JTAG protocol. The scan manager uses the JTAG-AP as a parallel-to-serial converter for the I/O scan chains. The I/O scan chains are connected only to the serial output data (`TDI` JTAG signal) and serial clock (`TCK` JTAG signal).

The HPS I/O pins are divided into six banks. Each I/O bank is either a vertical (VIO) or horizontal (HIO) I/O, based on its location on the die.

Table 6-1: Bank Usage of IOCSR Scan Chains

The following table shows the mapping of the IOCSR scan chains to the I/O banks.

IOCSR Scan Chain	Bank Type	HPS I/O Bank	Usage
0	VIO	I/O bank 7D and I/O bank 7E	EMAC
1	VIO	I/O bank 7B and I/O bank 7C	SD/MMC, NAND, and quad SPI
2	VIO	I/O bank 7A	Trace, SPI, UART, I^{2}C, and CAN
3	HIO	I/O bank 6	SDRAM DDR

When the FPGA JTAG TAP controller is in `CONFIG_IO` mode, the controller can override the scan manager JTAG-AP and configure the HPS I/O pins. For more information, refer to the *Configuring HPS I/O Scan Chains* section of this chapter.

Note: `CONFIG_IO` mode is commonly used to configure the I/O pin properties prior to performing boundary scan testing.

Note: The HPS JTAG pins does not support boundary scan tests (BST). To perform boundary scan testing on HPS I/O pins, use the FPGA JTAG.

Related Information

- **Configuring HPS I/O Scan Chains** on page 6-4
- **Communicating with the JTAG TAP Controller** on page 6-5

Functional Description of the Scan Manager

The scan manager serves the following purposes:

- Configuring HPS I/O scan chains
- Communicating with the JTAG TAP controller

Configuring HPS I/O Scan Chains

The HPS I/O pins are configured through a series of scan chains.

I/O pin configuration involves such steps as setting the I/O standard and drive strength for each I/O bank. After a cold reset, all the I/O scan chains in the HPS must be configured prior to being used to communicate with external devices.

Software uses the scan manager to write configuration data to the scan chains. Separate I/O configuration data files for FPGA and HPS are generated by the Quartus® II software when the configuration image for the FPGA portion of the system-on-a-chip (SoC) device is assembled. The HPS configuration data is written to the scan manager by software.

Before configuring a specific I/O bank, the corresponding scan chain must be enabled by writing to the bits in the en register. The scan manager must not be active during this process. Software reads the active bit of the `stat` register to determine the scan manager state.

Alternatively, when the FPGA JTAG TAP controller receives the `CONFIG_IO` JTAG instruction, the control block enters `CONFIG_IO` mode. When the control block is in `CONFIG_IO` mode, the controller can override the scan manager JTAG-AP and configure the HPS I/O pins. The `CONFIG_IO` instruction configures all configurable I/O pins in the SoC device including the FPGA I/O pins and the HPS I/O pins. The FPGA and HPS portions of the device must both be powered on to execute the `CONFIG_IO` instruction. External logic connected to the FPGA JTAG pins sends the `CONFIG_IO` instruction, which provides I/O configuration data for all FPGA and HPS I/O pins. While `CONFIG_IO` mode is active, the HPS is held in cold reset to prevent software from potentially interfering with the I/O configuration.

Related Information

- **stat** on page 6-9
  Information about configuring the scan manager's stat register
- **en** on page 6-11
  Information about configuring the scan manager's en register
- **Scan Manager Address Map and Register Definitions** on page 6-8
- **System Manager** on page 5-1
  The HPS I/O pins need to be frozen before configuring them. For more information, refer to the *System Manager* chapter.

Communicating with the JTAG TAP Controller

After the system manager undergoes a cold reset, access to the JTAG TAP controller in the FPGA control block is through the dedicated FPGA JTAG I/O pins. If necessary, you can configure your system to use the scan manager to provide the HPS processor access to the JTAG TAP controller, instead. This feature allows the processor to send JTAG instructions to the FPGA portion of the device.

To connect scan chain 7 between the scan manager and the FPGA JTAG TAP controller, the following features must be enabled:

- The scan chain for the FPGA JTAG TAP controller—To enable scan chain 7, set the `fpgajtag` field of the en register in the scan manager. For more information, refer to *"Scan Manager Address Map and Register Definitions"*.
- The FPGA JTAG logic source select—This source select determines whether the scan manager or the dedicated FPGA JTAG pins are connected to the FPGA JTAG TAP controller in the FPGA portion of the device. On system manager cold reset, the dedicated FPGA JTAG pins are selected. The source select is configured through the `fpgajtagen` bit of the `ctrl` register in the `scanmgrgrp` group of the system manager. The FPGA JTAG pins and scan manager connection to the TAP controller must both be inactive when switching between them. The mechanism to ensure both are inactive is user-defined.

Note: Before connecting or disconnecting the scan chain between the scan manager and the FPGA JTAG TAP controller, ensure that both the FPGA JTAG `TCK` and scan manager `TCK` signals are de-asserted. Altera recommends resetting the FPGA JTAG TAP controller using the scan manager's `nTRST` signal after the scan manager is connected to the controller.

Related Information

- **en** on page 6-11
  Information about configuring the scan manager's en register
- **Scan Manager Address Map and Register Definitions** on page 6-8
- **System Manager** on page 5-1
  For information about the system manager, including details about configuring the `ctrl` register, refer to the *System Manager* chapter.

JTAG-AP FIFO Buffer Access and Byte Command Protocol

The JTAG-AP contains FIFO buffers for byte commands and responses. The buffers are accessed through the `fifosinglebyte`, `fifodoublebyte`, `fifotriplebyte`, and `fifoquadbyte` registers. The JTAG-AP stalls processor access to the registers when the buffer does not contain enough data for read access, or when the buffer does not contain enough free space to accept data for write access.

Note: Software should read the `rfifocnt` and `wfifocnt` fields of the `stat` register to determine the buffer status before performing the access to avoid being stalled by the JTAG-AP.

JTAG-AP scan chains 0, 1, 2 and 3 are write-only ports connected to the HPS IOCSRs and JTAG-AP scan chain 7 is a read-write port connected to the FPGA JTAG TAP controller. The processor can send data to scan chains 0-3, and send and receive data from scan chain 7 by accessing the command and response FIFO buffers in the JTAG-AP.

Note: Attempting to access data at invalid or non-aligned offsets can produce unpredictable results that require a reset to recover.

The JTAG commands and `TDI` data must be sent to the JTAG-AP using an encoded byte protocol. Similarly, the `TDO` data received from JTAG-AP is encoded. All commands are 8 bits wide in the byte command protocol.

Table 6-2: JTAG-AP Byte Command Protocol

Bits of the Command Byte								Opcode
7	6	5	4	3	2	1	0	
0	Opcode Payload							`TMS`
1	0	0	Opcode Payload					`TDI_TDO`
1	0	1	X	X	X	X	X	Reserved
1	1	0	X	X	X	X	X	Reserved
1	1	1	X	X	X	X	X	Reserved

Related Information

- **fifosinglebyte** on page 6-12
  Information about configuring the scan manager's fifosinglebyte register
- **fifodoublebyte** on page 6-13
  Information about configuring the scan manager's fifodoublebyte register
- **fifotriplebyte** on page 6-13
  Information about configuring the scan manager's fifotriplebyte register
- **fifoquadbyte** on page 6-14
  Information about configuring the scan manager's fifoquadbyte register
- **stat** on page 6-9
  Information about configuring the scan manager's stat register
- **Scan Manager Address Map and Register Definitions** on page 6-8
- **http://infocenter.arm.com/**
  For details about the byte command protocol, refer to the *DAP Components* chapter of the *CoreSight SoC Technical Reference Manual*, which you can download from the ARM Infocenter website.

Clocks

The scan manager is connected to the `spi_m_clk` clock generated by the clock manager.

The scan manager generates two clocks. One clock routes to the control block of the FPGA portion of the SoC device with a frequency of `spi_m_clk` / 6 and runs at a maximum of 33 MHz. The other clock routes to the HPS I/O scan chains with a frequency of `sp i_m_clk` / 2 and runs at a maximum frequency of 100 MHz.

Note: The `spi_m_clk` can potentially run faster than the scan manager supports so that SPI masters can support 60 Mbps rates. When the SPI master is running faster than what is supported by the scan manager, the scan manager cannot be used and must be held in reset.

Related Information

Clock Manager on page 2-1
For more information, including minimum and maximum clock frequencies, refer to the *Clock Manager* chapter.

Resets

The reset manager provides the `scan_manager_rst_n` reset signal to the scan manager for both cold and warm resets.

Because glitches can happen on the output clocks during a warm reset, the scan manager temporarily stops generation of the JTAG-AP and I/O configuration clocks. This action ensures that a warm reset does not cause output clock glitches.

Before asserting warm reset, the reset manager sends a request to the scan manager. The scan manager stops the output clock generation and acknowledges the reset manager. The reset manager then issues the warm reset. To enable this warm reset handshake, configure the `scanmgrhsen` bit of the reset manager `ctrl` register.

Related Information

- **Reset Manager** on page 3-1
  For more information about reset handshaking, refer to the *Reset Manager* chapter.

- System Manager on page 5-1
  For information about the system manager, including details about configuring the `ctrl` register, refer to the *System Manager* chapter.

Scan Manager Address Map and Register Definitions

This section lists the scan manager register address map and describes the registers.

Related Information

- Introduction to Cyclone V Hard Processor System on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor* chapter.
- Cyclone V Address Map and Register Definitions
  Web-based address map and register definitions

JTAG-AP Register Name Cross Reference Table

To clarify how Altera uses the JTAG-AP, the ARM registers are renamed in the SoC device. The following table cross references the ARM and Altera register names.

Table 6-3: JTAG-AP Register Names

Altera Register Name	ARM Register Name
`stat`	`CSW` (control/status word)
`en`	`PSEL`
`fifosinglebyte`	`BWFIFO1` for writes, `BRFIFO1` for reads
`fifodoublebyte`	`BWFIFO2` for writes, `BRFIFO2` for reads
`fifotriplebyte`	`BWFIFO3` for writes, `BRFIFO3` for reads
`fifoquadbyte`	`BWFIFO4` for writes, `BRFIFO4` for reads

Related Information

http://infocenter.arm.com/
For more information about the ARM JTAG-AP, refer to the *DAP Components* chapter of the *CoreSight SoC Technical Reference Manual*, which you can download from the ARM Infocenter website.

Scan Manager Module Registers Address Map

Registers in the Scan Manager module. These registers are implemented by an ARM JTAG-AP module from the ARM DAP. Some register and field names have been changed to match the usage in the Scan Manager. If modified, the corresponding names from the ARM documentation are provided. Only registers and fields that are relevant to the JTAG-AP use in the Scan Manager are listed.

Base Address: `0xFFF02000`

Scan Manager Module Registers

Register	Offset	Width	Acces s	Reset Value	Description
`stat` on page 6-9	0x0	32	RW	0x0	Control/Status Word Register
`en` on page 6-11	0x4	32	RW	0x0	Scan-Chain Enable Register
`fifosinglebyte` on page 6-12	0x10	32	RW	0x0	FIFO Single Byte Register
`fifodoublebyte` on page 6-13	0x14	32	RW	0x0	FIFO Double Byte Register
`fifotriplebyte` on page 6-13	0x18	32	RW	0x0	FIFO Triple Byte Register
`fifoquadbyte` on page 6-14	0x1C	32	RW	0x0	FIFO Quad Byte Register

stat

Consist of control bit and status information.

Module Instance	Base Address	Register Address
scanmgr	0xFFF02000	0xFFF02000

Offset: `0x0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
active RO 0x0	wfifocnt RO 0x0			Reserved	rfifocnt RO 0x0			Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ignore RO 0x0	Reserved	trst RW 0x0	Reserved

stat Fields

Bit	Name	Description	Access	Reset
31	`active`	Indicates if the Scan-Chain Engine is processing commands from the Command FIFO or not. The Scan-Chain Engine is only guaranteed to be inactive if both the ACTIVE and WFIFOCNT fields are zero. The name of this field in ARM documentation is SERACTV. **Value** **Description** 0x0 The Scan-Chain Engine may or may not be processing commands from the Command FIFO. The Scan-Chain Engine is only guaranteed to be inactive if both this ACTIVE field and the WFIFOCNT fields are both zero. 0x1 The Scan-Chain Engine is processing commands from the Command FIFO.	RO	0x0
30:28	`wfifocnt`	Command FIFO outstanding byte count. Returns the number of command bytes held in the Command FIFO that have yet to be processed by the Scan-Chain Engine.	RO	0x0
26:24	`rfifocnt`	Response FIFO outstanding byte count. Returns the number of bytes of response data available in the Response FIFO.	RO	0x0
3	`ignore`	Ignore this field. Its value is undefined (may be 0 or 1). The name of this field in ARM documentation is PORTCONNECTED.	RO	0x0
1	`trst`	Specifies the value of the nTRST signal driven to the FPGA JTAG only. The FPGA JTAG scan-chain must be enabled via the EN register to drive the value specified in this field. The nTRST signal is driven with the inverted value of this field.The nTRST signal is active low so, when this bit is set to 1, FPGA JTAG is reset. The name of this field in ARM documentation is TRST_OUT. **Value** **Description** 0x0 Don't reset FPGA JTAG. 0x1 Reset FPGA JTAG. Must have the FPGA JTAG scan-chain enabled in the EN register to take effect.	RW	0x0

en

This register is used to enable one of the 5 scan-chains (0-3 and 7). Only one scan-chain must be enabled at a time. A scan-chain is enabled by writing its corresponding enable field. Software must use the System Manager to put the corresponding I/O scan-chain into the frozen state before attempting to send I/O configuration data to the I/O scan-chain. Software must only write to this register when the Scan-Chain Engine is inactive.Writing this field at any other time has unpredictable results. This means that before writing to this field, software must read the STAT register and check that the ACTIVE and WFIFOCNT fields are both zero. The name of this register in ARM documentation is PSEL.

Module Instance	Base Address	Register Address
scanmgr	0xFFF02000	0xFFF02004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								fpgajtag RW 0x0	Reserved			ioscanchain3 RW 0x0	ioscanchain2 RW 0x0	ioscanchain1 RW 0x0	ioscanchain0 RW 0x0

en Fields

Bit	Name	Description	Access	Reset
7	fpgajtag	Used to enable or disable FPGA JTAG scan-chain.Software must use the System Manager to enable the Scan Manager to drive the FPGA JTAG before attempting to communicate with the FPGA JTAG via the Scan Manager. The name of this field in ARM documentation is PSEL7. **Value** / **Description** 0x0 Disable scan-chain 0x1 Enable scan-chain	RW	0x0

Bit	Name	Description	Access	Reset
3	`ioscanchain3`	Used to enable or disable I/O Scan-Chain 3 The name of this field in ARM documentation is PSEL3. **Value** — **Description** 0x0 — Disable scan-chain 0x1 — Enable scan-chain	RW	0x0
2	`ioscanchain2`	Used to enable or disable I/O Scan-Chain 2 The name of this field in ARM documentation is PSEL2. **Value** — **Description** 0x0 — Disable scan-chain 0x1 — Enable scan-chain	RW	0x0
1	`ioscanchain1`	Used to enable or disable I/O Scan-Chain 1 The name of this field in ARM documentation is PSEL1. **Value** — **Description** 0x0 — Disable scan-chain 0x1 — Enable scan-chain	RW	0x0
0	`ioscanchain0`	Used to enable or disable I/O Scan-Chain 0 The name of this field in ARM documentation is PSEL0. **Value** — **Description** 0x0 — Disable scan-chain 0x1 — Enable scan-chain	RW	0x0

fifosinglebyte

Writes to the FIFO Single Byte Register write a single byte value to the command FIFO. If the command FIFO is full, the APB write operation is stalled until the command FIFO is not full. Reads from the Single Byte FIFO Register read a single byte value from the command FIFO. If the command FIFO is empty, the APB read operation is stalled until the command FIFO is not empty. See the ARM documentation for a description of the read and write values. The name of this register in ARM documentation is BWFIFO1 for writes and BRFIFO1 for reads.

Module Instance	Base Address	Register Address
`scanmgr`	`0xFFF02000`	`0xFFF02010`

Offset: `0x10`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RW 0x0							

fifosinglebyte Fields

Bit	Name	Description	Access	Reset
7:0	value	Transfers single byte value to/from command FIFO	RW	0x0

fifodoublebyte

Writes to the FIFO Double Byte Register write a double byte value to the command FIFO. If the command FIFO is full, the APB write operation is stalled until the command FIFO is not full. Reads from the Double Byte FIFO Register read a double byte value from the command FIFO. If the command FIFO is empty, the APB read operation is stalled until the command FIFO is not empty. See the ARM documentation for a description of the read and write values. The name of this register in ARM documentation is BWFIFO2 for writes and BRFIFO2 for reads.

Module Instance	Base Address	Register Address
scanmgr	0xFFF02000	0xFFF02014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

fifodoublebyte Fields

Bit	Name	Description	Access	Reset
15:0	value	Transfers double byte value to/from command FIFO	RW	0x0

fifotriplebyte

Writes to the FIFO Triple Byte Register write a triple byte value to the command FIFO. If the command FIFO is full, the APB write operation is stalled until the command FIFO is not full. Reads from the Triple Byte FIFO Register read a triple byte value from the command FIFO. If the command FIFO is empty, the APB read operation is stalled until the command FIFO is not empty. See the ARM documentation for a description of the read and write values. The name of this register in ARM documentation is BWFIFO3 for writes and BRFIFO3 for reads.

Module Instance	Base Address	Register Address
scanmgr	0xFFF02000	0xFFF02018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								value RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

fifotriplebyte Fields

Bit	Name	Description	Access	Reset
23:0	value	Transfers triple byte value to/from command FIFO	RW	0x0

fifoquadbyte

Writes to the FIFO Quad Byte Register write a quad byte value to the command FIFO. If the command FIFO is full, the APB write operation is stalled until the command FIFO is not full. Reads from the Quad Byte FIFO Register read a quad byte value from the command FIFO. If the command FIFO is empty, the APB read operation is stalled until the command FIFO is not empty. See the ARM documentation for a description of the read and write values. The name of this register in ARM documentation is BWFIFO4 for writes and BRFIFO4 for reads.

Module Instance	Base Address	Register Address
scanmgr	0xFFF02000	0xFFF0201C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

fifoquadbyte Fields

Bit	Name	Description	Access	Reset
31:0	value	Transfers quad byte value to/from command FIFO	RW	0x0

Document Revision History

Table 6-4: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Add address map and register definitions
February 2014	2014.02.28	Update to "Scan Manager Block Diagram and System Integration" section
December 2013	2013.12.30	Minor formatting issues
November 2012	1.2	Added JTAG-AP descriptions.
May 2012	1.1	Added block diagram and system integration, functional description, and address map and register definitions sections.
January 2012	1.0	Initial release.

System Interconnect 7

2014.06.30

cv_54004 Subscribe Send Feedback

The hard processor system (HPS) level 3 (L3) interconnect and level 4 (L4) peripheral buses are implemented with the ARM CoreLink™ Network Interconnect (NIC-301). The NIC-301 provides a foundation for a high-performance HPS interconnect based on the ARM Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI), Advanced High-Performance Bus (AHB™), and Advanced Peripheral Bus (APB™) protocols. The L3 interconnect implements a multilayer, nonblocking architecture that supports multiple simultaneous transactions between masters and slaves, including the Cortex-A9 microprocessor unit (MPU) subsystem. The interconnect provides five independent L4 buses to access control and status registers (CSRs) of peripherals, managers, and memory controllers

Related Information

http://infocenter.arm.com/
Additional information is available in the *AMBA Network Interconnect (NIC-301) Technical Reference Manual*, revision r2p3, which you can download from the ARM info center website.

Features of the L3 System Interconnect

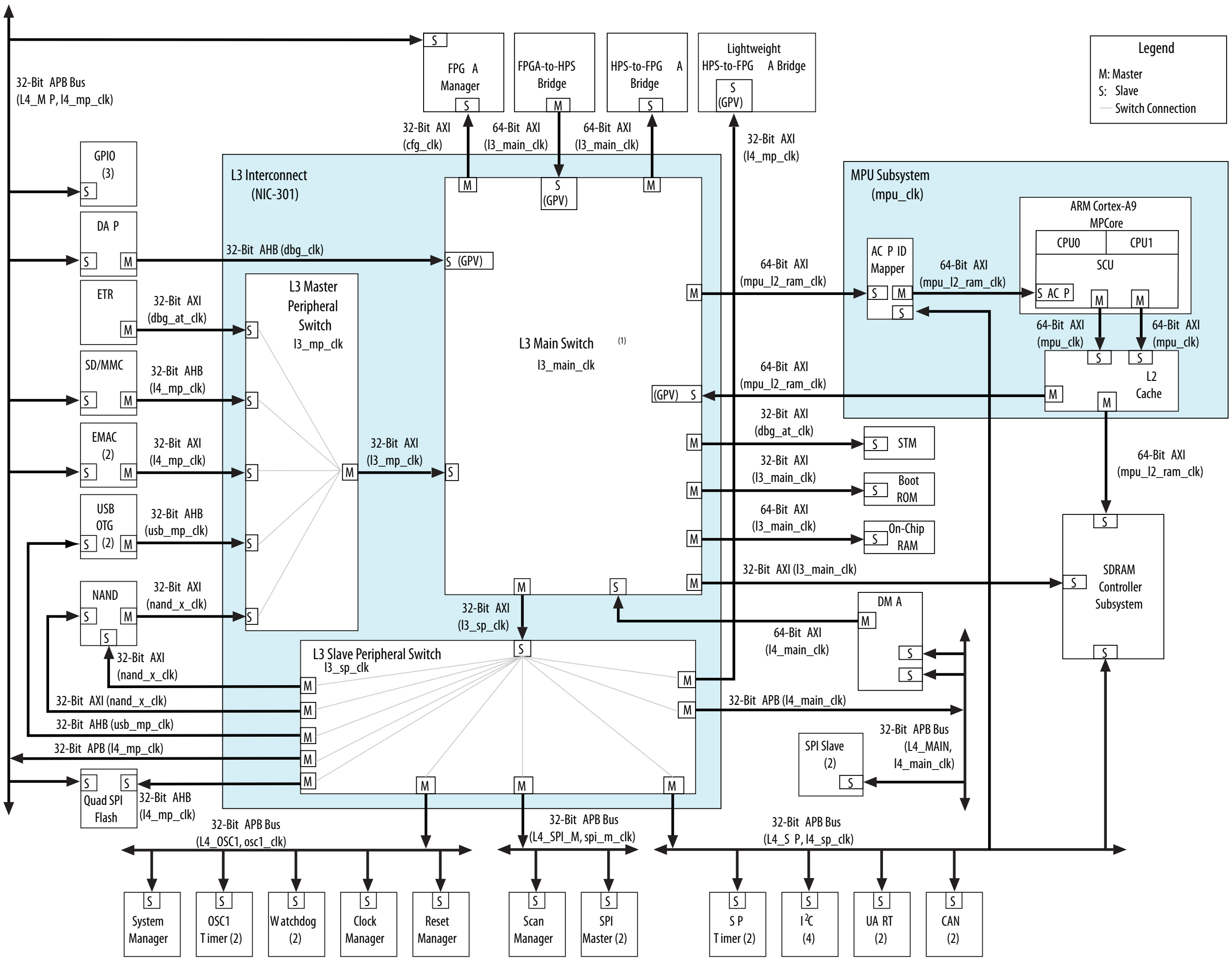
The L3 system interconnect supports high-throughput peripheral devices. The L3 interconnect has the following characteristics:

- Main internal data width of 64 bits
- Programmable master priority with single-cycle arbitration
- Full pipelining to prevent master stalls
- Programmable control for FIFO buffer transaction release
- Security of the following types:
  - Secure
  - Nonsecure
  - Per transaction security
- Five independent L4 buses

Interconnect Block Diagram and System Integration

Interconnect Block Diagram

The following figure shows the L3 interconnect and L4 buses.



Note: Interconnect slaves are available for connection from peripheral masters. Interconnect masters connect to peripheral slaves. This terminology is the reverse of conventional terminology used in Qsys.

Related Information

- Master to Slave Connectivity Matrix on page 7-4
- Main Connectivity Matrix on page 7-3

System Interconnect Architecture

The L3 interconnect is a partially-connected switch fabric. Not all masters can access all slaves.

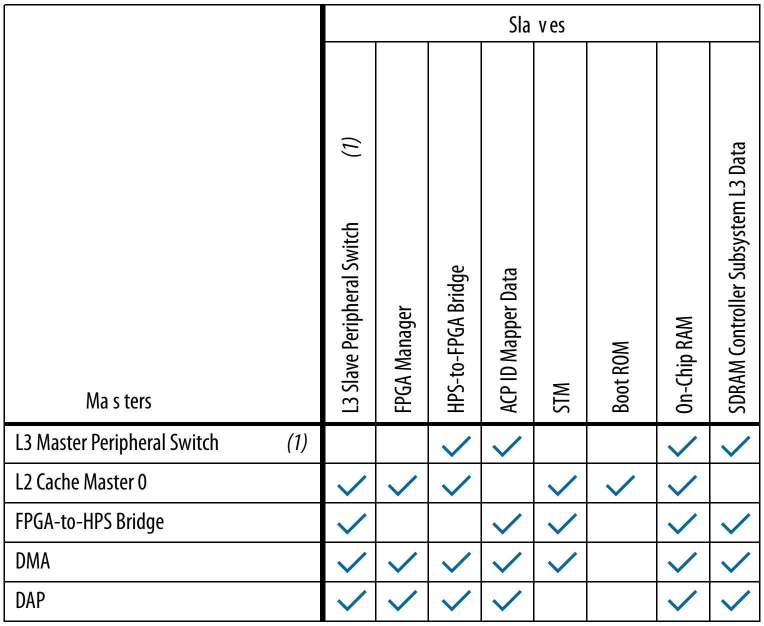
Internally, the L3 interconnect is partitioned into the following subswitches:

- L3 interconnect
  - Interconnect used to transfer high-throughput 64-bit data
  - Operates at up to half the MPU main clock frequency
  - Provides masters with low-latency connectivity to AXI bridges, on-chip memories, SDRAM, and FPGA manager
- L3 master peripheral switch
  - Used to connect memory-mastering peripherals to the interconnect
  - 32-bit data width
  - Operates at up to half the interconnect clock frequency
- L3 slave peripheral switch
  - Used to provide access to level 3 and 4 slave interfaces for masters of the master peripheral and interconnects
  - 32-bit data width
  - Five independent L4 buses

The L3 master and slave peripheral switches are fully-connected crossbars. The L3 interconnect is a partially-connected crossbar. The following table shows the connectivity matrix of all the master and slave interfaces of the L3 interconnect.

Main Connectivity Matrix

The L3 master and slave peripheral switches are fully-connected crossbars. The L3 interconnect is a partially-connected crossbar. The following table shows the connectivity matrix of all the master and slave interfaces of the L3 interconnect.



	Sla ves							
Ma s ters	L3 Slave Peripheral Switch (1)	FPGA Manager	HPS-to-FPGA Bridge	ACP ID Mapper Data	STM	Boot ROM	On-Chip RAM	SDRAM Controller Subsystem L3 Data
L3 Master Peripheral Switch (1)			✓	✓			✓	✓
L2 Cache Master 0	✓	✓	✓		✓	✓	✓	
FPGA-to-HPS Bridge	✓			✓	✓		✓	✓
DMA	✓	✓	✓	✓	✓		✓	✓
DAP	✓	✓	✓	✓			✓	✓

[8] For details of the masters and slaves connected to the L3 master peripheral switch and L3 master peripheral switch, refer to "Interconnect Block Diagram".

Related Information

Interconnect Block Diagram on page 7-2

Functional Description of the Interconnect

Master to Slave Connectivity Matrix

The interconnect is a partially-connected crossbar. The following table shows the connectivity matrix of all the master and slave interfaces of the interconnect.

Figure 7-1: Interconnect Connectivity Matrix

	Slaves																
Masters	L4 SP Bus Slaves	L4 MP Bus Slaves	L4 OSC1 Bus Slaves	L4 MAIN Bus Slaves	L4 SPIM Bus Slaves	Lightweight HPS-to-FPGA Bridge	USB OTG 0/1 CSR	NAND CSR	NAND Command and Data	Quad SPI Flash Data	FPGA Manager	HPS-to-FPGA Bridge	ACP ID Mapper Data	STM	Boot ROM	On-Chip RAM	SDRAM Controller Subsystem L3 Data
L2 Cache Master 0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	
FPGA-to-HPS Bridge	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓		✓	✓
DMA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
EMAC 0/1												✓	✓			✓	✓
USB OTG 0/1												✓	✓			✓	✓
NAND												✓	✓			✓	✓
SD/MMC												✓	✓			✓	✓
ETR												✓	✓			✓	✓
DAP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓

Address Remapping

The interconnect supports address remapping through the `remap` register in the `l3regs` group. Remapping allows software to control which memory device (SDRAM, on-chip RAM, or boot ROM) is accessible at address 0x0 and the accessibility of the HPS-to-FPGA and lightweight HPS-to-FPGA bridges. The `remap` register is one of the NIC-301 Global Programmers View (GPV) registers. The following L3 masters can manipulate `remap`, because it maps into their address space:

- MPU
- FPGA-to-HPS bridge
- DAP

The remapping bits in the `remap` register are not mutually exclusive. The lowest order remap bit has higher priority when multiple slaves are remapped to the same address. Each bit allows different combinations of address maps to be formed. There is only one remapping register available in the GPV, so modifying the `remap` register affects all memory maps of all the masters of the interconnect.

The effects of the remap bits can be categorized in the following groups:

- MPU master interface
    - L2 cache master 0 interface
- Non-MPU master interfaces
    - DMA master interface
    - Master peripheral interfaces
    - Debug Access Port (DAP) master interface
    - FPGA-to-HPS bridge master interface

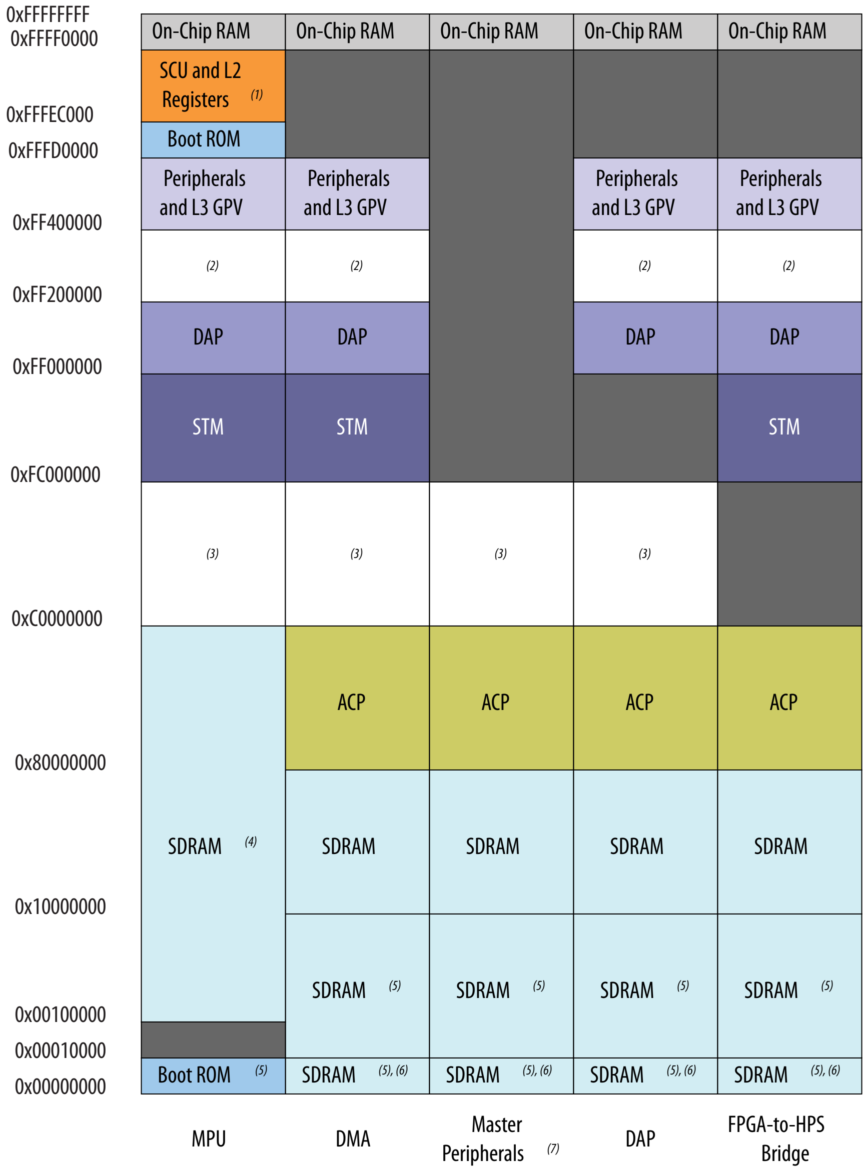
Related Information

- **L3 (NIC-301) GPV Registers Address Map** on page 7-17
  Information about the GPV registers
- **remap** on page 7-24
  Description of the `remap` register
- **Cortex-A9 Microprocessor Unit Subsystem** on page 9-1
  For general information about the MPU subsystem, refer to the Cortex-A9 Microprocessor Unit Subsystem chapter.
- **HPS Peripheral Master Input IDs** on page 9-29
  For information about virtual ID mapping in the ACP ID mapper, refer to "HPS Peripheral Master Input IDs" in the Cortex-A9 Microprocessor Unit Subsystem chapter.

Available Address Maps

Figure 7-2: Address Maps for Interconnect Masters

The following figure shows the default interconnect address maps for all masters. The figure is not to scale.



Notes on Address Maps

(1) Transactions on these addresses pass through the L2 interconnect.

(2) This region can be configured to access slaves on the lightweight HPS-to-FPGA bridge, by using the `remap` register.

(3) This region can be configured to access slaves on the HPS-to-FPGA bridge, by using the `remap` register.

(4) The MPU accesses SDRAM through a dedicated port.

[5] This region can be configured to access the ACP, by using the remap register.

[6] This region can be configured to access on-chip RAM, by using the remap register.

[7] The following peripherals can master the interconnect:

- Ethernet MACs
- USB-2 OTG controllers
- NAND controllers
- ETR
- SD/MMC controller

For the MPU L3 master, either the boot ROM or on-chip RAM maps to address 0x0 and obscures the lowest 64 KB of SDRAM. The address space from 0x00010000 to 0x00100000 is not accessible because the MPU L2 filter registers only have a granularity of 1 MB. After booting completes, the MPU can change address filtering to use the lowest 1 MB of SDRAM.

For non-MPU masters, either the on-chip RAM or the SDRAM maps to address 0x0. When mapped to address 0x0, the on-chip RAM obscures the lowest 64 KB of SDRAM for non-MPU masters.

Related Information

remap on page 7-24
Description of the remap register

Memory Map Remap Bits

Bit Name	Bit Offset	Description
mpuzero	0	When set to 0, the boot ROM maps to address 0x0 for the MPU L3 master. When set to 1, the on-chip RAM maps to address 0x0 for the MPU L3 master. This bit has no effect on non-MPU masters. Note that regardless of this setting, the boot ROM also always maps to address 0xffff0000 and the on-chip RAM also always maps to address 0xfffd0000 for the MPU L3 master.
nonmpuzero	1	When set to 0, the SDRAM maps to address 0x0 for the non-MPU L3 masters. When set to 1, the on-chip RAM maps to address 0x0 for the non-MPU masters. This bit has no effect on the MPU L3 master. Note that regardless of this setting, the on-chip RAM also always maps to address 0xfffd0000 for the non-MPU L3 masters.
Reserved	2	Must always be set to 0.
hps2fpga	3	When set to 1, the HPS-to-FPGA bridge slave port is visible to the L3 masters. When set to 0, accesses to the associated address range return an AXI decode error to the master.
lwhp2fpga	4	When set to 1, the lightweight HPS-to-FPGA bridge slave port is visible to the L3 masters. When set to 0, accesses to the associated address range return an AXI decode error to the master.

Bit Name	Bit Offset	Description
Reserved	31:5	Must always be set to 0.

Note: L2 filter registers in the MPU subsystem, not the interconnect, allow the SDRAM to be remapped to address `0x0` for the MPU.

Master Caching and Buffering Overrides

Some of the peripheral masters connected to the interconnect do not have the ability to drive the caching and buffering signals of their interfaces. The system manager provides registers so that you can enable cacheable and bufferable transactions for these masters. The system manager drives the caching and buffering signals of the following masters:

Master Peripheral	System Manager Register Group	Register
EMAC0 and EMAC1	`emacgrp`	`l3master`
USB OTG 0 and USB OTG 1	`usbgrp`	`l3master`
NAND flash	`nandgrp`	`l3master`
SD/MMC	`sdmmcgrp`	`l3master`

At reset time, the system manager drives the cache and buffering signals for these masters low. In other words, the masters listed do not support cacheable or bufferable accesses until you enable them after reset. There is no synchronization between the system manager and the interconnect, so avoid changing these settings when any of the masters are active.

Related Information

System Manager on page 5-1

For more information about enabling or disabling these features, refer to the *System Manager* chapter.

Security

Slave Security

The interconnect enforces security through the slave settings. The slave settings are controlled by the address region control registers accessible through the GPV registers. Each L3 and L4 slave has its own security check and programmable security settings. After reset, every slave of the interconnect is set to a secure state (referred to as boot secure). The only accesses allowed to secure slaves are by secure masters.

The GPV can only be accessed by secure masters. The security state of the interconnect is not accessible through the GPV as the security registers are write-only. Any nonsecure accesses to the GPV receive a `DECERR` response, and no register access is provided. Updates to the security settings through the GPV do not take effect until all transactions to the affected slave have completed.

Master Security

Masters of the interconnect are either secure, nonsecure, or the security is set on a per transaction basis. The DAP is capable of performing only secure accesses. The L2 cache master 0, FPGA-to-HPS-bridge, and DMA perform secure and nonsecure accesses on a per transaction basis. All other interconnect masters perform nonsecure accesses.

Accesses to secure slaves by unsecure masters result in a response of DECERR and the transaction does not reach the slave.

Related Information

Interconnect Master Properties on page 7-10

Arbitration

At the entry point to the interconnect, all transactions are allocated a local quality of service (QoS). QoS specifies the transaction's arbitration priority. The interconnect allows transactions with a higher QoS to use a greater share of interconnect bandwidth. The transaction arbitration throughout the interconnect uses this QoS value. The QoS controls for each master connected to the interconnect are separated into read and write QoS priority values.

At any arbitration node, a fixed priority exists for transactions with different QoS values. The highest QoS value has the highest priority. If there are coincident transactions at an arbitration node with the same QoS value that require arbitration, then the interconnect uses a least recently used (LRU) algorithm.

You can programmatically configure the QoS value for each master through the appropriate `write_qos` register.

Related Information

L3 (NIC-301) GPV Registers Address Map on page 7-17
Information about the GPV `write_qos` registers

Cyclic Dependency Avoidance Schemes

The AXI protocol permits re-ordering of transactions. As a result, when routing concurrent multiple transactions from a single point of divergence to multiple slaves, the interconnect might need to enforce rules to prevent deadlock.

Each master of the interconnect is configured with one of three possible cyclic dependency avoidance schemes (CDAS). The same CDAS scheme is configured for both read and write transactions, but they operate independently.

Single Slave on page 7-9

Single Slave Per ID on page 7-10

Single Active Slave on page 7-10

Related Information

Interconnect Master Properties on page 7-10
Contains descriptions of the CDAS implementation for the masters.

Single Slave

Single slave (SS) ensures the following conditions at a slave interface of a switch:

- All outstanding read transactions are to a single end destination.
- All outstanding write transactions are to a single end destination.

If a master issues another transaction to a different destination than the current destination for that transaction type (read or write), the network stalls the transactions until all the outstanding transactions of that type have completed.

Single Slave Per ID

Single slave per ID (SSPID) ensures the following conditions at a slave interface of a switch:

- All outstanding read transactions with the same ID go the same destination.
- All outstanding write transactions with the same ID go the same destination.

When a master issues a transaction, the following situations can occur:

- If the transaction has an ID that does not match any outstanding transactions, it passes the CDAS.
- If the transaction has an ID that matches the ID of an outstanding transaction, and the destinations also match, it passes the CDAS.
- If the transaction has an ID that matches the ID of an outstanding transaction, and the destinations do not match, the transaction fails the CDAS check and stalls.

Single Active Slave

Single active slave (SAS) is the same as the SSPID scheme, with an added check for write transactions. SAS ensures that a master cannot issue a new write address until all of the data from the previous write transaction has been sent.

Interconnect Master Properties

The system interconnect connects to various slave interfaces through the L3 interconnect and L3 slave peripheral switch.

Table 7-1: Interconnect Master Interfaces

TrustZone security:

- Secure: All transactions are marked TrustZone secure
- Nonsecure: All transactions are marked TrustZone non-secure
- Per transaction: Transactions can be marked TrustZone secure or TrustZone non-secure, depending on the state of the interconnect master.

Issuance is based on the number of read, write, and total transactions.

The FIFO buffer depth for AXI is based on the AW, AR, R, W, and B channels. For AHB and APB, the depth is based on W, A, and D channels.

Master	Interface Width	Clock	Switch	TrustZone Security	GPV Access	CDAS	Issuance	FIFO Buffer Depth	Type
L2 cache M0	64	`mpu_l2_ram_clk`	L3 interconnect	Per Transaction	Yes	SSPID	7, 12, 19	2, 2, 2, 2, 2	AXI
FPGA-to-HPS bridge	64	`l3_main_clk`	L3 interconnect	Per Transaction	Yes	SAS	16, 16, 32	2, 2, 6, 6, 2	AXI
DMA	64	`l4_main_clk`	L3 interconnect	Per Transaction	No	SSPID	8, 8, 8	2, 2, 2, 2, 2	AXI

Master	Interface Width	Clock	Switch	TrustZone Security	GPV Access	CDAS	Issuance	FIFO Buffer Depth	Type
EMAC 0/1	32	`l4_main_clk`	L3 master peripheral switch	Secure	No	SSPID	16, 16, 32	2, 2, 2, 2, 2	AXI
USB OTG 0/1	32	`usb_mp_clk`	L3 master peripheral switch	Nonsecure	No	SSPID	2, 2, 4	2, 2, 2	AHB
NAND	32	`nand_x_clk`	L3 master peripheral switch	Nonsecure	No	SSPID	1, 8, 9	2, 2, 2, 2, 2	AXI
SD/MMC	32	`l4_mp_clk`	L3 master peripheral switch	Nonsecure	No	SSPID	2, 2, 4	2, 2, 2	AHB
ETR	32	`dbg_at_clk`	L3 master peripheral switch	Per Transaction	No	SSPID	32, 1, 32	2, 2, 2, 2, 2	AXI
DAP	32	`dbg_clk`	L3 interconnect	Secure	Yes	SS	1, 1, 1	2, 2, 2	AHB

Interconnect Slave Properties

The interconnect connects to various slave interfaces through the L3 interconnect, L3 slave peripheral switch, and the five L4 peripheral buses. After reset, all slave interfaces are set to the secure state.

Table 7-2: Interconnect Slave Interfaces

Acceptance is based on the number of read, write, and total transactions. The FIFO buffer depth for AXI is based on the AW, AR, R, W, and B channels. For AHB and APB, the depth is based on the W, A, and D channels.

Slave	Interface Width	Clock	Mastered By	Acceptance[9]	Buffer Depth [10]	Type
SDRAM subsystem CSR	32	l4_sp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
SP timer 0/1	32	l4_sp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
I^2C 0/1/2/3	32	l4_sp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
UART 0/1	32	l4_sp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
CAN 0/1	32	l4_sp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
GPIO 0/1/2	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
ACP ID mapper CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
FPGA manager CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
DAP CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
Quad SPI flash CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
SD/MMC CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
EMAC 0/1 CSR	32	l4_mp_clk	L4 SP bus master	1, 1, 1	2, 2, 2	APB
System manager	32	osc1_clk	L4 OSC1 bus master	1, 1, 1	2, 2, 2	APB
OSC1 timer 0/1	32	osc1_clk	L4 OSC1 bus master	1, 1, 1	2, 2, 2	APB
Watchdog 0/1	32	osc1_clk	L4 OSC1 bus master	1, 1, 1	2, 2, 2	APB
Clock manager	32	osc1_clk	L4 OSC1 bus master	1, 1, 1	2, 2, 2	APB
Reset manager	32	osc1_clk	L4 OSC1 bus master	1, 1, 1	2, 2, 2	APB
DMA secure CSR	32	l4_main_clk	L4 main bus master	1, 1, 1	2, 2, 2	APB

[9] Acceptance is based on the number of read, write, and total transactions.

Slave	Interface Width	Clock	Mastered By	Acceptance[9]	Buffer Depth [10]	Type
DMA nonsecure CSR	32	l4_main_clk	L4 main bus master	1, 1, 1	2, 2, 2	APB
SPI slave 0/1	32	l4_main_clk	L4 main bus master	1, 1, 1	2, 2, 2	APB
Scan manager	32	spi_m_clk	L4 main bus master	1, 1, 1	2, 2, 2	APB
SPI master 0/1	32	spi_m_clk	L4 main bus master	1, 1, 1	2, 2, 2	APB
Lightweight HPS-to-FPGA bridge	32	l4_main_clk	L3 slave peripheral switch	16, 16, 32	2, 2, 2, 2, 2	AXI
USB OTG 0/1	32	usb_mp_clk	L3 slave peripheral switch	1, 1, 1	2, 2, 2	AHB
NAND CSR	32	nand_x_clk	L3 slave peripheral switch	1, 1, 1	2, 2, 2	AXI
NAND command and data	32	nand_x_clk	L3 slave peripheral switch	1, 1, 1	2, 2, 2	AXI
Quad SPI flash data	32	l4_mp_clk	L3 slave peripheral switch	1, 1, 1	2, 2, 2	AHB
FPGA manager data	32	cfg_clk	L3 interconnect	1, 2, 3	2, 2, 2, 32, 2	AXI
HPS-to-FPGA bridge	64	l3_main_clk	L3 interconnect	16, 16, 32	2, 2, 6, 6, 2	AXI
ACP ID mapper data	64	mpu_l2_ram_clk	L3 interconnect	13, 5, 18	2, 2, 2, 2, 2	AXI
STM	32	dbg_at_clk	L3 interconnect	1, 2, 2	2, 2, 2, 2, 2	AXI
On-chip boot ROM	32	l3_main_clk	L3 interconnect	1, 1, 2	0, 0, 0, 0, 0	AXI

[9] Acceptance is based on the number of read, write, and total transactions.

[10] The FIFO buffer depth for AXI is based on the AW, AR, R, W, and B channels. For AHB and APB, the depth is based on W, A, and D channels.

Slave	Interface Width	Clock	Mastered By	Acceptance[9]	Buffer Depth[10]	Type
On-chip RAM	64	`l3_main_clk`	L3 interconnect	2, 2, 2	0, 0, 0, 8, 0	AXI
SDRAM subsystem L3 data	32	`l3_main_clk`	L3 interconnect	16, 16, 16	2, 2, 2, 2, 2	AXI

Upsizing Data Width Function

The upsizing function combines narrow transactions into wider transactions to increase the overall system bandwidth. Upsizing only packs data for read or write transactions that are cacheable. If the interconnect splits input-exclusive transactions into more than one output bus transaction, it removes the exclusive information from the multiple transactions it creates.

The upsizing function can expand the data width by the following ratios:

- 1:2
- 1:4

If multiple responses from created transactions are combined into one response, then the following order of priority applies:

- `DECERR` is the highest priority
- `SLVERR` is the next highest priority
- `OKAY` is the lowest priority.

Related Information

http://infocenter.arm.com/

Additional information is available in the *AMBA Network Interconnect (NIC-301) Technical Reference Manual*, revision r2p3, which you can download from the ARM Infocenter website.

Incrementing Bursts

The interconnect converts all input `INCR` bursts that complete within a single output data width to an `INCR1` burst of the minimum `SIZE` possible, and packs all `INCR` bursts into `INCR` bursts of the optimal size possible for maximum data throughout.

Wrapping Bursts

All `WRAP` bursts are either passed through unconverted as `WRAP` bursts, or converted to one or two `INCR` bursts of the output bus. The interconnect converts input `WRAP` bursts that have a total payload less than the output data width to a single `INCR` burst.

[9] Acceptance is based on the number of read, write, and total transactions.

[10] The FIFO buffer depth for AXI is based on the AW, AR, R, W, and B channels. For AHB and APB, the depth is based on W, A, and D channels.

[10] The FIFO buffer depth for AXI is based on the AW, AR, R, W, and B channels. For AHB and APB, the depth is based on W, A, and D channels.

Fixed Bursts

All FIXED bursts pass through unconverted.

Bypass Merge

Bypass merge is accessible through the GPV registers and is only accessible to secure masters. If the programmable bit bypass_merge is enabled, the interconnect does not alter any transactions that could pass through legally without alteration.

Downsizing Data Width Function

The downsizing function reduces the data width of a transaction to match the optimal data width at the destination. Downsizing does not merge multiple-transaction data narrower than the destination bus if the transactions are marked as noncacheable.

The downsizing function reduces the data width by the following ratios:

- 2:1
- 4:1

Incrementing Bursts

The interconnect converts INCR bursts that fall within the maximum payload size of the output data bus to a single INCR burst. It converts INCR bursts that are greater than the maximum payload size of the output data bus to multiple INCR bursts.

INCR bursts with a size that matches the output data width pass through unconverted.

The interconnect packs INCR bursts with a SIZE smaller than the output data width to match the output width whenever possible, using the upsizing function.

Related Information

Upsizing Data Width Function on page 7-14

For more information about AXI terms such as DECERR, WRAP, and INCR, refer to the *AMBA AXI Protocol Specification v1.0*, which you can download from the ARM website.

Wrapping Bursts

The interconnect always converts WRAP bursts to WRAP bursts of twice the length, up to the output data width maximum size of WRAP16, and treats the WRAP burst as two INCR bursts that can each be converted into one or more INCR bursts.

Fixed Bursts

The interconnect converts FIXED bursts to one or more INCR1 or INCRn bursts depending on the downsize ratio.

Bypass Merge

Bypass merge is accessible through the GPV registers and is only accessible to secure masters. If the programmable bit bypass_merge in the fn_mod2 register is enabled, the interconnect does not perform any packing of beats to match the optimal size for maximum throughput, up to the output data width size.

If an exclusive transaction is split into multiple transactions at the output of the downsizing function, the exclusive flag is removed and the master never receives an EXOKAY response. Response priority is the same as for the upsizing function.

Related Information

- **fn_mod2** on page 7-80
- **Upsizing Data Width Function** on page 7-14
  For more information about AXI terms such as DECERR, WRAP, and INCR, refer to the *AMBA AXI Protocol Specification v1.0*, which you can download from the ARM website.

Lock Support

Lock is not supported by the interconnect. For atomic accesses, masters can perform exclusive accesses when sharing data located in the HPS SDRAM.

Related Information

Functional Description—HPS Memory Controller on page 11-1
For more information about exclusive access support, refer to the *SDRAM Controller Subsystem* chapter.

FIFO Buffers and Clocks

The interconnect contains FIFO buffers in the majority of the interfaces exposed to the HPS master and slaves, as well as between the subswitches. These FIFO buffers also provide clock domain crossing for masters and slaves that operate at a different clock frequency than the switch they connect to.

Data Release Mechanism

For interconnect ports with data FIFO buffers whose depth is greater than zero, you can set a write tidemark function, `wr_tidemark`. This tidemark level stalls the release of the transaction until one of the following situations occurs:

- The interconnect receives the `WLAST` beat of a burst.
- The write data FIFO buffer becomes full.
- The number of occupied slots in the write data FIFO buffer exceeds the write tidemark.

Related Information

- **Interconnect Master Properties** on page 7-10
  Indicates which master interfaces have data FIFO buffers with a nonzero depth
- **Interconnect Slave Properties** on page 7-11
  Indicates which slave interfaces have data FIFO buffers with a nonzero depth

Resets

The interconnect has one reset signal. The reset manager drives this signal to the interconnect on a cold or warm reset. On reset, the boot ROM is mapped to address 0x0.

Related Information

Reset Manager on page 3-1

Interconnect Address Map and Register Definitions

This section lists the interconnect register address map and describes the registers.

Note: Interconnect slaves are available for connection from peripheral masters. Interconnect masters connect to peripheral slaves. This terminology is the reverse of conventional terminology used in Qsys.

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor* chapter.
- **Cyclone V Address Map and Register Definitions**
  Web-based address map and register definitions

L3 (NIC-301) GPV Registers Address Map

Registers to control L3 interconnect settings

Base Address: `0xFF800000`

L3 (NIC-301) GPV Registers

Register	Offset	Width	Acces s	Reset Value	Description
remap on page 7-24	0x0	32	WO	0x0	Remap

Security Register Group

Register	Offset	Width	Acces s	Reset Value	Description
l4main on page 7-27	0x8	32	WO	0x0	L4 main peripherals security
l4sp on page 7-29	0xC	32	WO	0x0	L4 SP Peripherals Security
l4mp on page 7-32	0x10	32	WO	0x0	L4 MP Peripherals Security
l4osc1 on page 7-35	0x14	32	WO	0x0	L4 OSC1 Peripherals Security
l4spim on page 7-37	0x18	32	WO	0x0	L4 SPIM Peripherals Security
stm on page 7-38	0x1C	32	WO	0x0	STM Peripheral Security
lwhps2fpgaregs on page 7-39	0x20	32	WO	0x0	LWHPS2FPGA AXI Bridge Registers Peripheral Security
usb1 on page 7-40	0x28	32	WO	0x0	USB1 Registers Peripheral Security
nanddata on page 7-40	0x2C	32	WO	0x0	NAND Flash Controller Data Peripheral Security
usb0 on page 7-41	0x80	32	WO	0x0	USB0 Registers Peripheral Security
nandregs on page 7-42	0x84	32	WO	0x0	NAND Flash Controller Registers Peripheral Security
qspidata on page 7-42	0x88	32	WO	0x0	QSPI Flash Controller Data Peripheral Security
fpgamgrdata on page 7-43	0x8C	32	WO	0x0	FPGA Manager Data Peripheral Security

Register	Offset	Width	Acces s	Reset Value	Description
hps2fpgaregs on page 7-44	0x90	32	WO	0x0	HPS2FPGA AXI Bridge Registers Peripheral Security
acp on page 7-44	0x94	32	WO	0x0	MPU ACP Peripheral Security
rom on page 7-45	0x98	32	WO	0x0	ROM Peripheral Security
ocram on page 7-46	0x9C	32	WO	0x0	On-chip RAM Peripheral Security
sdrdata on page 7-46	0xA0	32	WO	0x0	SDRAM Data Peripheral Security

ID Register Group

Register	Offset	Width	Acces s	Reset Value	Description
periph_id_4 on page 7-48	0x1FD0	32	RO	0x4	Peripheral ID4 Register
periph_id_0 on page 7-48	0x1FE0	32	RO	0x1	Peripheral ID0 Register
periph_id_1 on page 7-49	0x1FE4	32	RO	0xB3	Peripheral ID1 Register
periph_id_2 on page 7-49	0x1FE8	32	RO	0x6B	Peripheral ID2 Register
periph_id_3 on page 7-49	0x1FEC	32	RO	0x0	Peripheral ID3 Register
comp_id_0 on page 7-50	0x1FF0	32	RO	0xD	Component ID0 Register
comp_id_1 on page 7-50	0x1FF4	32	RO	0xF0	Component ID1 Register
comp_id_2 on page 7-51	0x1FF8	32	RO	0x5	Component ID2 Register
comp_id_3 on page 7-51	0x1FFC	32	RO	0xB1	Component ID3 Register

L4 MAIN

Register	Offset	Width	Acces s	Reset Value	Description
fn_mod_bm_iss on page 7-52	0x2008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register

L4 SP

Register	Offset	Width	Acces s	Reset Value	Description
fn_mod_bm_iss on page 7-53	0x3008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register

L4 MP

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-54	0x4008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register

L4 OSC1

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-55	0x5008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register

L4 SPIM

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-56	0x6008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register

STM

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-57	0x7008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-58	0x7108	32	RW	0x0	Issuing Functionality Modification Register

LWHPS2FPGA

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-59	0x8008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-59	0x8108	32	RW	0x0	Issuing Functionality Modification Register

USB1

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-60	0xA008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
ahb_cntl on page 7-61	0xA044	32	RW	0x0	AHB Control Register

NANDDATA

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-62	0xB008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-63	0xB108	32	RW	0x0	Issuing Functionality Modification Register

USB0

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-64	0x20008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
ahb_cntl on page 7-65	0x20044	32	RW	0x0	AHB Control Register

NANDREGS

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-66	0x21008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-67	0x21108	32	RW	0x0	Issuing Functionality Modification Register

QSPIDATA

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-68	0x22008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
ahb_cntl on page 7-68	0x22044	32	RW	0x0	AHB Control Register

FPGAMGRDATA

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-70	0x23008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
wr_tidemark on page 7-70	0x23040	32	RW	0x4	Write Tidemark
fn_mod on page 7-71	0x23108	32	RW	0x0	Issuing Functionality Modification Register

HPS2FPGA

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-72	0x24008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
wr_tidemark on page 7-73	0x24040	32	RW	0x4	Write Tidemark
fn_mod on page 7-73	0x24108	32	RW	0x0	Issuing Functionality Modification Register

ACP

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-74	0x25008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-75	0x25108	32	RW	0x0	Issuing Functionality Modification Register

Boot ROM

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-76	0x26008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
fn_mod on page 7-76	0x26108	32	RW	0x0	Issuing Functionality Modification Register

On-chip RAM

Register	Offset	Width	Access	Reset Value	Description
fn_mod_bm_iss on page 7-77	0x27008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
wr_tidemark on page 7-78	0x27040	32	RW	0x4	Write Tidemark
fn_mod on page 7-79	0x27108	32	RW	0x0	Issuing Functionality Modification Register

DAP

Register	Offset	Width	Access	Reset Value	Description
fn_mod2 on page 7-80	0x42024	32	RW	0x0	Functionality Modification 2 Register

Register	Offset	Width	Access	Reset Value	Description
fn_mod_ahb on page 7-81	0x42028	32	RW	0x0	Functionality Modification AHB Register
read_qos on page 7-82	0x42100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-83	0x42104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-83	0x42108	32	RW	0x0	Issuing Functionality Modification Register

MPU

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-84	0x43100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-85	0x43104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-85	0x43108	32	RW	0x0	Issuing Functionality Modification Register

SDMMC

Register	Offset	Width	Access	Reset Value	Description
fn_mod_ahb on page 7-86	0x44028	32	RW	0x0	Functionality Modification AHB Register
read_qos on page 7-87	0x44100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-88	0x44104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-88	0x44108	32	RW	0x0	Issuing Functionality Modification Register

DMA

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-89	0x45100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-90	0x45104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-90	0x45108	32	RW	0x0	Issuing Functionality Modification Register

FPGA2HPS

Register	Offset	Width	Access	Reset Value	Description
wr_tidemark on page 7-91	0x46040	32	RW	0x4	Write Tidemark
read_qos on page 7-92	0x46100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-92	0x46104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-93	0x46108	32	RW	0x0	Issuing Functionality Modification Register

ETR

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-94	0x47100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-94	0x47104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-95	0x47108	32	RW	0x0	Issuing Functionality Modification Register

EMAC0

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-96	0x48100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-96	0x48104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-97	0x48108	32	RW	0x0	Issuing Functionality Modification Register

EMAC1

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-98	0x49100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-98	0x49104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-99	0x49108	32	RW	0x0	Issuing Functionality Modification Register

USB0

Register	Offset	Width	Access	Reset Value	Description
fn_mod_ahb on page 7-100	0x4A028	32	RW	0x0	Functionality Modification AHB Register

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-101	0x4A100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-102	0x4A104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-102	0x4A108	32	RW	0x0	Issuing Functionality Modification Register

NAND

Register	Offset	Width	Access	Reset Value	Description
read_qos on page 7-103	0x4B100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-104	0x4B104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-104	0x4B108	32	RW	0x0	Issuing Functionality Modification Register

USB1

Register	Offset	Width	Access	Reset Value	Description
fn_mod_ahb on page 7-105	0x4C028	32	RW	0x0	Functionality Modification AHB Register
read_qos on page 7-106	0x4C100	32	RW	0x0	Read Channel QoS Value
write_qos on page 7-107	0x4C104	32	RW	0x0	Write Channel QoS Value
fn_mod on page 7-107	0x4C108	32	RW	0x0	Issuing Functionality Modification Register

remap

The L3 interconnect has separate address maps for the various L3 Masters. Generally, the addresses are the same for most masters. However, the sparse interconnect of the L3 switch causes some masters to have holes in their memory maps. The remap bits are not mutually exclusive. Each bit can be set independently and in combinations. Priority for the bits is determined by the bit offset: lower offset bits take precedence over higher offset bits.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800000

Offset: 0x0

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											lwhps2fpga WO 0x0	hps2fpga WO 0x0	Reserved	nonmpuzero WO 0x0	mpuzero WO 0x0

remap Fields

Bit	Name	Description	Access	Reset
4	lwhps2fpga	Controls whether the Lightweight HPS2FPGA AXI Bridge is visible to L3 masters or not. **Value** **Description** 0x0 The LWHPS2FPGA AXI Bridge is not visible to L3 masters. Accesses to the associated address range return an AXI decode error to the master. 0x1 The LWHPS2FPGA AXI Bridge is visible to L3 masters.	WO	0x0
3	hps2fpga	Controls whether the HPS2FPGA AXI Bridge is visible to L3 masters or not. **Value** **Description** 0x0 The HPS2FPGA AXI Bridge is not visible to L3 masters. Accesses to the associated address range return an AXI decode error to the master. 0x1 The HPS2FPGA AXI Bridge is visible to L3 masters.	WO	0x0

Bit	Name	Description	Access	Reset
1	`nonmpuzero`	Controls the mapping of address 0x0 for L3 masters other than the MPU. Determines whether address 0x0 for these masters is mapped to the SDRAM or on-chip RAM. Only affects the following masters: DMA controllers (standalone and those built in to peripherals), FPGA-to-HPS Bridge, and DAP. **Value** **Description** 0x0 Maps the SDRAM to address 0x0 for the non-MPU L3 masters. 0x1 Maps the On-chip RAM to address 0x0 for the non-MPU L3 masters. Note that the On-chip RAM is also always mapped to address 0xffff_0000 for the non-MPU L3 masters independent of this field's value.	WO	0x0
0	`mpuzero`	Controls whether address 0x0 for the MPU L3 master is mapped to the Boot ROM or On-chip RAM. This field only has an effect on the MPU L3 master. **Value** **Description** 0x0 Maps the Boot ROM to address 0x0 for the MPU L3 master. Note that the Boot ROM is also always mapped to address 0xfffd_0000 for the MPU L3 master independent of this field's value. 0x1 Maps the On-chip RAM to address 0x0 for the MPU L3 master. Note that the On-chip RAM is also always mapped to address 0xffff_0000 for the MPU L3 master independent of this field's value.	WO	0x0

Security Register Group Register Descriptions

Registers that control slave security.

Offset: `0x8`

l4main on page 7-27
Controls security settings for L4 main peripherals

l4sp on page 7-29
Controls security settings for L4 SP peripherals.

l4mp on page 7-32
Controls security settings for L4 MP peripherals.

l4osc1 on page 7-35
Controls security settings for L4 OSC1 peripherals.

l4spim on page 7-37
Controls security settings for L4 SPIM peripherals.

stm on page 7-38
Controls security settings for STM peripheral.

lwhps2fpgaregs on page 7-39
Controls security settings for LWHPS2FPGA AXI Bridge Registers peripheral.

usb1 on page 7-40
Controls security settings for USB1 Registers peripheral.

nanddata on page 7-40
Controls security settings for NAND Flash Controller Data peripheral.

usb0 on page 7-41
Controls security settings for USB0 Registers peripheral.

nandregs on page 7-42
Controls security settings for NAND Flash Controller Registers peripheral.

qspidata on page 7-42
Controls security settings for QSPI Flash Controller Data peripheral.

fpgamgrdata on page 7-43
Controls security settings for FPGA Manager Data peripheral.

hps2fpgaregs on page 7-44
Controls security settings for HPS2FPGA AXI Bridge Registers peripheral.

acp on page 7-44
Controls security settings for MPU ACP peripheral.

rom on page 7-45
Controls security settings for ROM peripheral.

ocram on page 7-46
Controls security settings for On-chip RAM peripheral.

sdrdata on page 7-46
Controls security settings for SDRAM Data peripheral.

l4main

Controls security settings for L4 main peripherals

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF800008`

Offset: `0x8`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												dmanonsecure WO 0x0	dmasecure WO 0x0	spis1 WO 0x0	spis0 WO 0x0

l4main Fields

Bit	Name	Description	Access	Reset
3	dmanonsecure	Controls whether secure or non-secure masters can access the DMA Non-secure slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
2	dmasecure	Controls whether secure or non-secure masters can access the DMA Secure slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
1	spis1	Controls whether secure or non-secure masters can access the SPI Slave 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
0	spis0	Controls whether secure or non-secure masters can access the SPI Slave 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

l4sp

Controls security settings for L4 SP peripherals.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80000C

Offset: 0xC

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					sptimer1 WO 0x0	can1 WO 0x0	can0 WO 0x0	uart1 WO 0x0	uart0 WO 0x0	i2c3 WO 0x0	i2c2 WO 0x0	i2c1 WO 0x0	i2c0 WO 0x0	sptimer0 WO 0x0	sdrregs WO 0x0

l4sp Fields

Bit	Name	Description	Access	Reset
10	sptimer1	Controls whether secure or non-secure masters can access the SP Timer 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
9	`can1`	Controls whether secure or non-secure masters can access the CAN 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
8	`can0`	Controls whether secure or non-secure masters can access the CAN 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
7	`uart1`	Controls whether secure or non-secure masters can access the UART 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
6	`uart0`	Controls whether secure or non-secure masters can access the UART 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
5	i2c3	Controls whether secure or non-secure masters can access the I2C3 (EMAC 1) slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
4	i2c2	Controls whether secure or non-secure masters can access the I2C2 (EMAC 0) slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
3	i2c1	Controls whether secure or non-secure masters can access the I2C1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
2	i2c0	Controls whether secure or non-secure masters can access the I2C0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
1	sptimer0	Controls whether secure or non-secure masters can access the SP Timer 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
0	sdrregs	Controls whether secure or non-secure masters can access the SDRAM Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

l4mp

Controls security settings for L4 MP peripherals.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800010

Offset: 0x10

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						gpio2 WO 0x0	gpio1 WO 0x0	gpio0 WO 0x0	acpidmap WO 0x0	emac1 WO 0x0	emac0 WO 0x0	sdmmc WO 0x0	qspiregs WO 0x0	dap WO 0x0	fpgamgrregs WO 0x0

l4mp Fields

Bit	Name	Description	Access	Reset
9	gpio2	Controls whether secure or non-secure masters can access the GPIO 2 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
8	gpio1	Controls whether secure or non-secure masters can access the GPIO 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
7	gpio0	Controls whether secure or non-secure masters can access the GPIO 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
6	acpidmap	Controls whether secure or non-secure masters can access the ACP ID Mapper slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
5	emac1	Controls whether secure or non-secure masters can access the EMAC 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
4	emac0	Controls whether secure or non-secure masters can access the EMAC 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
3	sdmmc	Controls whether secure or non-secure masters can access the SDMMC slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
2	qspiregs	Controls whether secure or non-secure masters can access the QSPI Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

Bit	Name	Description	Access	Reset
1	`dap`	Controls whether secure or non-secure masters can access the DAP slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
0	`fpgamgrregs`	Controls whether secure or non-secure masters can access the FPGA Manager Register slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

l4osc1

Controls security settings for L4 OSC1 peripherals.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF800014`

Offset: `0x14`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									osc1timer1 WO 0x0	osc1timer0 WO 0x0	sysmgr WO 0x0	rstmgr WO 0x0	clkmgr WO 0x0	l4wd1 WO 0x0	l4wd0 WO 0x0

l4osc1 Fields

Bit	Name	Description	Access	Reset
6	`osc1timer1`	Controls whether secure or non-secure masters can access the OSC1 Timer 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`
5	`osc1timer0`	Controls whether secure or non-secure masters can access the OSC1 Timer 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`
4	`sysmgr`	Controls whether secure or non-secure masters can access the System Manager slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`
3	`rstmgr`	Controls whether secure or non-secure masters can access the Reset Manager slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`

Bit	Name	Description	Access	Reset
2	`clkmgr`	Controls whether secure or non-secure masters can access the Clock Manager slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`
1	`l4wd1`	Controls whether secure or non-secure masters can access the L4 Watchdog Timer 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`
0	`l4wd0`	Controls whether secure or non-secure masters can access the L4 Watchdog Timer 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	`WO`	`0x0`

l4spim

Controls security settings for L4 SPIM peripherals.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF800018`

Offset: `0x18`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													scanmgr WO 0x0	spim1 WO 0x0	spim0 WO 0x0

l4spim Fields

Bit	Name	Description	Access	Reset
2	scanmgr	Controls whether secure or non-secure masters can access the Scan Manager slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
1	spim1	Controls whether secure or non-secure masters can access the SPI Master 1 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0
0	spim0	Controls whether secure or non-secure masters can access the SPI Master 0 slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

stm

Controls security settings for STM peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80001C

Offset: `0x1C`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

stm Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the STM slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

lwhps2fpgaregs

Controls security settings for LWHPS2FPGA AXI Bridge Registers peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800020

Offset: `0x20`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

lwhps2fpgaregs Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the LWHPS2FPGA AXI Bridge Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

usb1

Controls security settings for USB1 Registers peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800028

Offset: 0x28

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

usb1 Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the USB1 Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

nanddata

Controls security settings for NAND Flash Controller Data peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80002C

Offset: `0x2C`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

nanddata Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the NAND Flash Controller Data slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

usb0

Controls security settings for USB0 Registers peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800080

Offset: `0x80`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

usb0 Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the USB0 Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

nandregs

Controls security settings for NAND Flash Controller Registers peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800084

Offset: 0x84

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

nandregs Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the NAND Flash Controller Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

qspidata

Controls security settings for QSPI Flash Controller Data peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800088

Offset: `0x88`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

qspidata Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the QSPI Flash Controller Data slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

fpgamgrdata

Controls security settings for FPGA Manager Data peripheral.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF80008C`

Offset: `0x8C`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

fpgamgrdata Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the FPGA Manager Data slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

hps2fpgaregs

Controls security settings for HPS2FPGA AXI Bridge Registers peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800090

Offset: 0x90

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

hps2fpgaregs Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the HPS2FPGA AXI Bridge Registers slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

acp

Controls security settings for MPU ACP peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800094

Offset: 0x94

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

acp Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the MPU ACP slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

rom

Controls security settings for ROM peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF800098

Offset: 0x98

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

rom Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the ROM slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

ocram

Controls security settings for On-chip RAM peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80009C

Offset: `0x9C`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

ocram Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the On-chip RAM slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

sdrdata

Controls security settings for SDRAM Data peripheral.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF8000A0

Offset: `0xA0`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															s WO 0x0

sdrdata Fields

Bit	Name	Description	Access	Reset
0	s	Controls whether secure or non-secure masters can access the SDRAM Data slave. **Value** **Description** 0x0 The slave can only be accessed by a secure master. 0x1 The slave can only be accessed by a secure or non-secure masters.	WO	0x0

ID Register Group Register Descriptions

Contains registers that identify the ARM NIC-301 IP Core.

Offset: `0x1000`

periph_id_4 on page 7-48
JEP106 continuation code

periph_id_0 on page 7-48
Peripheral ID0

periph_id_1 on page 7-49
Peripheral ID1

periph_id_2 on page 7-49
Peripheral ID2

periph_id_3 on page 7-49
Peripheral ID3

comp_id_0 on page 7-50
Component ID0

comp_id_1 on page 7-50
Component ID1

comp_id_2 on page 7-51
Component ID2

comp_id_3 on page 7-51
Component ID3

periph_id_4

JEP106 continuation code

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FD0

Offset: 0x1FD0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								periph_id_4 RO 0x4							

periph_id_4 Fields

Bit	Name	Description	Access	Reset
7:0	periph_id_4	JEP106 continuation code	RO	0x4

periph_id_0

Peripheral ID0

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FE0

Offset: 0x1FE0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								pn7to0 RO 0x1							

periph_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	pn7to0	Part Number [7:0]	RO	0x1

periph_id_1

Peripheral ID1

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FE4

Offset: 0x1FE4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								jep3to0_pn11to8 RO 0xB3							

periph_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	jep3to0_pn11to8	JEP106[3:0], Part Number [11:8]	RO	0xB3

periph_id_2

Peripheral ID2

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FE8

Offset: 0x1FE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_jepcode_jep6to4 RO 0x6B							

periph_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	rev_jepcode_jep6to4	Revision, JEP106 code flag, JEP106[6:4]	RO	0x6B

periph_id_3

Peripheral ID3

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FEC

Offset: 0x1FEC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_and RO 0x0				cust_mod_num RO 0x0			

periph_id_3 Fields

Bit	Name	Description	Access	Reset
7:4	rev_and	Revision	RO	0x0
3:0	cust_mod_num	Customer Model Number	RO	0x0

comp_id_0

Component ID0

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FF0

Offset: 0x1FF0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xD							

comp_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xD

comp_id_1

Component ID1

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FF4

Offset: 0x1FF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								genipcompcls_preamble RO 0xF0							

comp_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	genipcompcls_preamble	Generic IP component class, Preamble	RO	0xF0

comp_id_2

Component ID2

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FF8

Offset: 0x1FF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0x5							

comp_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0x5

comp_id_3

Component ID3

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF801FFC

Offset: `0x1FFC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xB1							

comp_id_3 Fields

Bit	Name	Description	Access	Reset
7:0	`preamble`	Preamble	RO	0xB1

Master Register Group Register Descriptions

Registers associated with master interfaces in the L3 Interconnect. Note that a master in the L3 Interconnect connects to a slave in a module.

Offset: `0x2000`

L4 MAIN Register Descriptions

Registers associated with the L4 MAIN master. This master is used to access the APB slaves on the L4 MAIN bus.

Offset: `0x0`

fn_mod_bm_iss on page 7-52
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF802008`

Offset: `0x2008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

L4 SP Register Descriptions

Registers associated with the L4 SP master. This master is used to access the APB slaves on the L4 SP bus.

Offset: `0x1000`

fn_mod_bm_iss on page 7-53
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF803008

Offset: `0x3008`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td>wr
RW
0x0</td><td>rd
RW 0x0</td></tr>
</table>

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

L4 MP Register Descriptions

Registers associated with the L4 MP master. This master is used to access the APB slaves on the L4 MP bus.

Offset: 0x2000

fn_mod_bm_iss on page 7-54
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF804008

Offset: 0x4008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

L4 OSC1 Register Descriptions

Registers associated with the L4 OSC1 master. This master is used to access the APB slaves on the L4 OSC1 bus.

Offset: `0x3000`

fn_mod_bm_iss on page 7-55
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF805008`

Offset: `0x5008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	`wr`	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	`RW`	`0x0`
0	`rd`	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	`RW`	`0x0`

L4 SPIM Register Descriptions

Registers associated with the L4 SPIM master. This master is used to access the APB slaves on the L4 SPIM bus.

Offset: `0x4000`

fn_mod_bm_iss on page 7-56
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF806008`

Offset: `0x6008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

STM Register Descriptions

Registers associated with the STM master. This master is used to access the STM AXI slave.

Offset: `0x5000`

fn_mod_bm_iss on page 7-57
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-58
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF807008

Offset: `0x7008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF807108

Offset: 0x7108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

LWHPS2FPGA Register Descriptions

Registers associated with the LWHPS2FPGA AXI Bridge master. This master is used to access the LWHPS2FPGA AXI Bridge slave. This slave is used to access the registers for all 3 AXI bridges and to access slaves in the FPGA connected to the LWHPS2FPGA AXI Bridge.

Offset: `0x6000`

fn_mod_bm_iss on page 7-59
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-59
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF808008`

Offset: `0x8008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	`wr`	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	`rd`	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF808108

Offset: 0x8108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

USB1 Register Descriptions

Registers associated with the USB1 master. This master is used to access the registers in USB1.

Offset: 0x8000

fn_mod_bm_iss on page 7-60
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

ahb_cntl on page 7-61
Sets the block issuing capability to one outstanding transaction.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80A008

Offset: 0xA008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ahb_cntl

Sets the block issuing capability to one outstanding transaction.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80A044

Offset: 0xA044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														force_incr RW 0x0	decerr_en RW 0x0

ahb_cntl Fields

Bit	Name	Description	Access	Reset
1	`force_incr`	**Value** / **Description** 0x0 Multiple outstanding write transactions 0x1 If a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat. Also, causes all transactions that are to be output to the AHB domain to be an undefined length INCR.	RW	0x0
0	`decerr_en`	**Value** / **Description** 0x0 No DECERR response. 0x1 If the AHB protocol conversion function receives an unaligned address or a write data beat without all the byte strobes set, creates a DECERR response.	RW	0x0

NANDDATA Register Descriptions

Registers associated with the NANDDATA master. This master is used to access data in the NAND flash controller.

Offset: `0x9000`

fn_mod_bm_iss on page 7-62
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-63
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF80B008`

Offset: `0xB008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF80B108

Offset: 0xB108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

USB0 Register Descriptions

Registers associated with the USB0 master. This master is used to access the registers in USB0.

Offset: 0x1e000

fn_mod_bm_iss on page 7-64
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

ahb_cntl on page 7-65
Sets the block issuing capability to one outstanding transaction.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF820008

Offset: 0x20008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ahb_cntl

Sets the block issuing capability to one outstanding transaction.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF820044

Offset: 0x20044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														force_incr RW 0x0	decerr_en RW 0x0

ahb_cntl Fields

Bit	Name	Description	Access	Reset
1	force_incr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 If a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat. Also, causes all transactions that are to be output to the AHB domain to be an undefined length INCR.	RW	0x0

Bit	Name	Description	Access	Reset
0	decerr_en	**Value** **Description** 0x0 No DECERR response. 0x1 If the AHB protocol conversion function receives an unaligned address or a write data beat without all the byte strobes set, creates a DECERR response.	RW	0x0

NANDREGS Register Descriptions

Registers associated with the NANDREGS master. This master is used to access the registers in the NAND flash controller.

Offset: `0x1f000`

fn_mod_bm_iss on page 7-66
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-67
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF821008

Offset: `0x21008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF821108

Offset: 0x21108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

QSPIDATA Register Descriptions

Registers associated with the QSPIDATA master. This master is used to access data in the QSPI flash controller.

Offset: 0x20000

fn_mod_bm_iss on page 7-68
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

ahb_cntl on page 7-68
Sets the block issuing capability to one outstanding transaction.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF822008

Offset: 0x22008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** / **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** / **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ahb_cntl

Sets the block issuing capability to one outstanding transaction.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF822044

Offset: 0x22044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														force_incr RW 0x0	decerr_en RW 0x0

ahb_cntl Fields

Bit	Name	Description	Access	Reset
1	force_incr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 If a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat. Also, causes all transactions that are to be output to the AHB domain to be an undefined length INCR.	RW	0x0
0	decerr_en	**Value** **Description** 0x0 No DECERR response. 0x1 If the AHB protocol conversion function receives an unaligned address or a write data beat without all the byte strobes set, creates a DECERR response.	RW	0x0

FPGAMGRDATA Register Descriptions

Registers associated with the FPGAMGRDATA master. This master is used to send FPGA configuration image data to the FPGA Manager.

Offset: 0x21000

fn_mod_bm_iss on page 7-70
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

wr_tidemark on page 7-70
Controls the release of the transaction in the write data FIFO.

fn_mod on page 7-71
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF823008

Offset: 0x23008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

wr_tidemark

Controls the release of the transaction in the write data FIFO.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF823040

Offset: 0x23040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x4			

wr_tidemark Fields

Bit	Name	Description	Access	Reset
3:0	level	Stalls the transaction in the write data FIFO until the number of occupied slots in the write data FIFO exceeds the level. Note that the transaction is released before this level is achieved if the network receives the WLAST beat or the write FIFO becomes full.	RW	0x4

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF823108

Offset: 0x23108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

HPS2FPGA Register Descriptions

Registers associated with the HPS2FPGA AXI Bridge master. This master is used to access the HPS2FPGA AXI Bridge slave. This slave is used to access slaves in the FPGA connected to the HPS2FPGA AXI Bridge.

Offset: `0x22000`

fn_mod_bm_iss on page 7-72
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

wr_tidemark on page 7-73
Controls the release of the transaction in the write data FIFO.

fn_mod on page 7-73
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF824008`

Offset: `0x24008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	`wr`	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	`rd`	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

wr_tidemark

Controls the release of the transaction in the write data FIFO.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF824040

Offset: 0x24040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x4			

wr_tidemark Fields

Bit	Name	Description	Access	Reset
3:0	level	Stalls the transaction in the write data FIFO until the number of occupied slots in the write data FIFO exceeds the level. Note that the transaction is released before this level is achieved if the network receives the WLAST beat or the write FIFO becomes full.	RW	0x4

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF824108

Offset: 0x24108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ACP Register Descriptions

Registers associated with the ACP master. This master is used to access the MPU ACP slave via the ACP ID Mapper.

Offset: 0x23000

fn_mod_bm_iss on page 7-74
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-75
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF825008

Offset: 0x25008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF825108

Offset: 0x25108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

Boot ROM Register Descriptions

Registers associated with the Boot ROM master. This master is used to access the contents of the Boot ROM.

Offset: `0x24000`

fn_mod_bm_iss on page 7-76
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

fn_mod on page 7-76
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF826008

Offset: `0x26008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF826108`

Offset: `0x26108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	`wr`	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	`rd`	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

On-chip RAM Register Descriptions

Registers associated with the On-chip RAM master. This master is used to access the contents of the On-chip RAM.

Offset: `0x25000`

fn_mod_bm_iss on page 7-77
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

wr_tidemark on page 7-78
Controls the release of the transaction in the write data FIFO.

fn_mod on page 7-79
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF827008

Offset: 0x27008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

wr_tidemark

Controls the release of the transaction in the write data FIFO.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF827040

Offset: 0x27040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x4			

wr_tidemark Fields

Bit	Name	Description	Access	Reset
3:0	level	Stalls the transaction in the write data FIFO until the number of occupied slots in the write data FIFO exceeds the level. Note that the transaction is released before this level is achieved if the network receives the WLAST beat or the write FIFO becomes full.	RW	0x4

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF827108

Offset: 0x27108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

Slave Register Group Register Descriptions

Registers associated with slave interfaces.

Offset: 0x42000

DAP Register Descriptions

Registers associated with the DAP slave interface. This slave is used by the DAP to access slaves attached to the L3/L4 Interconnect.

Offset: `0x0`

fn_mod2 on page 7-80
Controls bypass merge of upsizing/downsizing.

fn_mod_ahb on page 7-81
Controls how AHB-lite burst transactions are converted to AXI tranactions.

read_qos on page 7-82
QoS (Quality of Service) value for the read channel.

write_qos on page 7-83
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-83
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod2

Controls bypass merge of upsizing/downsizing.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF842024

Offset: `0x42024`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															bypass_merge RW 0x0

fn_mod2 Fields

Bit	Name	Description	Access	Reset
0	bypass_merge	Controls bypass merge of upsizing/downsizing. **Value** **Description** 0x0 The network can alter transactions. 0x1 The network does not alter any transactions that could pass through the upsizer legally without alteration.	RW	0x0

fn_mod_ahb

Controls how AHB-lite burst transactions are converted to AXI tranactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF842028

Offset: 0x42028

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td>wr_incr_override
RW
0x0</td><td>rd_incr_override
RW 0x0</td></tr>
</table>

fn_mod_ahb Fields

Bit	Name	Description	Access	Reset
1	`wr_incr_override`	Controls how AHB-lite write burst transactions are converted to AXI tranactions. **Value** — **Description** 0x0 — The L3 Interconnect converts AHB-lite write bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 — The L3 Interconnect converts AHB-lite write bursts to AXI single transactions.	RW	0x0
0	`rd_incr_override`	Controls how AHB-lite read burst transactions are converted to AXI tranactions. **Value** — **Description** 0x0 — The L3 Interconnect converts AHB-lite read bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 — The L3 Interconnect converts AHB-lite read bursts to AXI single transactions.	RW	0x0

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF842100`

Offset: `0x42100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF842104`

Offset: `0x42104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	`RW`	`0x0`

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF842108`

Offset: `0x42108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

MPU Register Descriptions

Registers associated with the MPU slave interface. This slave is used by the MPU to access slaves attached to the L3/L4 Interconnect.

Offset: `0x1000`

read_qos on page 7-84
QoS (Quality of Service) value for the read channel.

write_qos on page 7-85
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-85
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF843100

Offset: `0x43100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF843104

Offset: `0x43104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF843108

Offset: `0x43108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** / **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** / **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

SDMMC Register Descriptions

Registers associated with the SDMMC slave interface. This slave is used by the DMA controller built into the SDMMC to access slaves attached to the L3/L4 Interconnect.

Offset: 0x2000

fn_mod_ahb on page 7-86
Controls how AHB-lite burst transactions are converted to AXI tranactions.

read_qos on page 7-87
QoS (Quality of Service) value for the read channel.

write_qos on page 7-88
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-88
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_ahb

Controls how AHB-lite burst transactions are converted to AXI tranactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF844028

Offset: 0x44028

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr_incr_override RW 0x0	rd_incr_override RW 0x0

fn_mod_ahb Fields

Bit	Name	Description	Access	Reset
1	wr_incr_override	Controls how AHB-lite write burst transactions are converted to AXI tranactions. **Value** **Description** 0x0 The L3 Interconnect converts AHB-lite write bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite write bursts to AXI single transactions.	RW	0x0
0	rd_incr_override	Controls how AHB-lite read burst transactions are converted to AXI tranactions. **Value** **Description** 0x0 The L3 Interconnect converts AHB-lite read bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite read bursts to AXI single transactions.	RW	0x0

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF844100

Offset: 0x44100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	`RW`	`0x0`

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF844104`

Offset: `0x44104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	`RW`	`0x0`

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF844108`

Offset: `0x44108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

DMA Register Descriptions

Registers associated with the DMA Controller slave interface. This slave is used by the DMA Controller to access slaves attached to the L3/L4 Interconnect.

Offset: 0x3000

read_qos on page 7-89
QoS (Quality of Service) value for the read channel.

write_qos on page 7-90
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-90
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF845100

Offset: 0x45100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF845104

Offset: 0x45104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF845108

Offset: 0x45108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

FPGA2HPS Register Descriptions

Registers associated with the FPGA2HPS AXI Bridge slave interface. This slave is used by the FPGA2HPS AXI Bridge to access slaves attached to the L3/L4 Interconnect.

Offset: 0x4000

wr_tidemark on page 7-91
Controls the release of the transaction in the write data FIFO.

read_qos on page 7-92
QoS (Quality of Service) value for the read channel.

write_qos on page 7-92
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-93
Sets the block issuing capability to multiple or single outstanding transactions.

wr_tidemark

Controls the release of the transaction in the write data FIFO.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF846040

Offset: 0x46040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x4			

wr_tidemark Fields

Bit	Name	Description	Access	Reset
3:0	`level`	Stalls the transaction in the write data FIFO until the number of occupied slots in the write data FIFO exceeds the level. Note that the transaction is released before this level is achieved if the network receives the WLAST beat or the write FIFO becomes full.	RW	0x4

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF846100

Offset: `0x46100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF846104

Offset: `0x46104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF846108

Offset: 0x46108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ETR Register Descriptions

Registers associated with the ETR (TMC) slave interface. This slave is used by the ETR to access slaves attached to the L3/L4 Interconnect.

Offset: `0x5000`

read_qos on page 7-94
QoS (Quality of Service) value for the read channel.

write_qos on page 7-94
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-95
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF847100

Offset: `0x47100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF847104

Offset: `0x47104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF847108

Offset: 0x47108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

EMAC0 Register Descriptions

Registers associated with the EMAC0 slave interface. This slave is used by the DMA controller built into the EMAC0 to access slaves attached to the L3/L4 Interconnect.

Offset: 0x6000

read_qos on page 7-96
QoS (Quality of Service) value for the read channel.

write_qos on page 7-96
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-97
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF848100

Offset: 0x48100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF848104

Offset: 0x48104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF848108

Offset: 0x48108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

EMAC1 Register Descriptions

Registers associated with the EMAC1 slave interface. This slave is used by the DMA controller built into the EMAC1 to access slaves attached to the L3/L4 Interconnect.

Offset: `0x7000`

read_qos on page 7-98
QoS (Quality of Service) value for the read channel.

write_qos on page 7-98
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-99
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF849100`

Offset: `0x49100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	`RW`	`0x0`

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF849104`

Offset: `0x49104`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF849108

Offset: 0x49108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

USB0 Register Descriptions

Registers associated with the USB0 slave interface. This slave is used by the DMA controller built into the USB0 to access slaves attached to the L3/L4 Interconnect.

Offset: `0x8000`

fn_mod_ahb on page 7-100
Controls how AHB-lite burst transactions are converted to AXI tranactions.

read_qos on page 7-101
QoS (Quality of Service) value for the read channel.

write_qos on page 7-102
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-102
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_ahb

Controls how AHB-lite burst transactions are converted to AXI tranactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84A028

Offset: `0x4A028`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr_incr_override RW 0x0	rd_incr_override RW 0x0

fn_mod_ahb Fields

Bit	Name	Description	Access	Reset
1	`wr_incr_override`	Controls how AHB-lite write burst transactions are converted to AXI tranactions. **Value** **Description** 0x0 The L3 Interconnect converts AHB-lite write bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite write bursts to AXI single transactions.	RW	0x0
0	`rd_incr_override`	Controls how AHB-lite read burst transactions are converted to AXI tranactions. **Value** **Description** 0x0 The L3 Interconnect converts AHB-lite read bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite read bursts to AXI single transactions.	RW	0x0

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF84A100`

Offset: `0x4A100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF84A104`

Offset: `0x4A104`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">pri
RW 0x0</td></tr>
</table>

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	`pri`	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	`RW`	`0x0`

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF84A108`

Offset: `0x4A108`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">Reserved</td><td>wr
RW
0x0</td><td>rd
RW 0x0</td></tr>
</table>

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

NAND Register Descriptions

Registers associated with the NAND slave interface. This slave is used by the DMA controller built into the NAND flash controller to access slaves attached to the L3/L4 Interconnect.

Offset: `0x9000`

read_qos on page 7-103
QoS (Quality of Service) value for the read channel.

write_qos on page 7-104
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-104
Sets the block issuing capability to multiple or single outstanding transactions.

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
`l3regs`	`0xFF800000`	`0xFF84B100`

Offset: `0x4B100`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84B104

Offset: 0x4B104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84B108

Offset: 0x4B108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

USB1 Register Descriptions

Registers associated with the USB1 slave interface. This slave is used by the DMA controller built into the USB1 to access slaves attached to the L3/L4 Interconnect.

Offset: 0xa000

fn_mod_ahb on page 7-105
Controls how AHB-lite burst transactions are converted to AXI tranactions.

read_qos on page 7-106
QoS (Quality of Service) value for the read channel.

write_qos on page 7-107
QoS (Quality of Service) value for the write channel.

fn_mod on page 7-107
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_ahb

Controls how AHB-lite burst transactions are converted to AXI tranactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84C028

Offset: 0x4C028

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr_incr_override RW 0x0	rd_incr_override RW 0x0

fn_mod_ahb Fields

Bit	Name	Description	Access	Reset
1	wr_incr_override	Controls how AHB-lite write burst transactions are converted to AXI tranactions. **Value** / **Description** 0x0 The L3 Interconnect converts AHB-lite write bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite write bursts to AXI single transactions.	RW	0x0
0	rd_incr_override	Controls how AHB-lite read burst transactions are converted to AXI tranactions. **Value** / **Description** 0x0 The L3 Interconnect converts AHB-lite read bursts to AXI transactions in accordance with the default behavior as specified in the ARM NIC-301 documentation. 0x1 The L3 Interconnect converts AHB-lite read bursts to AXI single transactions.	RW	0x0

read_qos

QoS (Quality of Service) value for the read channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84C100

Offset: 0x4C100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

read_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the read channel. A higher value has a higher priority.	RW	0x0

write_qos

QoS (Quality of Service) value for the write channel.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84C104

Offset: 0x4C104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												pri RW 0x0			

write_qos Fields

Bit	Name	Description	Access	Reset
3:0	pri	QoS (Quality of Service) value for the write channel. A higher value has a higher priority.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
l3regs	0xFF800000	0xFF84C108

Offset: 0x4C108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

Document Revision History

Table 7-3: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Corrected master interconnect security properties for: • Ethernet MAC • ETR • Added address map and register descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.2	Minor updates.
June 2012	1.1	• Added interconnect connectivity matrix. • Rearranged functional description sections. • Simplified address remapping section. • Added address map and register definitions section.
January 2012	1.0	Initial release.

HPS-FPGA Bridges

8

2014.07.31

cv_54005 Subscribe Send Feedback

This chapter describes the bridges in the hard processor system (HPS) used to communicate data between the FPGA fabric and HPS logic. The bridges use the Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) protocol, and are based on the AMBA Network Interconnect (NIC-301).

The HPS contains the following HPS-FPGA bridges:

- FPGA-to-HPS Bridge
- HPS-to-FPGA Bridge
- Lightweight HPS-to-FPGA Bridge

Related Information

- **I^{2}C Controller** on page 20-1
- **FPGA-to-HPS Bridge** on page 8-4
- **HPS-to-FPGA Bridge Clocks and Resets** on page 8-48
- **Lightweight HPS-to-FPGA Bridge Clocks and Resets** on page 8-48
- **http://infocenter.arm.com/**
  Additional information is available in the AMBA AXI Protocol Specification v1.0 and the AMBA Network Interconnect (NIC-301) Technical Reference Manual, which you can download from the ARM Infocenter website.

Features of the HPS-FPGA Bridges

The HPS-FPGA bridges allow masters in the FPGA fabric to communicate with slaves in the HPS logic and vice versa. For example, you can instantiate additional memories or peripherals in the FPGA fabric, and master interfaces belonging to components in the HPS logic can access them. You can also instantiate components such as a Nios® II processor in the FPGA fabric and their master interfaces can access memories or peripherals in the HPS logic.

Table 8-1: HPS-FPGA Bridge Features

Feature	FPGA-to-HPS Bridge	HPS-to-FPGA Bridge	Lightweight HPS-to-FPGA Bridge
Supports the AMBA AXI3 interface protocol	Y	Y	Y
Implements clock crossing and manages the transfer of data across the clock domains in the HPS logic and the FPGA fabric	Y	Y	Y
Performs data width conversion between the HPS logic and the FPGA fabric	Y	Y	Y
Allows configuration of FPGA interface widths at instantiation time	Y	Y	N

Each bridge consists of a master-slave pair with one interface exposed to the FPGA fabric and the other exposed to the HPS logic. The FPGA-to-HPS bridge exposes an AXI slave interface that you can connect to AXI master or Avalon-MM interfaces in the FPGA fabric. The HPS-to-FPGA and lightweight HPS-to-FPGA bridges expose an AXI master interface that you can connect to AXI or Avalon-MM slave interfaces in the FPGA fabric.

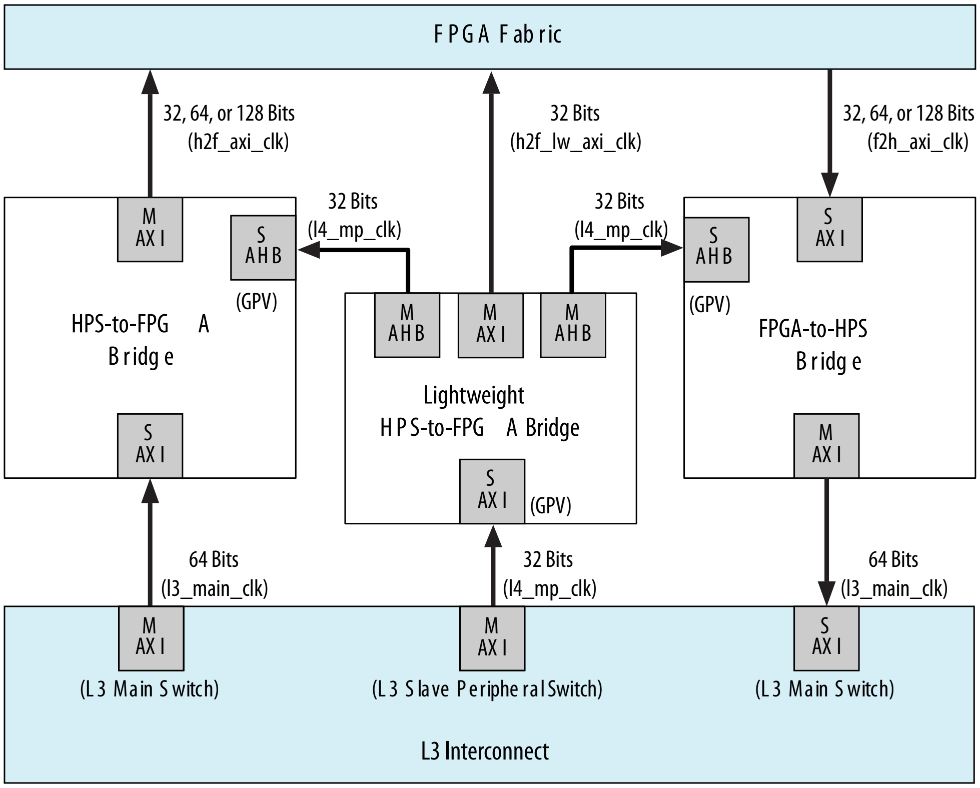
Related Information

AXI Bridges on page 27-3
Information about configuring the AXI bridges

HPS-FPGA Bridges Block Diagram and System Integration

Figure 8-1: HPS-FPGA Bridge Connectivity

The following figure shows the HPS-FPGA bridges in the context of the FPGA fabric and the L3 interconnect to the HPS. Each master (M) and slave (S) interface is shown with its data width(s). The clock domain for each interconnect is shown in parentheses.



The HPS-to-FPGA bridge is mastered by the level 3 (L3) main switch and the lightweight HPS-to-FPGA bridge is mastered by the L3 slave peripheral switch.

The FPGA-to-HPS bridge masters the L3 main switch, allowing any master implemented in the FPGA fabric to access most slaves in the HPS. For example, the FPGA-to-HPS bridge can access the accelerator coherency port (ACP) of the Cortex-A9 MPU subsystem to perform cache-coherent accesses to the SDRAM subsystem.

All three bridges contain global programmers view (GPV) registers. The GPV registers control the behavior of the bridge. Access to the GPV registers of all three bridges is provided through the lightweight HPS-to-FPGA bridge.

Related Information

- **Clocks and Resets** on page 8-47
- **Functional Description of the Interconnect** on page 7-4
  Detailed information about connectivity, such as which masters have access to each bridge

Functional Description of the HPS-FPGA Bridges

The Global Programmers View

The HPS-to-FPGA bridge includes a set of registers called the GPV. The GPV provides settings to control the bridge properties and behavior. Access to the GPV registers of all three bridges is provided through the lightweight HPS-to-FPGA bridge.

The GPV registers can only be accessed by secure masters in the HPS or the FPGA fabric.

FPGA-to-HPS Bridge

The FPGA-to-HPS bridge provides access to the peripherals and memory in the HPS. This access is available to any master implemented in the FPGA fabric. You can configure the bridge slave, which is exposed to the FPGA fabric, to support 32-, 64-, or 128-bit data. The master interface of the bridge, connected to the L3 interconnect, has a data width of 64 bits.

Table 8-2: FPGA-to-HPS Bridge Properties

The following table lists the properties of the FPGA-to-HPS bridge, including the configurable slave interface exposed to the FPGA fabric.

Bridge Property	FPGA Slave Interface	L3 Master Interface
Data width[11]	32, 64, or 128 bits	64 bits
Clock domain	`f2h_axi_clk`	`l3_main_clk`
Byte address width	32 bits	32 bits
ID width	8 bits	8 bits
Read acceptance	16 transactions	16 transactions
Write acceptance	16 transactions	16 transactions
Total acceptance	32 transactions	32 transactions

The FPGA-to-HPS bridge address map contains a GPV. The GPV registers provide settings that adjust the bridge slave properties when the FPGA slave interface is configured to be 32 or 128 bits wide. The slave issuing capability can be adjusted, through the `fn_mod` register, to allow one or multiple transactions to be outstanding in the HPS. The slave bypass merge feature can also be enabled, through the `bypass_merge` bit in the `fn_mod2` register. This feature ensures that the upsizing and downsizing logic does not alter any transactions when the FPGA slave interface is configured to be 32 or 128 bits wide.

Note: It is critical to provide the correct `l4_mp_clk` clock to support access to the GPV, as described in "GPV Clocks".

[11] The bridge slave data width is user-configurable at the time you instantiate the HPS component in your system.

Related Information

- The Global Programmers View on page 8-4
- GPV Clocks on page 8-48

FPGA-to-HPS Access to ACP

When the error correction code (ECC) option is enabled in the level 2 (L2) cache controller, all accesses from the FPGA-to-HPS bridge to the ACP must be 64 bits wide and aligned on 8-byte boundaries after up- or downsizing takes place.

Table 8-3: FPGA Master and FPGA-to-HPS Bridge Configurations

The following table lists some possible master and FPGA-to-HPS bridge slave configurations that support accesses to the L2 cache with ECC enabled.

Soft Logic Master Width	Soft Logic Master Alignment	Soft Logic Master Burst Size (Width)	Soft Logic Master Burst Length	FPGA-to-HPS Bridge Slave Width
32 bits	8 bytes	4 bytes	2, 4, 6, 8, 10, 12, 14, or 16 beats	32 bits
64 bits	8 bytes	8 bytes	1 to 16 beats	32 bits
128 bits	8 or 16 bytes	8 or 16 bytes	1 to 16 beats	32 bits
32 bits	8 bytes	4 bytes	2, 4, 6, 8, 10, 12, 14, or 16 beats	64 bits
64 bits	8 bytes	8 bytes	1 to 16 beats	64 bits
128 bits	8 or 16 bytes	8 or 16 bytes	1 to 16 beats	64 bits
32 bits	8 bytes	4 bytes	2, 4, 6, 8, 10, 12, 14, or 16 beats	128 bits
64 bits	8 bytes	8 bytes	1 to 16 beats	128 bits
128 bits	8 or 16 bytes	8 or 16 bytes	1 to 16 beats	128 bits

Related Information

Cortex-A9 Microprocessor Unit Subsystem on page 9-1
More information about the ECC option of the L2 cache or about interconnect master IDs

FPGA-to-HPS Bridge Slave Signals

The FPGA-to-HPS bridge slave address channels support user-sideband signals, routed to the ACP in the MPU subsystem. All the signals have a fixed width except the data and write strobes for the read and write data channels. The variable width signals depend on the data width setting of the bridge.

The following tables list all the signals exposed by the FPGA-to-HPS slave interface to the FPGA fabric.

Table 8-4: FPGA-to-HPS Bridge Slave Write Address Channel Signals

Signal	Width	Direction	Description
AWID	8 bits	Input	Write address ID
AWADDR	32 bits	Input	Write address
AWLEN	4 bits	Input	Burst length
AWSIZE	3 bits	Input	Burst size
AWBURST	2 bits	Input	Burst type
AWLOCK	2 bits	Input	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
AWCACHE	4 bits	Input	Cache policy type
AWPROT	3 bits	Input	Protection type
AWVALID	1 bit	Input	Write address channel valid
AWREADY	1 bit	Output	Write address channel ready
AWUSER	5 bits	Input	User sideband signals

Table 8-5: FPGA-to-HPS Bridge Slave Write Data Channel Signals

Signal	Width	Direction	Description
WID	8 bits	Input	Write ID
WDATA	32, 64, or 128 bits	Input	Write data
WSTRB	4, 8, or 16 bits	Input	Write data strobes
WLAST	1 bit	Input	Write last data identifier
WVALID	1 bit	Input	Write data channel valid
WREADY	1 bit	Output	Write data channel ready

Table 8-6: FPGA-to-HPS Bridge Slave Write Response Channel Signals

Signal	Width	Direction	Description
BID	8 bits	Output	Write response ID

Signal	Width	Direction	Description
BRESP	2 bits	Output	Write response
BVALID	1 bit	Output	Write response channel valid
BREADY	1 bit	Input	Write response channel ready

Table 8-7: FPGA-to-HPS Bridge Slave Read Address Channel Signals

Signal	Width	Direction	Description
ARID	8 bits	Input	Read address ID
ARADDR	32 bits	Input	Read address
ARLEN	4 bits	Input	Burst length
ARSIZE	3 bits	Input	Burst size
ARBURST	2 bits	Input	Burst type
ARLOCK	2 bits	Input	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
ARCACHE	4 bits	Input	Cache policy type
ARPROT	3 bits	Input	Protection type
ARVALID	1 bit	Input	Read address channel valid
ARREADY	1 bit	Output	Read address channel ready
ARUSER	5 bits	Input	Read user sideband signals

Table 8-8: FPGA-to-HPS Bridge Slave Read Data Channel Signals

Signal	Width	Direction	Description
RID	8 bits	Output	Read ID
RDATA	32, 64, or 128 bits	Output	Read data
RRESP	2 bits	Output	Read response
RLAST	1 bit	Output	Read last data identifier
RVALID	1 bit	Output	Read data channel valid

Signal	Width	Direction	Description
RREADY	1 bit	Input	Read data channel ready

FPGA2HPS AXI Bridge Module Address Map

Registers in the FPGA2HPS AXI Bridge Module.

Base Address: 0xFF600000

ID Register Group

Register	Offset	Width	Access	Reset Value	Description
periph_id_4 on page 8-9	0x1FD0	32	RO	0x4	Peripheral ID4 Register
periph_id_0 on page 8-10	0x1FE0	32	RO	0x1	Peripheral ID0 Register
periph_id_1 on page 8-10	0x1FE4	32	RO	0xB3	Peripheral ID1 Register
periph_id_2 on page 8-10	0x1FE8	32	RO	0x6B	Peripheral ID2 Register
periph_id_3 on page 8-11	0x1FEC	32	RO	0x0	Peripheral ID3 Register
comp_id_0 on page 8-11	0x1FF0	32	RO	0xD	Component ID0 Register
comp_id_1 on page 8-12	0x1FF4	32	RO	0xF0	Component ID1 Register
comp_id_2 on page 8-12	0x1FF8	32	RO	0x5	Component ID2 Register
comp_id_3 on page 8-13	0x1FFC	32	RO	0xB1	Component ID3 Register

32-bit Slave

Register	Offset	Width	Access	Reset Value	Description
fn_mod2 on page 8-14	0x42024	32	RW	0x0	Functionality Modification 2 Register
fn_mod on page 8-14	0x42108	32	RW	0x0	Issuing Functionality Modification Register

128-bit Slave

Register	Offset	Width	Access	Reset Value	Description
fn_mod2 on page 8-15	0x44024	32	RW	0x0	Functionality Modification 2 Register
fn_mod on page 8-16	0x44108	32	RW	0x0	Issuing Functionality Modification Register

ID Register Group Register Descriptions

Contains registers that identify the ARM NIC-301 IP Core.

Offset: `0x1000`

periph_id_4 on page 8-9
JEP106 continuation code

periph_id_0 on page 8-10
Peripheral ID0

periph_id_1 on page 8-10
Peripheral ID1

periph_id_2 on page 8-10
Peripheral ID2

periph_id_3 on page 8-11
Peripheral ID3

comp_id_0 on page 8-11
Component ID0

comp_id_1 on page 8-12
Component ID1

comp_id_2 on page 8-12
Component ID2

comp_id_3 on page 8-13
Component ID3

periph_id_4

JEP106 continuation code

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FD0

Offset: `0x1FD0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								periph_id_4 RO 0x4							

periph_id_4 Fields

Bit	Name	Description	Access	Reset
7:0	periph_id_4	JEP106 continuation code	RO	0x4

periph_id_0

Peripheral ID0

Module Instance	Base Address	Register Address
`fpga2hpsregs`	`0xFF600000`	`0xFF601FE0`

Offset: `0x1FE0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								pn7to0 RO 0x1							

periph_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	`pn7to0`	Part Number [7:0]	`RO`	`0x1`

periph_id_1

Peripheral ID1

Module Instance	Base Address	Register Address
`fpga2hpsregs`	`0xFF600000`	`0xFF601FE4`

Offset: `0x1FE4`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								jep3to0_pn11to8 RO 0xB3							

periph_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	`jep3to0_pn11to8`	JEP106[3:0], Part Number [11:8]	`RO`	`0xB3`

periph_id_2

Peripheral ID2

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FE8

Offset: 0x1FE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_jepcode_jep6to4 RO 0x6B							

periph_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	rev_jepcode_jep6to4	Revision, JEP106 code flag, JEP106[6:4]	RO	0x6B

periph_id_3

Peripheral ID3

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FEC

Offset: 0x1FEC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_and RO 0x0				cust_mod_num RO 0x0			

periph_id_3 Fields

Bit	Name	Description	Access	Reset
7:4	rev_and	Revision	RO	0x0
3:0	cust_mod_num	Customer Model Number	RO	0x0

comp_id_0

Component ID0

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FF0

Offset: 0x1FF0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xD							

comp_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xD

comp_id_1

Component ID1

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FF4

Offset: 0x1FF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								genipcompcls_preamble RO 0xF0							

comp_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	genipcompcls_preamble	Generic IP component class, Preamble	RO	0xF0

comp_id_2

Component ID2

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FF8

Offset: 0x1FF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0x5							

comp_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0x5

comp_id_3

Component ID3

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF601FFC

Offset: 0x1FFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xB1							

comp_id_3 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xB1

Slave Register Group Register Descriptions

Registers associated with slave interfaces.

Offset: 0x42000

32-bit Slave Register Descriptions

Registers associated with the 32-bit AXI slave interface. These registers are only active when the FPGA2HPS AXI Bridge is configured with a 32-bit FPGA AXI slave interface.

Offset: 0x0

fn_mod2 on page 8-14
Controls bypass merge of upsizing/downsizing.

fn_mod on page 8-14
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod2
Controls bypass merge of upsizing/downsizing.

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF642024

Offset: 0x42024

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															bypass_merge RW 0x0

fn_mod2 Fields

Bit	Name	Description	Access	Reset
0	bypass_merge	Controls bypass merge of upsizing/downsizing. **Value** **Description** 0x0 The network can alter transactions. 0x1 The network does not alter any transactions that could pass through the upsizer legally without alteration.	RW	0x0

fn_mod
Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF642108

Offset: 0x42108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

128-bit Slave Register Descriptions

Registers associated with the 128-bit AXI slave interface. These registers are only active when the FPGA2HPS AXI Bridge is configured with a 128-bit FPGA AXI slave interface.

Offset: 0x2000

fn_mod2 on page 8-15
Controls bypass merge of upsizing/downsizing.

fn_mod on page 8-16
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod2

Controls bypass merge of upsizing/downsizing.

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF644024

Offset: 0x44024

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															bypass_merge RW 0x0

fn_mod2 Fields

Bit	Name	Description	Access	Reset
0	bypass_merge	Controls bypass merge of upsizing/downsizing. **Value** **Description** 0x0 The network can alter transactions. 0x1 The network does not alter any transactions that could pass through the upsizer legally without alteration.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
fpga2hpsregs	0xFF600000	0xFF644108

Offset: 0x44108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** / **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** / **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

HPS-to-FPGA Bridge

The HPS-to-FPGA bridge provides a configurable-width, high-performance master interface to the FPGA fabric. The bridge provides most masters in the HPS with access to logic, peripherals, and memory implemented in the FPGA. The effective size of the address space is 0x3FFF0000, or 1 gigabyte (GB) minus 64 megabytes (MB). The address space size is less than 1 GB because 64 MB is occupied by peripherals, lightweight HPS-to-FPGA bridge, on-chip RAM, and boot ROM in the HPS. You can configure the bridge master exposed to the FPGA fabric for 32-, 64-, or 128-bit data. The amount of address space exposed to the MPU subsystem can also be reduced through the L2 cache address filtering mechanism.

The slave interface of the bridge in the HPS logic has a data width of 64 bits. The bridge provides width adaptation and clock crossing logic that allows the logic in the FPGA to operate in any clock domain, asynchronous from the HPS.

Note: The HPS-to-FPGA bridge is accessed if the MPU boots from the FPGA. Before the MPU boots from the FPGA, the FPGA portion of the SoC device must be configured, and the HPS-to-FPGA bridge must be remapped into addressable space.

Table 8-9: HPS-to-FPGA Bridge Properties

The following table lists the properties of the HPS-to-FPGA bridge, including the configurable master interface exposed to the FPGA fabric.

Bridge Property	L3 Slave Interface	FPGA Master Interface
Data width [12]	64 bits	32, 64, or 128 bits
Clock domain	`l3_main_clk`	`h2f_axi_clk`
Byte address width	32 bits	30 bits
ID width	12 bits	12 bits

[12] The bridge master data width is user-configurable at the time you instantiate the HPS component in your system.

Bridge Property	L3 Slave Interface	FPGA Master Interface
Read acceptance	16 transactions	16 transactions
Write acceptance	16 transactions	16 transactions
Total acceptance	32 transactions	32 transactions

The HPS-to-FPGA bridge's GPV, described in "The Global Programmers View", provides settings to adjust the bridge master properties. The master issuing capability can be adjusted, through the `fn_mod` register, to allow one or multiple transactions to be outstanding in the FPGA fabric. The master bypass merge feature can also be enabled, through the `bypass_merge` bit in the `fn_mod2` register. This feature ensures that the upsizing and downsizing logic does not alter any transactions when the FPGA master interface is configured to be 32 or 128 bits wide.

Note: It is critical to provide the correct `l4_mp_clk` clock to support access to the GPV, as described in "GPV Clocks".

Related Information

- **The Global Programmers View** on page 8-4
- **GPV Clocks** on page 8-48
- **Functional Description of the Interconnect** on page 7-4
  Detailed information about connectivity, such as which masters have access to each bridge
- **Cortex-A9 Microprocessor Unit Subsystem** on page 9-1
  Details about L2 cache address filtering
- **AXI Bridges** on page 27-3
  Information about configuring the AXI bridges

HPS-to-FPGA Bridge Master Signals

All the HPS-to-FPGA bridge master signals have a fixed width except the data and write strobes for the read and write data channels. The variable-width signals depend on the data width setting of the bridge interface exposed to the FPGA logic.

The following tables list all the signals exposed by the HPS-to-FPGA master interface to the FPGA fabric.

Table 8-10: HPS-to-FPGA Bridge Master Write Address Channel Signals

Signal	Width	Direction	Description
`AWID`	12 bits	Output	Write address ID
`AWADDR`	30 bits	Output	Write address
`AWLEN`	4 bits	Output	Burst length
`AWSIZE`	3 bits	Output	Burst size
`AWBURST`	2 bits	Output	Burst type

Signal	Width	Direction	Description
AWLOCK	2 bits	Output	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
AWCACHE	4 bits	Output	Cache policy type
AWPROT	3 bits	Output	Protection type
AWVALID	1 bit	Output	Write address channel valid
AWREADY	1 bit	Input	Write address channel ready

Table 8-11: HPS-to-FPGA Bridge Master Write Data Channel Signals

Signal	Width	Direction	Description
WID	12 bits	Output	Write ID
WDATA	32, 64, or 128 bits	Output	Write data
WSTRB	4, 8, or 16 bits	Output	Write data strobes
WLAST	1 bit	Output	Write last data identifier
WVALID	1 bit	Output	Write data channel valid
WREADY	1 bit	Input	Write data channel ready

Table 8-12: HPS-to-FPGA Bridge Master Write Response Channel Signals

Signal	Width	Direction	Description
BID	12 bits	Input	Write response ID
BRESP	2 bits	Input	Write response
BVALID	1 bit	Input	Write response channel valid
BREADY	1 bit	Output	Write response channel ready

Table 8-13: HPS-to-FPGA Bridge Master Read Address Channel Signals

Signal	Width	Direction	Description
ARID	12 bits	Output	Read address ID
ARADDR	30 bits	Output	Read address

Signal	Width	Direction	Description
ARLEN	4 bits	Output	Burst length
ARSIZE	3 bits	Output	Burst size
ARBURST	2 bits	Output	Burst type
ARLOCK	2 bits	Output	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
ARCACHE	4 bits	Output	Cache policy type
ARPROT	3 bits	Output	Protection type
ARVALID	1 bit	Output	Read address channel valid
ARREADY	1 bit	Input	Read address channel ready

Table 8-14: HPS-to-FPGA Bridge Master Read Data Channel Signals

Signal	Width	Direction	Description
RID	12 bits	Input	Read ID
RDATA	32, 64, or 128 bits	Input	Read data
RRESP	2 bits	Input	Read response
RLAST	1 bit	Input	Read last data identifier
RVALID	1 bit	Input	Read data channel valid
RREADY	1 bit	Output	Read data channel ready

HPS2FPGA AXI Bridge Module Address Map

Registers in the HPS2FPGA AXI Bridge Module.

Base Address: 0xFF500000

ID Register Group

Register	Offset	Width	Access	Reset Value	Description
periph_id_4 on page 8-22	0x1FD0	32	RO	0x4	Peripheral ID4 Register
periph_id_0 on page 8-22	0x1FE0	32	RO	0x1	Peripheral ID0 Register

Register	Offset	Width	Access	Reset Value	Description
periph_id_1 on page 8-23	0x1FE4	32	RO	0xB3	Peripheral ID1 Register
periph_id_2 on page 8-23	0x1FE8	32	RO	0x6B	Peripheral ID2 Register
periph_id_3 on page 8-24	0x1FEC	32	RO	0x0	Peripheral ID3 Register
comp_id_0 on page 8-24	0x1FF0	32	RO	0xD	Component ID0 Register
comp_id_1 on page 8-25	0x1FF4	32	RO	0xF0	Component ID1 Register
comp_id_2 on page 8-25	0x1FF8	32	RO	0x5	Component ID2 Register
comp_id_3 on page 8-26	0x1FFC	32	RO	0xB1	Component ID3 Register

32-bit Master

Register	Offset	Width	Access	Reset Value	Description
fn_mod2 on page 8-27	0x2024	32	RW	0x0	Functionality Modification 2 Register
fn_mod on page 8-27	0x2108	32	RW	0x0	Issuing Functionality Modification Register

128-bit Master

Register	Offset	Width	Access	Reset Value	Description
fn_mod2 on page 8-28	0x4024	32	RW	0x0	Functionality Modification 2 Register
fn_mod on page 8-29	0x4108	32	RW	0x0	Issuing Functionality Modification Register

ID Register Group Register Descriptions

Contains registers that identify the ARM NIC-301 IP Core.

Offset: `0x1000`

periph_id_4 on page 8-22
JEP106 continuation code

periph_id_0 on page 8-22
Peripheral ID0

periph_id_1 on page 8-23
Peripheral ID1

periph_id_2 on page 8-23
Peripheral ID2

periph_id_3 on page 8-24
Peripheral ID3

comp_id_0 on page 8-24
Component ID0

comp_id_1 on page 8-25
Component ID1

comp_id_2 on page 8-25
Component ID2

comp_id_3 on page 8-26
Component ID3

periph_id_4

JEP106 continuation code

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FD0

Offset: 0x1FD0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								periph_id_4 RO 0x4							

periph_id_4 Fields

Bit	Name	Description	Access	Reset
7:0	periph_id_4	JEP106 continuation code	RO	0x4

periph_id_0

Peripheral ID0

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FE0

Offset: 0x1FE0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								pn7to0 RO 0x1							

periph_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	pn7to0	Part Number [7:0]	RO	0x1

periph_id_1

Peripheral ID1

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FE4

Offset: 0x1FE4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								jep3to0_pn11to8 RO 0xB3							

periph_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	jep3to0_pn11to8	JEP106[3:0], Part Number [11:8]	RO	0xB3

periph_id_2

Peripheral ID2

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FE8

Offset: 0x1FE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_jepcode_jep6to4 RO 0x6B							

periph_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	rev_jepcode_jep6to4	Revision, JEP106 code flag, JEP106[6:4]	RO	0x6B

periph_id_3

Peripheral ID3

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FEC

Offset: 0x1FEC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_and RO 0x0				cust_mod_num RO 0x0			

periph_id_3 Fields

Bit	Name	Description	Access	Reset
7:4	rev_and	Revision	RO	0x0
3:0	cust_mod_num	Customer Model Number	RO	0x0

comp_id_0

Component ID0

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FF0

Offset: 0x1FF0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xD							

comp_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xD

comp_id_1

Component ID1

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FF4

Offset: 0x1FF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								genipcompcls_preamble RO 0xF0							

comp_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	genipcompcls_preamble	Generic IP component class, Preamble	RO	0xF0

comp_id_2

Component ID2

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FF8

Offset: 0x1FF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0x5							

comp_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0x5

comp_id_3

Component ID3

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF501FFC

Offset: 0x1FFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xB1							

comp_id_3 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xB1

Master Register Group Register Descriptions

Registers associated with master interfaces.

Offset: 0x2000

32-bit Master Register Descriptions

Registers associated with the 32-bit AXI master interface. These registers are only active when the HPS2FPGA AXI Bridge is configured with a 32-bit FPGA AXI master interface.

Offset: 0x0

fn_mod2 on page 8-27
Controls bypass merge of upsizing/downsizing.

fn_mod on page 8-27
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod2
Controls bypass merge of upsizing/downsizing.

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF502024

Offset: 0x2024

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															bypass_merge RW 0x0

fn_mod2 Fields

Bit	Name	Description	Access	Reset
0	bypass_merge	Controls bypass merge of upsizing/downsizing. **Value** **Description** 0x0 The network can alter transactions. 0x1 The network does not alter any transactions that could pass through the upsizer legally without alteration.	RW	0x0

fn_mod
Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF502108

Offset: 0x2108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

128-bit Master Register Descriptions

Registers associated with the 128-bit AXI master interface. These registers are only active when the HPS2FPGA AXI Bridge is configured with a 128-bit FPGA AXI master interface.

Offset: 0x2000

fn_mod2 on page 8-28
Controls bypass merge of upsizing/downsizing.

fn_mod on page 8-29
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod2

Controls bypass merge of upsizing/downsizing.

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF504024

Offset: 0x4024

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															bypass_merge RW 0x0

fn_mod2 Fields

Bit	Name	Description	Access	Reset
0	bypass_merge	Controls bypass merge of upsizing/downsizing. **Value** **Description** 0x0 The network can alter transactions. 0x1 The network does not alter any transactions that could pass through the upsizer legally without alteration.	RW	0x0

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
hps2fpgaregs	0xFF500000	0xFF504108

Offset: 0x4108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

Lightweight HPS-to-FPGA Bridge

The lightweight HPS-to-FPGA bridge provides a lower-performance interface to the FPGA fabric. This interface is useful for accessing the control and status registers of soft peripherals. The bridge provides a 2 MB address space and access to logic, peripherals, and memory implemented in the FPGA fabric. The MPU subsystem, direct memory access (DMA) controller, and debug access port (DAP) can use the lightweight HPS-to-FPGA bridge to access the FPGA fabric or GPV. Master interfaces in the FPGA fabric can also use the lightweight HPS-to-FPGA bridge to access the GPV registers in all three bridges.

The bridge master exposed to the FPGA fabric has a fixed data width of 32 bits. The slave interface of the bridge in the HPS logic has a fixed data width of 32 bits.

Use the lightweight HPS-to-FPGA bridge as a secondary, lower-performance master interface to the FPGA fabric. With a fixed width and a smaller address space, the lightweight bridge is useful for low-bandwidth traffic, such as memory-mapped register accesses to FPGA peripherals. This approach diverts traffic from the high-performance HPS-to-FPGA bridge, and can improve both CSR access latency and overall system performance.

Table 8-15: Lightweight HPS-to-FPGA Bridge Properties

This table lists the properties of the lightweight HPS-to-FPGA bridge, including the master interface exposed to the FPGA fabric.

Bridge Property	L3 Slave Interface	FPGA Master Interface
Data width	32 bits	32 bits
Clock domain	`l4_mp_clk`	`h2f_lw_axi_clk`
Byte address width	32 bits	21 bits
ID width	12 bits	12 bits
Read acceptance	16 transactions	16 transactions
Write acceptance	16 transactions	16 transactions

Bridge Property	L3 Slave Interface	FPGA Master Interface
Total acceptance	32 transactions	32 transactions

The lightweight HPS-to-FPGA bridge has three master interfaces. The master interface connected to the FPGA fabric provides a lightweight interface from the HPS to custom logic in the FPGA fabric. The two other master interfaces, connected to the HPS-to-FPGA and FPGA-to-HPS bridges, allow you to access the GPV registers for each bridge.

The lightweight HPS-to-FPGA bridge also has a set of registers GPV to control the behavior of its four interfaces (one slave and three masters).

The GPV allows you to set the bridge's issuing capabilities to support single or multiple transactions. The GPV also lets you set a write tidemark through the `wr_tidemark` register, to control how much data is buffered in the bridge before data is written to slaves in the FPGA fabric.

Note: It is critical to provide correct clock settings for the lightweight HPS-to-FPGA bridge, even if your design does not use this bridge. The `l4_mp_clk` clock is required for GPV access on the HPS-to-FPGA and FPGA-to-HPS bridges.

Related Information

- **HPS-FPGA Bridges Block Diagram and System Integration** on page 8-3
  Figure showing the lightweight HPS-to-FPGA bridge's three master interfaces
- **The Global Programmers View** on page 8-4
  Includes a description of the lightweight HPS-to-FPGA bridge GPV
- **Functional Description of the Interconnect** on page 7-4
  Detailed information about connectivity, such as which masters have access to each bridge
- **sfo1406668266210.ditamap**
  Detailed information about the `wr_tidemark` register
- **AXI Bridges** on page 27-3
  Information about configuring the AXI bridges

Lightweight HPS-to-FPGA Bridge Master Signals

All the lightweight HPS-to-FPGA bridge master signals have a fixed width.

The following tables list all the signals exposed by the lightweight HPS-to-FPGA master interface to the FPGA fabric.

Table 8-16: Lightweight HPS-to-FPGA Bridge Master Write Address Channel Signals

Signal	Width	Direction	Description
AWID	12 bits	Output	Write address ID
AWADDR	21 bits	Output	Write address
AWLEN	4 bits	Output	Burst length
AWSIZE	3 bits	Output	Burst size
AWBURST	2 bits	Output	Burst type

Signal	Width	Direction	Description
AWLOCK	2 bits	Output	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
AWCACHE	4 bits	Output	Cache policy type
AWPROT	3 bits	Output	Protection type
AWVALID	1 bit	Output	Write address channel valid
AWREADY	1 bit	Input	Write address channel ready

Table 8-17: Lightweight HPS-to-FPGA Bridge Master Write Data Channel Signals

Signal	Width	Direction	Description
WID	12 bits	Output	Write ID
WDATA	32 bits	Output	Write data
WSTRB	4 bits	Output	Write data strobes
WLAST	1 bit	Output	Write last data identifier
WVALID	1 bit	Output	Write data channel valid
WREADY	1 bit	Input	Write data channel ready

Table 8-18: Lightweight HPS-to-FPGA Bridge Master Write Response Channel Signals

Signal	Width	Direction	Description
BID	12 bits	Input	Write response ID
BRESP	2 bits	Input	Write response
BVALID	1 bit	Input	Write response channel valid
BREADY	1 bit	Output	Write response channel ready

Table 8-19: Lightweight HPS-to-FPGA Bridge Master Read Address Channel Signals

Signal	Width	Direction	Description
ARID	12 bits	Output	Read address ID
ARADDR	21 bits	Output	Read address

Signal	Width	Direction	Description
ARLEN	4 bits	Output	Burst length
ARSIZE	3 bits	Output	Burst size
ARBURST	2 bits	Output	Burst type
ARLOCK	2 bits	Output	Lock type—Valid values are 00 (normal access) and 01 (exclusive access)
ARCACHE	4 bits	Output	Cache policy type
ARPROT	3 bits	Output	Protection type
ARVALID	1 bit	Output	Read address channel valid
ARREADY	1 bit	Input	Read address channel ready

Table 8-20: Lightweight HPS-to-FPGA Bridge Master Read Data Channel Signals

Signal	Width	Direction	Description
RID	12 bits	Input	Read ID
RDATA	32 bits	Input	Read data
RRESP	2 bits	Input	Read response
RLAST	1 bit	Input	Read last data identifier
RVALID	1 bit	Input	Read data channel valid
RREADY	1 bit	Output	Read data channel ready

LWHPS2FPGA AXI Bridge Module Address Map

Registers in the LWHPS2FPGA AXI Bridge Module.

Base Address: 0xFF400000

ID Register Group

Register	Offset	Width	Access	Reset Value	Description
periph_id_4 on page 8-35	0x1FD0	32	RO	0x4	Peripheral ID4 Register
periph_id_0 on page 8-36	0x1FE0	32	RO	0x1	Peripheral ID0 Register

Register	Offset	Width	Acces s	Reset Value	Description
periph_id_1 on page 8-36	0x1FE4	32	RO	0xB3	Peripheral ID1 Register
periph_id_2 on page 8-37	0x1FE8	32	RO	0x6B	Peripheral ID2 Register
periph_id_3 on page 8-37	0x1FEC	32	RO	0x0	Peripheral ID3 Register
comp_id_0 on page 8-38	0x1FF0	32	RO	0xD	Component ID0 Register
comp_id_1 on page 8-38	0x1FF4	32	RO	0xF0	Component ID1 Register
comp_id_2 on page 8-39	0x1FF8	32	RO	0x5	Component ID2 Register
comp_id_3 on page 8-39	0x1FFC	32	RO	0xB1	Component ID3 Register

FPGA2HPS AXI Bridge Registers

Register	Offset	Width	Acces s	Reset Value	Description
fn_mod_bm_iss on page 8-40	0x2008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
ahb_cntl on page 8-41	0x2044	32	RW	0x0	AHB Control Register

HPS2FPGA AXI Bridge Registers

Register	Offset	Width	Acces s	Reset Value	Description
fn_mod_bm_iss on page 8-42	0x3008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
ahb_cntl on page 8-43	0x3044	32	RW	0x0	AHB Control Register

32-bit Master

Register	Offset	Width	Acces s	Reset Value	Description
fn_mod_bm_iss on page 8-44	0x5008	32	RW	0x0	Bus Matrix Issuing Functionality Modification Register
wr_tidemark on page 8-45	0x5040	32	RW	0x4	Write Tidemark
fn_mod on page 8-45	0x5108	32	RW	0x0	Issuing Functionality Modification Register

L3 Slave Register Group

Register	Offset	Width	Access	Reset Value	Description
fn_mod on page 8-46	0x45108	32	RW	0x0	Issuing Functionality Modification Register

ID Register Group Register Descriptions

Contains registers that identify the ARM NIC-301 IP Core.

Offset: 0x1000

periph_id_4 on page 8-35
JEP106 continuation code

periph_id_0 on page 8-36
Peripheral ID0

periph_id_1 on page 8-36
Peripheral ID1

periph_id_2 on page 8-37
Peripheral ID2

periph_id_3 on page 8-37
Peripheral ID3

comp_id_0 on page 8-38
Component ID0

comp_id_1 on page 8-38
Component ID1

comp_id_2 on page 8-39
Component ID2

comp_id_3 on page 8-39
Component ID3

periph_id_4

JEP106 continuation code

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FD0

Offset: 0x1FD0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								periph_id_4 RO 0x4							

periph_id_4 Fields

Bit	Name	Description	Access	Reset
7:0	periph_id_4	JEP106 continuation code	RO	0x4

periph_id_0

Peripheral ID0

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FE0

Offset: 0x1FE0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								pn7to0 RO 0x1							

periph_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	pn7to0	Part Number [7:0]	RO	0x1

periph_id_1

Peripheral ID1

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FE4

Offset: 0x1FE4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								jep3to0_pn11to8 RO 0xB3							

periph_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	jep3to0_pn11to8	JEP106[3:0], Part Number [11:8]	RO	0xB3

periph_id_2

Peripheral ID2

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FE8

Offset: 0x1FE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_jepcode_jep6to4 RO 0x6B							

periph_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	rev_jepcode_jep6to4	Revision, JEP106 code flag, JEP106[6:4]	RO	0x6B

periph_id_3

Peripheral ID3

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FEC

Offset: 0x1FEC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rev_and RO 0x0				cust_mod_num RO 0x0			

periph_id_3 Fields

Bit	Name	Description	Access	Reset
7:4	rev_and	Revision	RO	0x0
3:0	cust_mod_num	Customer Model Number	RO	0x0

comp_id_0

Component ID0

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FF0

Offset: 0x1FF0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xD							

comp_id_0 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0xD

comp_id_1

Component ID1

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FF4

Offset: 0x1FF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								genipcompcls_preamble RO 0xF0							

comp_id_1 Fields

Bit	Name	Description	Access	Reset
7:0	genipcompcls_preamble	Generic IP component class, Preamble	RO	0xF0

comp_id_2

Component ID2

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FF8

Offset: 0x1FF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0x5							

comp_id_2 Fields

Bit	Name	Description	Access	Reset
7:0	preamble	Preamble	RO	0x5

comp_id_3

Component ID3

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF401FFC

Offset: 0x1FFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								preamble RO 0xB1							

comp_id_3 Fields

Bit	Name	Description	Access	Reset
7:0	`preamble`	Preamble	RO	0xB1

Master Register Group Register Descriptions

Registers associated with master interfaces.

Offset: `0x2000`

FPGA2HPS AXI Bridge Registers Register Descriptions

Registers associated with the FPGA2HPS master interface. This master interface provides access to the registers in the FPGA2HPS AXI Bridge.

Offset: `0x0`

fn_mod_bm_iss on page 8-40
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

ahb_cntl on page 8-41
Sets the block issuing capability to one outstanding transaction.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
`lwhps2fpgaregs`	`0xFF400000`	`0xFF402008`

Offset: `0x2008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	`wr`	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0

Bit	Name	Description	Access	Reset
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ahb_cntl

Sets the block issuing capability to one outstanding transaction.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF402044

Offset: 0x2044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														force_incr RW 0x0	decerr_en RW 0x0

ahb_cntl Fields

Bit	Name	Description	Access	Reset
1	force_incr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 If a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat. Also, causes all transactions that are to be output to the AHB domain to be an undefined length INCR.	RW	0x0

Bit	Name	Description	Access	Reset
0	decerr_en	**Value** **Description** 0x0 No DECERR response. 0x1 If the AHB protocol conversion function receives an unaligned address or a write data beat without all the byte strobes set, creates a DECERR response.	RW	0x0

HPS2FPGA AXI Bridge Registers Register Descriptions

Registers associated with the HPS2FPGA master interface. This master interface provides access to the registers in the HPS2FPGA AXI Bridge.

Offset: `0x1000`

fn_mod_bm_iss on page 8-42
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

ahb_cntl on page 8-43
Sets the block issuing capability to one outstanding transaction.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF403008

Offset: `0x3008`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0

Bit	Name	Description	Access	Reset
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

ahb_cntl

Sets the block issuing capability to one outstanding transaction.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF403044

Offset: 0x3044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														force_incr RW 0x0	decerr_en RW 0x0

ahb_cntl Fields

Bit	Name	Description	Access	Reset
1	force_incr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 If a beat is received that has no write data strobes set, that write data beat is replaced with an IDLE beat. Also, causes all transactions that are to be output to the AHB domain to be an undefined length INCR.	RW	0x0

Bit	Name	Description	Access	Reset
0	decerr_en	**Value** **Description** 0x0 No DECERR response. 0x1 If the AHB protocol conversion function receives an unaligned address or a write data beat without all the byte strobes set, creates a DECERR response.	RW	0x0

32-bit Master Register Descriptions

Registers associated with the 32-bit AXI master interface. This master provides access to slaves in the FPGA.

Offset: 0x3000

fn_mod_bm_iss on page 8-44
Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

wr_tidemark on page 8-45
Controls the release of the transaction in the write data FIFO.

fn_mod on page 8-45
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod_bm_iss

Sets the issuing capability of the preceding switch arbitration scheme to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF405008

Offset: 0x5008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod_bm_iss Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

wr_tidemark

Controls the release of the transaction in the write data FIFO.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF405040

Offset: 0x5040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x4			

wr_tidemark Fields

Bit	Name	Description	Access	Reset
3:0	level	Stalls the transaction in the write data FIFO until the number of occupied slots in the write data FIFO exceeds the level. Note that the transaction is released before this level is achieved if the network receives the WLAST beat or the write FIFO becomes full.	RW	0x4

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF405108

Offset: 0x5108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** — **Description** 0x0 — Multiple outstanding write transactions 0x1 — Only a single outstanding write transaction	RW	0x0
0	rd	**Value** — **Description** 0x0 — Multiple outstanding read transactions 0x1 — Only a single outstanding read transaction	RW	0x0

Slave Register Group Register Descriptions

Registers associated with slave interfaces.

Offset: 0x42000

L3 Slave Register Group Register Descriptions

Registers associated with the 32-bit AXI slave interface. This slave connects to the L3 Interconnect.

Offset: 0x3000

fn_mod on page 8-46
Sets the block issuing capability to multiple or single outstanding transactions.

fn_mod

Sets the block issuing capability to multiple or single outstanding transactions.

Module Instance	Base Address	Register Address
lwhps2fpgaregs	0xFF400000	0xFF445108

Offset: 0x45108

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														wr RW 0x0	rd RW 0x0

fn_mod Fields

Bit	Name	Description	Access	Reset
1	wr	**Value** **Description** 0x0 Multiple outstanding write transactions 0x1 Only a single outstanding write transaction	RW	0x0
0	rd	**Value** **Description** 0x0 Multiple outstanding read transactions 0x1 Only a single outstanding read transaction	RW	0x0

FPGA Slaves Accessed Via Lightweight HPS2FPGA AXI Bridge (lwfpgaslaves) Address Map

This address space is allocated for FPGA-configured slaves driven by the lightweight HPS-to-FPGA bridge master. Address assignment within this space is user-defined. For more information about Lightweight HPS-to-FPGA bridges, refer to the *HPS-FPGA Bridges* chapter of the *Hard Processor System Technical Reference Manual.*

Table 8-21: lwfpgaslaves Address Range

Module Instance	Start Address	End Address
LWFPGASLAVES	0xFF200000	0xFF3FFFFF

Clocks and Resets

FPGA-to-HPS Bridge Clocks and Resets

The master interface of the bridge in the HPS logic operates in the `l3_main_clk` clock domain. The slave interface exposed to the FPGA fabric operates in the `f2h_axi_clk` clock domain provided by the user logic. The bridge provides clock crossing logic that allows the logic in the FPGA to operate in any clock domain, asynchronous from the HPS.

The FPGA-to-HPS bridge has one reset signal, `fpga2hps_bridge_rst_n`. The reset manager drives this signal to FPGA-to-HPS bridge on a cold or warm reset.

Related Information

- Clock Manager on page 2-1

- **Reset Manager** on page 3-1
- **HPS Component Interfaces** on page 28-1
  Information about the f2h_axi_clk clock

HPS-to-FPGA Bridge Clocks and Resets

The master interface into the FPGA fabric operates in the `h2f_axi_clk` clock domain. The `h2f_axi_clk` clock is provided by user logic. The slave interface of the bridge in the HPS logic operates in the `l3_main_clk` clock domain. The bridge provides clock crossing logic that allows the logic in the FPGA to operate in any clock domain, asynchronous from the HPS.

The HPS-to-FPGA bridge has one reset signal, `hps2fpga_bridge_rst_n`. The reset manager drives this signal to HPS-to-FPGA bridge on a cold or warm reset.

Related Information

- **Clock Manager** on page 2-1
- **Reset Manager** on page 3-1
- **HPS Component Interfaces** on page 28-1
  Information about the f2h_axi_clk clock

Lightweight HPS-to-FPGA Bridge Clocks and Resets

The master interface into the FPGA fabric operates in the `h2f_lw_axi_clk` clock domain provided by custom logic in the FPGA fabric. The slave interface of the bridge in the HPS logic operates in the `l4_mp_clk` clock domain. The bridge provides clock crossing logic that allows the logic in the FPGA to operate in any clock domain, asynchronous from the HPS.

The lightweight HPS-to-FPGA bridge has one reset signal, `lwhps2fpga_bridge_rst_n`. The reset manager drives this signal to the lightweight HPS-to-FPGA bridge on a cold or warm reset.

Related Information

- **Clock Manager** on page 2-1
- **Reset Manager** on page 3-1
- **HPS Component Interfaces** on page 28-1
  Information about the f2h_axi_clk clock

GPV Clocks

The FPGA-to-HPS and HPS-to-FPGA bridges have GPV slave interfaces, mastered by the lightweight HPS-to-FPGA bridge. These interfaces operate in the `l4_mp_clk` clock domain. Even if you do not use the lightweight HPS-to-FPGA bridge in your FPGA design, you must ensure that a valid `l4_mp_clk` clock is being generated, so that the GPV registers in the HPS-to-FPGA and FPGA-to-HPS bridges can be programmed. The GPV logic in all three bridges is in the `l4_mp_clk` domain.

Related Information

The Global Programmers View on page 8-4

Data Width Sizing

The HPS-to-FPGA and FPGA-to-HPS bridges allow 32-, 64-, and 128-bit interfaces to be exposed to the FPGA fabric. For 32-bit and 128-bit interfaces, the bridge performs data width conversion to the fixed 64-bit interface within the HPS. This conversion is called *upsizing* in the case of data being converted from a 64-bit interface to a 128-bit interface. It is called *downsizing* in the case of data being converted from a

64-bit interface to a 32-bit interface. If an exclusive access is split into multiple transactions, the transactions lose their exclusive access information.

During the upsizing or downsizing process, transactions can also be resized using a data merging technique. For example, in the case of a 32-bit to 64-bit upsizing, if the size of each beat entering the bridge's 32-bit interface is only two bytes, the bridge can merge up to four beats to form a single 64-bit beat. Similarly, in the case of a 128-bit to 64-bit downsizing, if the size of each beat entering the bridge's 128-bit interface is only four bytes, the bridge can merge two beats to form a single 64-bit beat.

The bridges do not perform transaction merging for accesses marked as noncacheable.

Note: You can set the `bypass_merge` bit in the GPV to prevent the bridge from merging data and responses. If the bridge merges multiple responses into a single response, that response is the one with the highest priority. The response types have the following priorities:

1. DECERR
2. SLVERR
3. OKAY

HPS-FPGA Bridges Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- FPGA-to-HPS Bridge Module
- HPS-to-FPGA Bridge Module
- Lightweight HPS-to-FPGA Bridge Module
- FPGA Slaves Accessed via Lightweight HPS-to-FPGA AXI Bridge

Related Information

- **FPGA2HPS AXI Bridge Module Address Map** on page 8-8
- **HPS2FPGA AXI Bridge Module Address Map** on page 8-20
- **LWHPS2FPGA AXI Bridge Module Address Map** on page 8-33
- **FPGA Slaves Accessed Via Lightweight HPS2FPGA AXI Bridge (lwfpgaslaves) Address Map** on page 8-47

This address space is allocated for FPGA-configured slaves driven by the lightweight HPS-to-FPGA bridge master. Address assignment within this space is user-defined. For more information about Lightweight HPS-to-FPGA bridges, refer to the *HPS-FPGA Bridges* chapter of the *Hard Processor System Technical Reference Manual.*

- **HPS Peripheral Region Address Map** on page 1-16
  Lists the base addresses of all modules
- **Cyclone V SoC HPS Address Map and Register Definitions**
  Web-based address map and register definitions

Document Revision History

Table 8-22: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address maps and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.1	Described GPV
January 2012	1.0	Initial release

Cortex-A9 Microprocessor Unit Subsystem 9

2014.06.30

cv_54006 Subscribe Send Feedback

The hard processor system (HPS) in the Altera SoC device includes a stand-alone, full-featured ARM Cortex-A9 MPCore™, single- or dual-core 32-bit application processor. The Cortex-A9 microprocessor unit (MPU) subsystem is composed of a Cortex-A9 MPCore, a level 2 (L2) cache, an Accelerator Coherency Port (ACP) ID mapper, and debugging modules.

Features of the Cortex-A9 MPU Subsystem

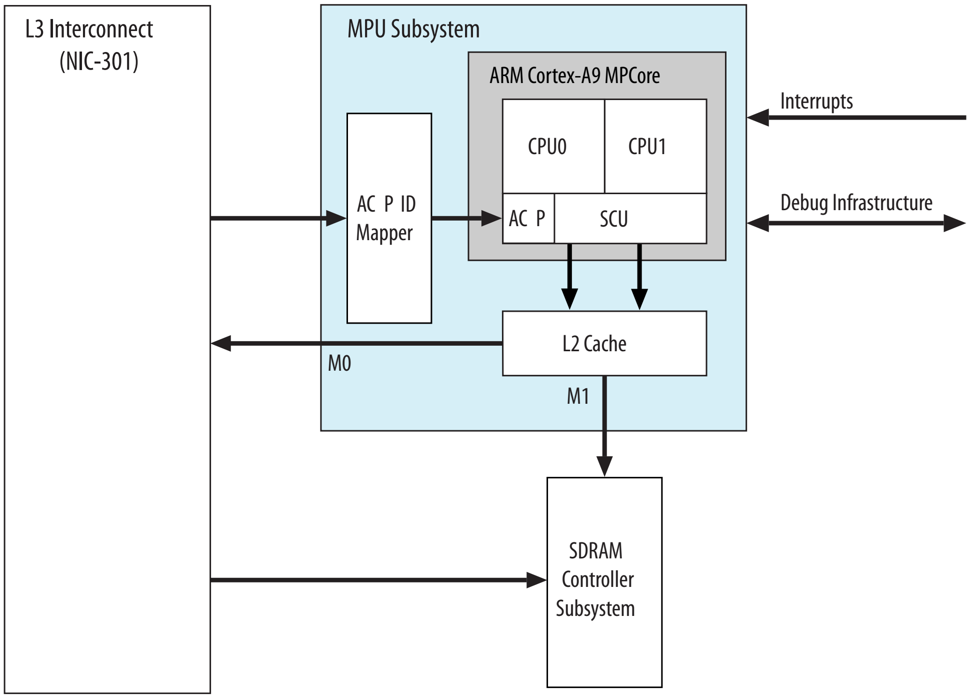
The Altera Cortex-A9 MPU subsystem provides the following features:

- One or two ARM Cortex-A9 processors each with the following support modules:
  - ARM NEON™ single instruction, multiple data (SIMD) coprocesoor
  - MMU
  - 32 KB instruction cache
  - 32 KB data cache
  - Private interval timer
  - Private watchdog timer
- Interrupt controller
- Global timer
- Snoop control unit (SCU)
- Accelerator Coherency Port (ACP)
- ARM L2 cache controller
- TrustZone® system security extensions
- Symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP) modes
- Debugging modules

Cortex-A9 MPU Subsystem Block Diagram and System Integration

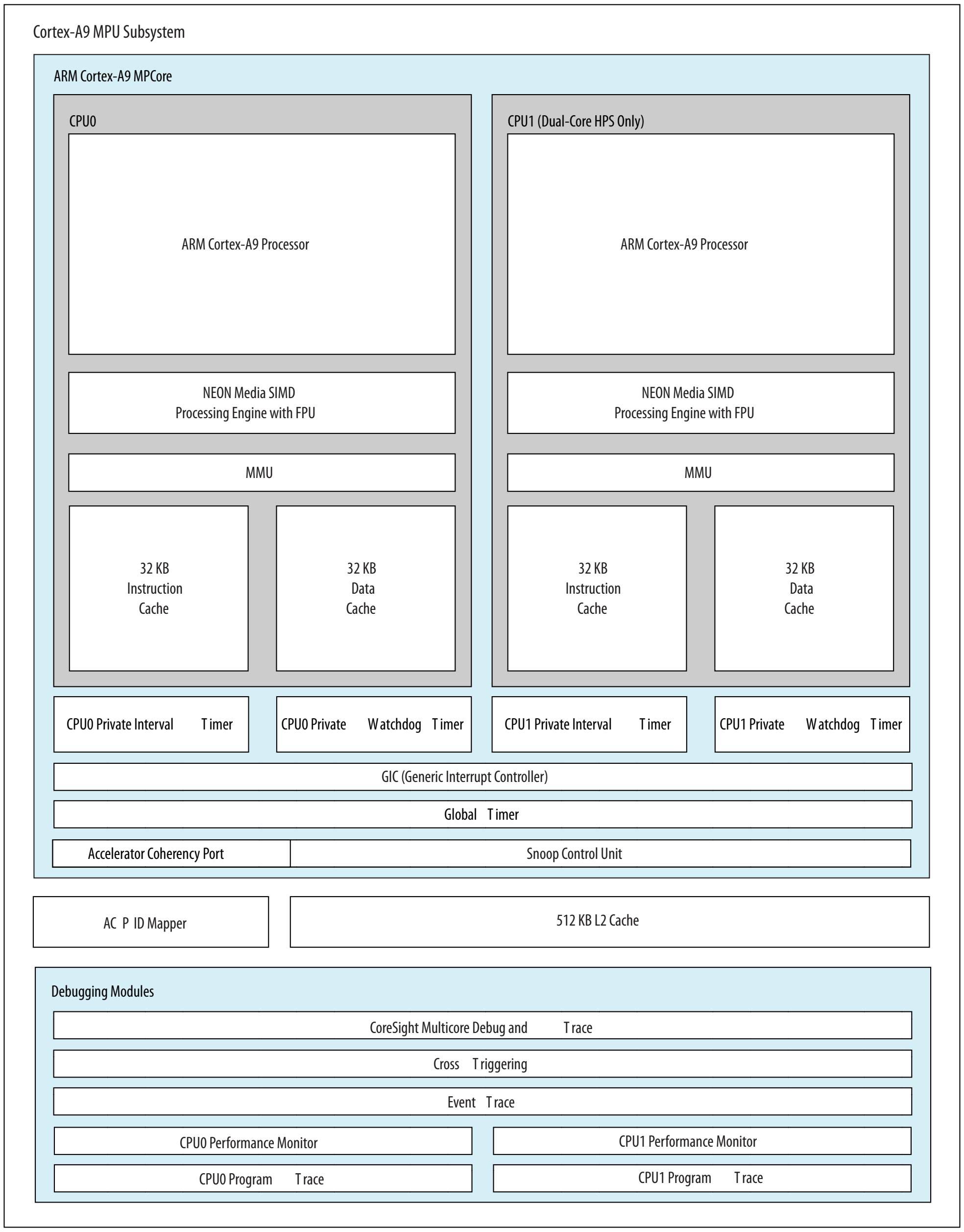
Cortex-A9 MPU Subsystem with L3 Interconnect

This block diagram shows a dual-core MPU subsystem in the context of the HPS, with the L2 cache. The L2 cache can access either the level 3 (L3) interconnect fabric or the SDRAM.



Cortex-A9 MPU Subsystem Internals

This figure shows a block diagram of the Altera Cortex-A9 MPU subsystem.



Cortex-A9 MPCore

The MPU subsystem includes a stand-alone, full-featured ARM Cortex-A9 MPCore single- or dual-core 32-bit application processor. The processor, like other HPS masters, can access IP in the FPGA fabric through the HPS-to-FPGA bridges.

Functional Description

The ARM Cortex-A9 MPCore contains the following blocks:

- One or two Cortex-A9 Revision r3p0 processors operating in SMP or AMP mode
- Snoop control unit (SCU)
- Private interval timer for each processor core
- Private watchdog timer for each processor core
- Global timer
- Interrupt controller

Each transaction originating from the Altera Cortex-A9 MPU subsystem can be flagged as secure or nonsecure.

Implementation Details

Table 9-1: Cortex-A9 MPCore Processor Configuration

This table shows the parameter settings for the Altera Cortex-A9 MPCore.

Feature	Options
Cortex-A9 processors	1 or 2
Instruction cache size per Cortex-A9 processor	32 KB
Data cache size per Cortex-A9 processor	32 KB
TLB size per Cortex-A9 processor	128 entries
Media Processing Engine with NEON™ technology per Cortex-A9 processor[13]	Included
Preload Engine per Cortex-A9 processor	Included
Number of entries in the Preload Engine FIFO per Cortex-A9 processor	16
Jazelle DBX extension per Cortex-A9 processor	Full
PTM interface per Cortex-A9 processor	Included
Support for parity error detection [14]	Included
Master ports	Two
Accelerator Coherency Port	Included

Related Information

http://infocenter.arm.com/

[13] Includes support for floating-point operations.

[14] For a description of the parity error scheme and parity error signals, refer to the Cortex-A9 Technical Reference Manual, available on the ARM website (infocenter.arm.com).

Cortex-A9 Processor

Each Cortex-A9 processor includes the following hardware blocks:

- ARM NEON™ single instruction, multiple data (SIMD) coprocessor with vector floating-point (VFP) v3 double-precision floating point unit for media and signal processing acceleration
  - Single- and double-precision IEEE-754 floating point math support
  - Integer and polynomial math support
- Level 1 (L1) cache with parity checking
  - 32 KB four-way set-associative instruction cache
  - 32 KB four-way set-associative data cache
- CoreSight™ Program Trace Macrocell (PTM) supporting instruction trace

Each Cortex-A9 processor supports the following features:

- Dual-issue superscalar pipeline with advanced branch prediction
- Out-of-order (OoO) dispatch and speculative instruction execution
- 2.5 million instructions per second (MIPS) per MHz, based on the Dhrystone 2.1 benchmark
- 128-entry translation lookaside buffer (TLB)
- TrustZone security extensions
- Configurable data endianness
- Jazelle® DBX Extensions for byte-code dynamic compiler support
- The Cortex-A9 processor architecture supports the following instruction sets:
  - The ARMv7-A performance-optimized instruction set
  - The memory-optimized Thumb®-2 mixed instruction set
    - Improves energy efficiency
    - 31% smaller memory footprint
    - 38% faster than the original Thumb instruction set
  - The Thumb instruction set—supported for legacy applications
- Each processor core in the Altera HPS includes a memory management unit (MMU) to support the memory management requirements of common modern operating systems.

The Cortex-A9 processors are designated CPU0 and CPU1.

Related Information

http://infocenter.arm.com/
Detailed documentation of ARM Cortex-A9 series processors is available on the ARM Infocenter website.

Reset

When a cold or warm reset is issued in the MPU, CPU0 is released from reset automatically, If CPU1 is present, then its reset signals are left asserted when a cold or warm reset is issued. After CPU0 comes out of reset, it can deassert CPU1's reset signals by clearing the `CPU1` bit in the MPU Module Reset (`mpumodrst`) register.

Related Information

Reset Manager on page 3-1

Interactive Debugging Features

Each Cortex-A9 processor has built-in debugging capabilities, including six hardware breakpoints (two with Context ID comparison capability) and four watchpoints. The interactive debugging features can be controlled by external JTAG tools or by processor-based monitor code.

Related Information

http://infocenter.arm.com/

For more information about the interactive debugging system, refer to the *Debug* chapter of the Cortex-A9 Technical Reference Manual, available on the ARM Infocenter website.

L1 Caches

Cache memory that is closely coupled with an associated processor is called level 1, or L1 cache. Each Cortex-A9 processor has two independent 32 KB L1 caches—one for instructions and one for data—allowing simultaneous instruction fetches and data access. Each L1 cache is four-way set associative, with 32 bytes per line, and supports parity checking.

Preload Engine

The preload engine (PLE) is a hardware block that enables the L2 cache to preload selected regions of memory.

The PLE signals the L2 cache when a cache line is needed in the L2 cache, by making the processor data master port start fetching the data. The processor data master does not complete the fetch or return the data to the processor. However, the L2 cache can then proceed to load the cache line. The data is only loaded to the L2 cache, not to the L1 cache or processor registers.

The preload functionality is under software control. The following PLE control parameters must be programmed:

- Programmed parameters, including the following:
  - Base address
  - Length of stride
  - Number of blocks
- A valid bit
- TrustZone memory protection for the cache memory, with an NS (non-secure) state bit
- A translation table base (TTB) address
- An Address Space Identifier (ASID) value

Related Information

http://infocenter.arm.com/

For more information about the PLE, refer to the *Preload Engine* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM Infocenter website.

Floating Point Unit

Each ARM® Cortex-A9 processor includes full support for IEEE-754 floating point operations.

The floating-point unit (FPU) fully supports half-, single-, and double-precision variants of the following operations:

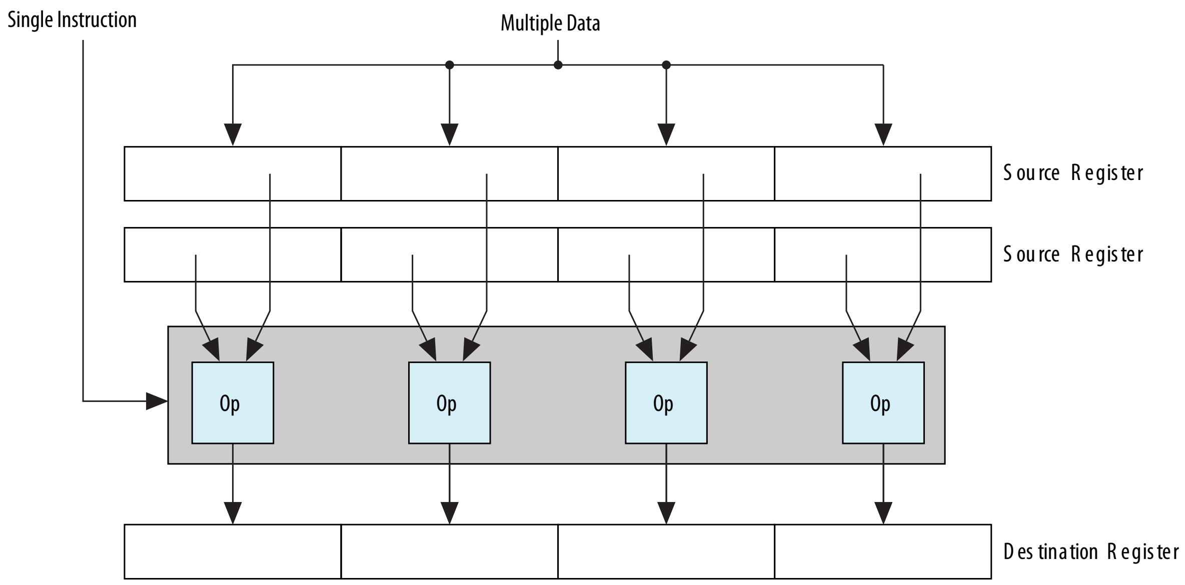
- Add
- Subtract
- Multiply
- Divide
- Multiply and accumulate (MAC)
- Square root

The FPU also converts between floating-point data formats and integers, including special operations to round towards zero required by high-level languages.

NEON Multimedia Processing Engine

The NEON™ multimedia processing engine (MPE) provides hardware acceleration for media and signal processing applications. Each ARM Cortex-A9 processor includes an ARM NEON MPE that supports SIMD processing.

Single Instruction, Multiple Data (SIMD) Processing



Features of the NEON MPE

The NEON™ processing engine accelerates multimedia and signal processing algorithms such as video encoding and decoding, 2-D and 3-D graphics, audio and speech processing, image processing, telephony, and sound synthesis.

The Cortex-A9 NEON MPE performs the following types of operations:

- SIMD and scalar single-precision floating-point computations
- Scalar double-precision floating-point computation
- SIMD and scalar half-precision floating-point conversion
- 8-bit, 16-bit, 32-bit, and 64-bit signed and unsigned integer SIMD computation
- 8-bit or 16-bit polynomial computation for single-bit coefficients

The following operations are available:

- Addition and subtraction
- Multiplication with optional accumulation (MAC)
- Maximum or minimum value driven lane selection operations
- Inverse square root approximation
- Comprehensive data-structure load instructions, including register-bank-resident table lookup

Related Information

http://infocenter.arm.com/
For more information about the Cortex-A9 NEON MPE, refer to the Cortex-A9 NEON™ Media Processing Engine Technical Reference Manual, which you can download from the ARM Infocenter website.

Memory Management Unit

The MMU is used in conjunction with the L1 and L2 caches to translate virtual addresses used by software to physical addresses used by hardware. Each processor has a private MMU.

TLBs Supported By the MMU

TLB Type	Memory Type	Number of Entries	Associativity
Micro TLB	Instruction	32	Fully associative
Micro TLB	Data	32	Fully associative
Main TLB	Instruction and Data	128	Two-way associative

TLB Features

The main TLB has the following features:

- Lockable entries using the lock-by-entry model
- Supports hardware page table walks to perform look-ups in the L1 data cache

The MPU address map is divided into the following regions:

- The boot region
- The SDRAM region
- The FPGA slaves region
- The HPS peripherals region

Related Information

http://infocenter.arm.com/
For more information about the MMU, refer to the *Memory Management Unit* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM Infocenter website.

The Boot Region

The boot region is 1 MB in size, based at address 0. After power-on, or after reset of the L3 interconnect, the boot region is occupied by the boot ROM, allowing the Cortex-A9 MPCore to boot. Although the boot region size is 1 MB, accesses beyond 64 KB are illegal because the boot ROM is only 64 KB.

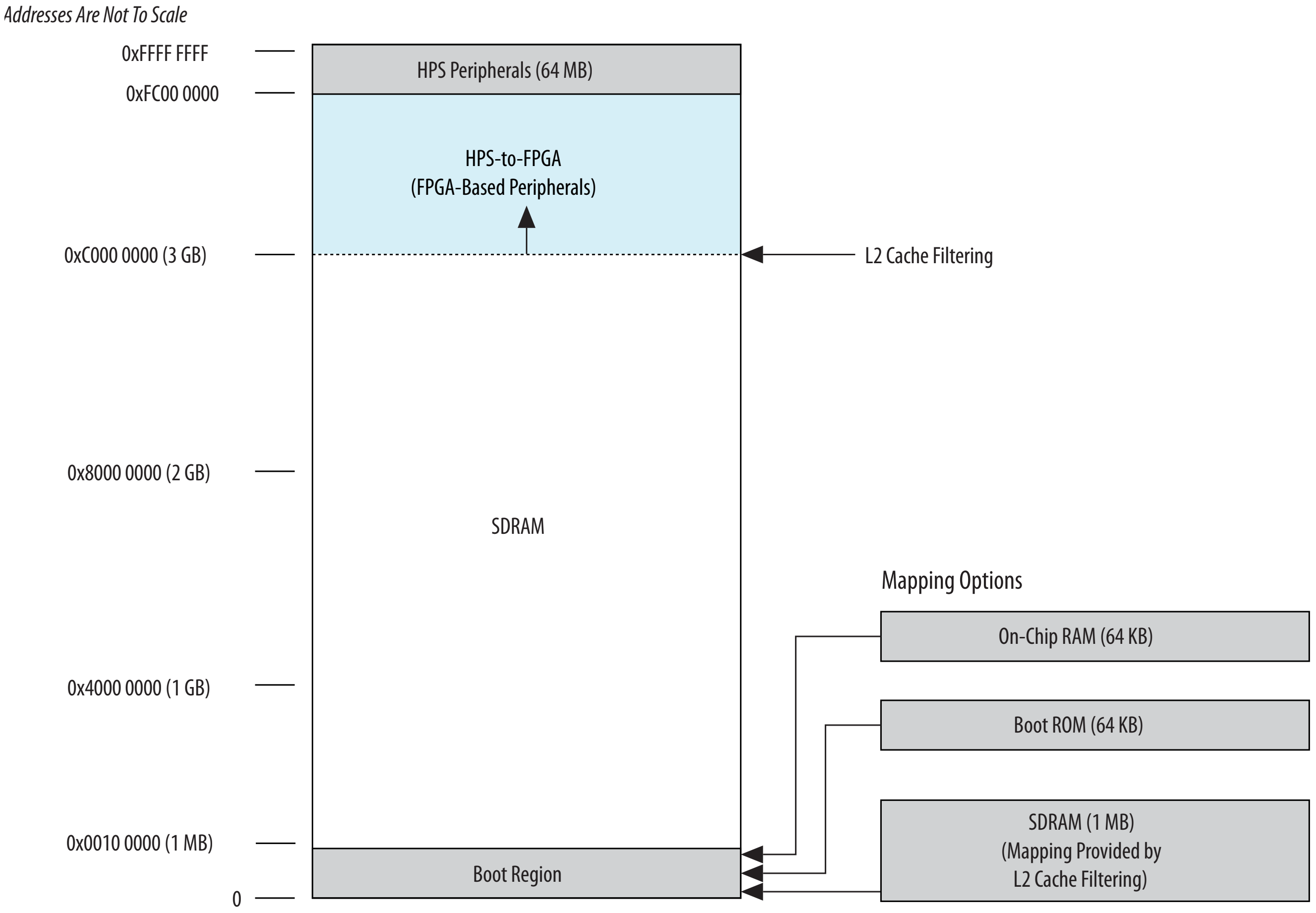
The 1 MB boot region can be subsequently remapped to the bottom 1 MB of SDRAM region.

Note: Alternatively, the boot region can be mapped to the 64 KB on-chip RAM.

Related Information

- **The SDRAM Region** on page 9-11
- **System Interconnect** on page 7-1

The MPCore Address Map



The SDRAM Region

The SDRAM region starts at address `0x100000` (1 MB). The top of the region is determined by the L2 cache filter.

The L2 cache contains a filtering mechanism that routes accesses to the SDRAM and L3 interconnect. The filter defines a filter range with start and end addresses. Any access within this filter range is routed to the SDRAM subsystem. Accesses outside of this filter range are routed to the L3 interconnect.

The start and end addresses are specified in the following register fields:

- `reg12_addr_filtering_start.address_filtering_start`
- `reg12_address_filtering_end.address_filtering_end`

To remap the lower 1MB of SDRAM into the boot region, set the filter start address to `0x0` to ensure accesses between `0x0` and `0xFFFFF` are routed to the SDRAM. Independently, you can set the filter end address in 1 MB increments above `0xC0000000` to extend the upper bounds of the SDRAM region. However, you achieve this extended range at the expense of the FPGA peripheral address span. Depending on the address filter settings in the L2 cache, the top of the SDRAM region can range from `0xBFFFFFFF` to `0xFBFFFFFF`.

Related Information

L2 Cache on page 9-50

The FPGA Slaves Region

The Cortex-A9 MPU subsystem supports the variable-sized FPGA slaves region to communicate with FPGA-based peripherals. This region can start as low as `0xC0000000`, depending on the L2 cache filter settings. The top of the FPGA slaves region is located at `0xFBFFFFFF`. As a result, the size of the FPGA slaves region can range from 0 to `0x3F000000` bytes.

The HPS Peripherals Region

The HPS peripherals region is the top 64 MB in the address space, starting at `0xFC000000` and extending to `0xFFFFFFFF`. The HPS peripherals region is always allocated to the HPS dedicated peripherals for the Altera Cortex-A9 MPU subsystem.

Performance Monitoring Unit

Each Cortex-A9 processor has a Performance Monitoring Unit (PMU). The PMU supports 58 events to gather statistics on the operation of the processor and memory system. Six counters in the PMU accumulate the events in real time. The PMU counters are accessible either from the processor itself, using the Coprocessor 14 (CP14) interface, or from an external debugger. The events are also supplied to the PTM and can be used for trigger or trace.

Related Information

http://infocenter.arm.com/
For more information about the PMU, refer to the *Performance Monitoring Unit* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM Infocenter website.

MPCore Timers

There is one interval timer and one watchdog timer for each processor.

Functional Description

Each timer is private, meaning that only its associated processor can access it. If the watchdog timer is not needed, it can be configured as a second interval timer.

Each private interval and watchdog timer has the following features:

- A 32-bit counter that optionally generates an interrupt when it reaches zero
- Configurable starting values for the counter
- An eight-bit prescaler value to qualify the clock period

Implementation Details

The timers are configurable to either single-shot or auto-reload mode. The timer blocks are clocked by `mpu_periph_clk`, running at ¼ the rate of `mpu_clk`.

Related Information

http://infocenter.arm.com/
For more information about private timers, refer to “About the private timer and watchdog blocks” in the *Global timer, Private timers, and Watchdog registers* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM Infocenter website.

Generic Interrupt Controller

Functional Description

The Generic Interrupt Controller (GIC) supports up to 180 interrupt sources, including dedicated peripherals and IP implemented in the FPGA fabric. In a dual-core system, the GIC is shared by both Cortex-A9 processors. Each processor also has 16 banked software-generated interrupts and 16 banked private peripheral interrupts, which occupy interrupt numbers 0 to 31.

Implementation Details

The configuration and control for the GIC is memory-mapped and accessed through the SCU. The GIC are clocked by `mpu_periph_clk`, running at ¼ the rate of `mpu_clk`.

Related Information

- **GIC Interrupt Map for the Cyclone V SoC HPS** on page 9-13
- **http://infocenter.arm.com/**
  For more information about the GIC, refer to the *Interrupt Controller* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM Infocenter website.

GIC Interrupt Map for the Cyclone V SoC HPS

Note: To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system

Table 9-2: GIC Interrupt Map

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
32	CortexA9_0	`cpu0_parityfail`	This interrupt combines the interrupts named: `cpu0_parityfail_*`.	Edge
33	CortexA9_0	`cpu0_parityfail_BTAC`	—	Edge
34	CortexA9_0	`cpu0_parityfail_GHB`	—	Edge
35	CortexA9_0	`cpu0_parityfail_I_Tag`	—	Edge
36	CortexA9_0	`cpu0_parityfail_I_Data`	—	Edge
37	CortexA9_0	`cpu0_parityfail_TLB`	—	Edge
38	CortexA9_0	`cpu0_parityfail_D_Outer`	—	Edge
39	CortexA9_0	`cpu0_parityfail_D_Tag`	—	Edge

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
40	CortexA9_0	`cpu0_parityfail_D_Data`	—	Edge
41	CortexA9_0	`cpu0_deflags0`	—	Level
42	CortexA9_0	`cpu0_deflags1`	—	Level
43	CortexA9_0	`cpu0_deflags2`	—	Level
44	CortexA9_0	`cpu0_deflags3`	—	Level
45	CortexA9_0	`cpu0_deflags4`	—	Level
46	CortexA9_0	`cpu0_deflags5`	—	Level
47	CortexA9_0	`cpu0_deflags6`	—	Level
48	CortexA9_1	`cpu1_parityfail`	This interrupt combines the interrupts named: `cpu0_parityfail_*`.	Edge
49	CortexA9_1	`cpu1_parityfail_BTAC`	—	Edge
50	CortexA9_1	`cpu1_parityfail_GHB`	—	Edge
51	CortexA9_1	`cpu1_parityfail_I_Tag`	—	Edge
52	CortexA9_1	`cpu1_parityfail_I_Data`	—	Edge
53	CortexA9_1	`cpu1_parityfail_TLB`	—	Edge
54	CortexA9_1	`cpu1_parityfail_D_Outer`	—	Edge
55	CortexA9_1	`cpu1_parityfail_D_Tag`	—	Edge
56	CortexA9_1	`cpu1_parityfail_D_Data`	—	Edge
57	CortexA9_1	`cpu1_deflags0`	—	Level
58	CortexA9_1	`cpu1_deflags1`	—	Level
59	CortexA9_1	`cpu1_deflags2`	—	Level
60	CortexA9_1	`cpu1_deflags3`	—	Level

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
61	CortexA9_1	cpu1_deflags4	—	Level
62	CortexA9_1	cpu1_deflags5	—	Level
63	CortexA9_1	cpu1_deflags6	—	Level
64	SCU	scu_parityfail0	—	Edge
65	SCU	scu_parityfail1	—	Edge
66	SCU	scu_ev_abort	—	Edge
67	L2-Cache	l2_ecc_byte_wr_IRQ	—	Edge
68	L2-Cache	l2_ecc_corrected_IRQ	—	Edge
69	L2-Cache	l2_ecc_uncorrected_IRQ	—	Edge
70	L2-Cache	l2_combined_IRQ	This interrupt combines: DECERRINTR, ECNTRINTR, ERRRDINTR, ERRRTINTR, ERRWDINTR, ERRWTINTR, PARRDINTR, PARRTINTR, and SLVERRINTR.	Level
71	DDR	ddr_ecc_error_IRQ	—	Level
72	FPGA	FPGA_IRQ0	—	Level or Edge
73	FPGA	FPGA_IRQ1	—	Level or Edge
74	FPGA	FPGA_IRQ2	—	Level or Edge
75	FPGA	FPGA_IRQ3	—	Level or Edge
76	FPGA	FPGA_IRQ4	—	Level or Edge
77	FPGA	FPGA_IRQ5	—	Level or Edge
78	FPGA	FPGA_IRQ6	—	Level or Edge
79	FPGA	FPGA_IRQ7	—	Level or Edge
80	FPGA	FPGA_IRQ8	—	Level or Edge

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
81	FPGA	FPGA_IRQ9	—	Level or Edge
82	FPGA	FPGA_IRQ10	—	Level or Edge
83	FPGA	FPGA_IRQ11	—	Level or Edge
84	FPGA	FPGA_IRQ12	—	Level or Edge
85	FPGA	FPGA_IRQ13	—	Level or Edge
86	FPGA	FPGA_IRQ14	—	Level or Edge
87	FPGA	FPGA_IRQ15	—	Level or Edge
88	FPGA	FPGA_IRQ16	—	Level or Edge
89	FPGA	FPGA_IRQ17	—	Level or Edge
90	FPGA	FPGA_IRQ18	—	Level or Edge
91	FPGA	FPGA_IRQ19	—	Level or Edge
92	FPGA	FPGA_IRQ20	—	Level or Edge
93	FPGA	FPGA_IRQ21	—	Level or Edge
94	FPGA	FPGA_IRQ22	—	Level or Edge
95	FPGA	FPGA_IRQ23	—	Level or Edge
96	FPGA	FPGA_IRQ24	—	Level or Edge
97	FPGA	FPGA_IRQ25	—	Level or Edge
98	FPGA	FPGA_IRQ26	—	Level or Edge
99	FPGA	FPGA_IRQ27	—	Level or Edge
100	FPGA	FPGA_IRQ28	—	Level or Edge
101	FPGA	FPGA_IRQ29	—	Level or Edge
102	FPGA	FPGA_IRQ30	—	Level or Edge

GIC Interrup t Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
103	FPGA	FPGA_IRQ31	—	Level or Edge
104	FPGA	FPGA_IRQ32	—	Level or Edge
105	FPGA	FPGA_IRQ33	—	Level or Edge
106	FPGA	FPGA_IRQ34	—	Level or Edge
107	FPGA	FPGA_IRQ35	—	Level or Edge
108	FPGA	FPGA_IRQ36	—	Level or Edge
109	FPGA	FPGA_IRQ37	—	Level or Edge
110	FPGA	FPGA_IRQ38	—	Level or Edge
111	FPGA	FPGA_IRQ39	—	Level or Edge
112	FPGA	FPGA_IRQ40	—	Level or Edge
113	FPGA	FPGA_IRQ41	—	Level or Edge
114	FPGA	FPGA_IRQ42	—	Level or Edge
115	FPGA	FPGA_IRQ43	—	Level or Edge
116	FPGA	FPGA_IRQ44	—	Level or Edge
117	FPGA	FPGA_IRQ45	—	Level or Edge
118	FPGA	FPGA_IRQ46	—	Level or Edge
119	FPGA	FPGA_IRQ47	—	Level or Edge
120	FPGA	FPGA_IRQ48	—	Level or Edge
121	FPGA	FPGA_IRQ49	—	Level or Edge
122	FPGA	FPGA_IRQ50	—	Level or Edge
123	FPGA	FPGA_IRQ51	—	Level or Edge
124	FPGA	FPGA_IRQ52	—	Level or Edge

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
125	FPGA	FPGA_IRQ53	—	Level or Edge
126	FPGA	FPGA_IRQ54	—	Level or Edge
127	FPGA	FPGA_IRQ55	—	Level or Edge
128	FPGA	FPGA_IRQ56	—	Level or Edge
129	FPGA	FPGA_IRQ57	—	Level or Edge
130	FPGA	FPGA_IRQ58	—	Level or Edge
131	FPGA	FPGA_IRQ59	—	Level or Edge
132	FPGA	FPGA_IRQ60	—	Level or Edge
133	FPGA	FPGA_IRQ61	—	Level or Edge
134	FPGA	FPGA_IRQ62	—	Level or Edge
135	FPGA	FPGA_IRQ63	—	Level or Edge
136	DMA	dma_IRQ0	—	Level
137	DMA	dma_IRQ1	—	Level
138	DMA	dma_IRQ2	—	Level
139	DMA	dma_IRQ3	—	Level
140	DMA	dma_IRQ4	—	Level
141	DMA	dma_IRQ5	—	Level
142	DMA	dma_IRQ6	—	Level
143	DMA	dma_IRQ7	—	Level
144	DMA	dma_irq_abort	—	Level
145	DMA	dma_ecc_corrected_IRQ	—	Level
146	DMA	dma_ecc_uncorrected_IRQ	—	Level

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
147	EMAC0	emac0_IRQ	This interrupt combines: sbd_intr_o, lpi_intr_o, and pmt_intr_o.	Level
148	EMAC0	emac0_tx_ecc_corrected_IRQ	—	Level
149	EMAC0	emac0_tx_ecc_uncorrected_IRQ	—	Level
150	EMAC0	emac0_rx_ecc_corrected_IRQ	—	Level
151	EMAC0	emac0_rx_ecc_uncorrected_IRQ	—	Level
152	EMAC1	emac1_IRQ	This interrupt combines: sbd_intr_o, lpi_intr_o, and pmt_intr_o.	Level
153	EMAC1	emac1_tx_ecc_corrected_IRQ	—	Level
154	EMAC1	emac1_tx_ecc_uncorrected_IRQ	—	Level
155	EMAC1	emac1_rx_ecc_corrected_IRQ	—	Level
156	EMAC1	emac1_rx_ecc_uncorrected_IRQ	—	Level
157	USB0	usb0_IRQ	—	Level
158	USB0	usb0_ecc_corrected_IRQ	—	Level
159	USB0	usb0_ecc_uncorrected_IRQ	—	Level
160	USB1	usb1_IRQ	—	Level
161	USB1	usb1_ecc_corrected_IRQ	—	Level
162	USB1	usb1_ecc_uncorrected_IRQ	—	Level
163	CAN0	can0_sts_IRQ	—	Level
164	CAN0	can0_mo_IRQ	—	Level

GIC Interrup t Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
165	CAN0	can0_ecc_corrected_IRQ	—	Level
166	CAN0	can0_ecc_uncorrected_IRQ	—	Level
167	CAN1	can1_sts_IRQ	—	Level
168	CAN1	can1_mo_IRQ	—	Level
169	CAN1	can1_ecc_corrected_IRQ	—	Level
170	CAN1	can1_ecc_uncorrected_IRQ	—	Level
171	SDMMC	sdmmc_IRQ	—	Level
172	SDMMC	sdmmc_porta_ecc_corrected_IRQ	—	Level
173	SDMMC	sdmmc_porta_ecc_uncorrected_IRQ	—	Level
174	SDMMC	sdmmc_portb_ecc_corrected_IRQ	—	Level
175	SDMMC	sdmmc_portb_ecc_uncorrected_IRQ	—	Level
176	NAND	nand_IRQ	—	Level
177	NAND	nandr_ecc_corrected_IRQ	—	Level
178	NAND	nandr_ecc_uncorrected_IRQ	—	Level
179	NAND	nandw_ecc_corrected_IRQ	—	Level
180	NAND	nandw_ecc_uncorrected_IRQ	—	Level
181	NAND	nande_ecc_corrected_IRQ	—	Level
182	NAND	nande_ecc_uncorrected_IRQ	—	Level
183	QSPI	qspi_IRQ	—	Level
184	QSPI	qspi_ecc_corrected_IRQ	—	Level
185	QSPI	qspi_ecc_uncorrected_IRQ	—	Level

GIC Interrupt Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
186	SPI0	spi0_IRQ	This interrupt combines: ssi_txe_intr, ssi_txo_intr, ssi_rxf_intr, ssi_rxo_intr, ssi_rxu_intr, and ssi_mst_intr.	Level
187	SPI1	spi1_IRQ	This interrupt combines: ssi_txe_intr, ssi_txo_intr, ssi_rxf_intr, ssi_rxo_intr, ssi_rxu_intr, and ssi_mst_intr.	Level
188	SPI2	spi2_IRQ	This interrupt combines: ssi_txe_intr, ssi_txo_intr, ssi_rxf_intr, ssi_rxo_intr, ssi_rxu_intr, and ssi_mst_intr.	Level
189	SPI3	spi3_IRQ	This interrupt combines: ssi_txe_intr, ssi_txo_intr, ssi_rxf_intr, ssi_rxo_intr, ssi_rxu_intr, and ssi_mst_intr.	Level
190	I2C0	i2c0_IRQ	This interrupt combines: ic_rx_under_intr, ic_rx_full_intr, ic_tx_over_intr, ic_tx_empty_intr, ic_rd_req_intr, ic_tx_abrt_intr, ic_rx_done_intr, ic_activity_intr, ic_stop_det_intr, ic_start_det_intr, and ic_gen_call_intr.	Level
191	I2C1	i2c1_IRQ	This interrupt combines: ic_rx_under_intr, ic_rx_full_intr, ic_tx_over_intr, ic_tx_empty_intr, ic_rd_req_intr, ic_tx_abrt_intr, ic_rx_done_intr, ic_activity_intr, ic_stop_det_intr, ic_start_det_intr, and ic_gen_call_intr.	Level

GIC Interrup t Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
192	I2C2	i2c2_IRQ	This interrupt combines: ic_rx_under_intr, ic_rx_full_intr, ic_tx_over_intr, ic_tx_empty_intr, ic_rd_req_intr, ic_tx_abrt_intr, ic_rx_done_intr, ic_activity_intr, ic_stop_det_intr, ic_start_det_intr, and ic_gen_call_intr.	Level
193	I2C3	i2c3_IRQ	This interrupt combines: ic_rx_under_intr, ic_rx_full_intr, ic_tx_over_intr, ic_tx_empty_intr, ic_rd_req_intr, ic_tx_abrt_intr, ic_rx_done_intr, ic_activity_intr, ic_stop_det_intr, ic_start_det_intr, and ic_gen_call_intr.	Level
194	UART0	uart0_IRQ	—	Level
195	UART1	uart1_IRQ	—	Level
196	GPIO0	gpio0_IRQ	—	Level
197	GPIO1	gpio1_IRQ	—	Level
198	GPIO2	gpio2_IRQ	—	Level
199	Timer0	timer_l4sp_0_IRQ	This interrupt combines: TIMINT1 and TIMINT2.	Level
200	Timer1	timer_l4sp_1_IRQ	This interrupt combines: TIMINT1 and TIMINT2.	Level
201	Timer2	timer_osc1_0_IRQ	This interrupt combines: TIMINT1 and TIMINT2.	Level
202	Timer3	timer_osc1_1_IRQ	This interrupt combines: TIMINT1 and TIMINT2.	Level
203	Watchdog0	wdog0_IRQ	—	Level
204	Watchdog1	wdog1_IRQ	—	Level

GIC Interrup t Number	Source Block	Interrupt Name	Combined Interrupts	Triggering
205	Clock manager	`clkmgr_IRQ`	—	Level
206	Clock manager	`mpuwakeup_IRQ`	—	Level
207	FPGA manager	`fpga_man_IRQ`	This interrupt combines: `fpga_man_irq[7]` through `fpga_man_irq[0]`.	Level
208	CoreSight	`nCTIIRQ[0]`	—	Level
209	CoreSight	`nCTIIRQ[1]`	—	Level
210	On-chip RAM	`ram_ecc_corrected_IRQ`	—	Level
211	On-chip RAM	`ram_ecc_uncorrected_IRQ`	—	Level

Related Information

Implementation Details on page 9-13

Global Timer

The MPU features a global 64-bit, auto-incrementing timer, which is primarily used by the operating system.

Functional Description

The global timer is accessible by the processors using memory-mapped access through the SCU. The global timer has the following features:

- 64-bit incrementing counter with an auto-incrementing feature. It continues incrementing after sending interrupts.
- Memory-mapped in the private memory region.
- Accessed at reset in Secure State only. It can only be set once, but secure code can read it at any time.
- Accessible to both Cortex-A9 processors in the MPCore.

Implementation Details

Each Cortex-A9 processor has a private 64-bit comparator that generates a private interrupt when the counter reaches the specified value. Each Cortex-A9 processor uses the banked ID, ID27, for this interrupt. ID27 is sent to the GIC as a Private Peripheral Interrupt (PPI).

The global timer are clocked by mpu_periph_clk, running at ¼ the rate of mpu_clk.

Related Information

http://infocenter.arm.com/

For more information about the global timer, refer to "About the Global Timer" in the *Global timer, Private timers, and Watchdog registers* chapter of the Cortex-A9 MPCore Technical Reference Manual, available on the ARM Infocenter website.

Snoop Control Unit

The SCU manages data traffic for the Cortex-A9 processors and the memory system, including the L2 cache. In a multi-master system, the processors and other masters can operate on shared data. The SCU ensures that each processor operates on the most up-to-date copy of data, maintaining cache coherency.

Functional Description

The SCU is used to connect the Cortex-A9 processors and the ACP to the L2 cache controller. The SCU performs the following functions:

- When the processors are set to SMP mode, the SCU maintains data cache coherency between the processors.

  Note: The SCU does not maintain coherency of the instruction caches.
- Initiates L2 cache memory accesses
- Arbitrates between processors requesting L2 access
- Manages ACP access with cache coherency capabilities.

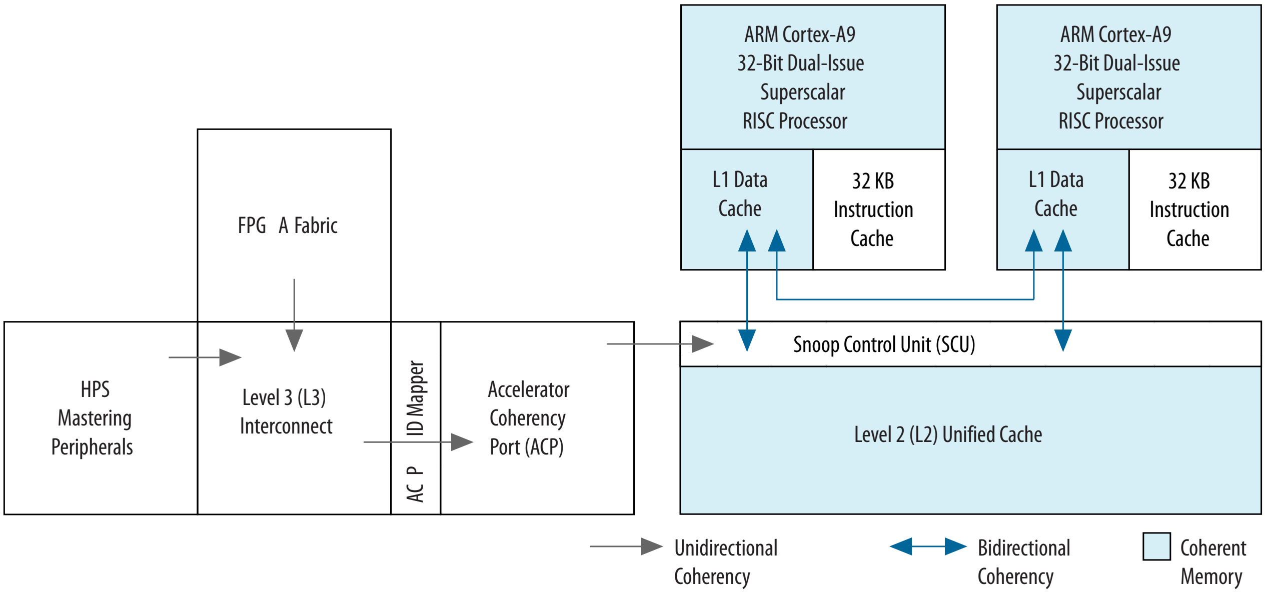
Related Information

http://infocenter.arm.com/

For more information about the SCU, refer to the *Snoop Control Unit* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM Infocenter website.

Coherent Memory, Snoop Control Unit, and Accelerator Coherency Port

The SCU in a dual-processor system, illustrating the flow of data among the L1 data caches and the SCU.


ARM Cortex-A9 32-Bit Dual-Issue Superscalar RISC Processor
L1 Data Cache
32 KB Instruction Cache
ARM Cortex-A9 32-Bit Dual-Issue Superscalar RISC Processor
L1 Data Cache
32 KB Instruction Cache
FPG A Fabric
HPS Mastering Peripherals
Level 3 (L3) Interconnect
ID Mapper
AC P
Accelerator Coherency Port (ACP)
Snoop Control Unit (SCU)
Level 2 (L2) Unified Cache
Unidirectional Coherency
Bidirectional Coherency
Coherent Memory

Related Information

Accelerator Coherency Port on page 9-25

Implementation Details

When the processor writes to any coherent memory location, the SCU ensures that the relevant data is coherent (updated, tagged, or invalidated). Similarly, the SCU monitors read operations from a coherent memory location. If the required data is already stored within the other processor's L1 cache, the data is returned directly to the requesting processor. If the data is not in L1 cache, the SCU issues a read to the L2 cache. If the data is not in the L2 cache memory, the read is finally forwarded to main memory. The primary goal is to minimize power consumption and maximize overall memory performance.

The SCU maintains bidirectional coherency between the L1 data caches belonging to the processors. When one processor writes to a location in its L1 cache, if the same location is cached in the other L1 cache, the SCU updates it.

Non-coherent data passes through as a standard read or write operation.

The SCU also arbitrates between the Cortex-A9 processors if both attempt simultaneous access to the L2 cache, and manages accesses from the ACP.

Accelerator Coherency Port

The ACP allows peripherals—including FPGA-based peripherals—to maintain data coherency with the Cortex-A9 MPCore processors and the SCU. Dedicated peripherals in the HPS, and those built in FPGA logic, access the coherent memory through the ACP ID mapper and the ACP.

Note: The entire 4 GB address space can be accessed coherently through the ACP.

The high-bandwidth peripherals, including the FPGA data ports, connect to the L3 interconnect.

Related Information

- ACP ID Mapper on page 9-26
- Coherent Memory, Snoop Control Unit, and Accelerator Coherency Port on page 9-24

Burst Sizes and Byte Strobes

The ACP improves system performance for hardware accelerators in the FPGA fabric. However, in order to achieve high levels of performance, you must use the one of the recommended burst types. The other burst types have significantly lower performance.

Related Information

Recommended Burst Types on page 9-25

Recommended Burst Types

Table 9-3: Recommended Burst Types

Burst Type	Beats	Width (Bits)	Address Type	Byte Strobes
Wrapping	4	64	64-bit aligned	Asserted
Incrementing	4	64	32-bit aligned	Asserted

Related Information

Burst Sizes and Byte Strobes on page 9-25

Burst Guidelines

Note: If the slave port of the FPGA-to-HPS bridge is not 64 bits wide, you must supply bursts to the FPGA-to-HPS bridge that are upsized or downsized to the burst types above. For example, if the slave data width of the FPGA-to-HPS bridge is 32 bits, then bursts of eight beats by 32 bits are required to access the ACP efficiently.

Note: If the address and burst size of the transaction to the ACP matches either of the conditions above, the logic in the MPU assumes the transaction has all its byte strobes set. If the byte strobes are not all set, then the write does not actually overwrite all the bytes in the word. Instead, the cache assumes the whole cache line is valid. If this line is dirty (and therefore gets written out to SDRAM), data corruption might occur.

Exclusive and Locked Accesses

The ACP does not support exclusive accesses to coherent memory. The ACP supports exclusive accesses to non-coherent memory; however, it is important that the exclusive access transaction is not affected by the upsizing and downsizing logic of the FPGA-to-HPS bridge or the L3 interconnect. If the exclusive access is broken into multiple transactions due to the sizing logic, the exclusive access bit is cleared by the bridge or interconnect and the exclusive access fails.

Note: Altera recommends that exclusive accesses bypass the ACP altogether, either through the 32-bit slave port of the SDRAM controller connected directly to the L3 interconnect or through the FPGA-to-SDRAM interface.

The ACP ID mapper does not support locked accesses. To ensure mutually exclusive access to shared data, use the exclusive access support built into the SDRAM controller.

Related Information

- **Functional Description—HPS Memory Controller** on page 11-1
  For more information about the exclusive access support of the SDRAM controller subsystem, refer to the *SDRAM Controller Subsystem* chapter.
- **Functional Description—HPS Memory Controller** on page 11-1

ACP ID Mapper

The ACP ID mapper is situated between the level 3 (L3) interconnect and the MPU subsystem ACP slave. It is responsible for mapping 12-bit Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) IDs (input IDs) from the L3 interconnect to 3-bit AXI IDs (output IDs) supported by the ACP slave port.

The ACP ID mapper also implements a 1 GB coherent window into 4 GB MPCore.

Functional Description

The ACP slave supports up to six masters. However, custom peripherals implemented in the FPGA fabric can have a larger number of masters that need to access the ACP slave. The ACP ID mapper allows these masters to access the ACP.

The ACP ID mapper resides between the interconnect and the ACP slave of the MPU subsystem. It has the following characteristics:

- Support for up to six concurrent ID mappings
- 1 GB coherent window into 4 GB MPCore address space
- Remaps the 5-bit user sideband signals used by the Snoop Control Unit (SCU) and L2 cache.

Related Information

http://infocenter.arm.com/
For more information about AXI user sideband signals, refer to the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, which you can download from the ARM Infocenter website.

Implementation Details

The ACP is accessed by masters that require access to coherent memory. The ACP slave port can be accessed by the master peripherals of the L3 interconnect, as well as by masters implemented in the FPGA fabric (via the FPGA-to-HPS bridge).

The ACP ID mapper supports the following ID mapping modes:

- Dynamic mapping
- Fixed mapping

Software can select the ID mapping on a per-ID basis. For input IDs that are configured for fixed mapping, there is a one-to-one mapping from input IDs to output IDs. When an input ID is configured for dynamic mapping, it is automatically mapped to an available output ID. The dynamic mode is more flexible because the hardware handles the mapping. The hardware mapping allows you to use one output ID for more than one input ID. Output IDs are assigned to input IDs on a first-come, first-served basis.

Out of the total of eight output IDs, only six are available to masters of the L3 interconnect. The first two output IDs (0 and 1) are dedicated to the Cortex-A9 processor cores in the MPU subsystem, leaving the last six output IDs (2-7) available to the ACP ID mapper. Output IDs 2-6 support fixed and dynamic modes of operation while output ID 7 supports dynamic only.

The operating modes are programmable through accesses to the control and status registers in the ACP ID mapper. At reset time, the ACP ID mapper defaults to dynamic ID mapping for all output IDs except ID 2, which resets to a fixed mapping for the Debug Access Port (DAP) input ID.

Related Information

Cortex-A9 MPU Subsystem with L3 Interconnect on page 9-2
This block diagram shows a dual-core MPU subsystem in the context of the HPS, with the L2 cache. The L2 cache can access either the level 3 (L3) interconnect fabric or the SDRAM.

ID Intended Usage

Table 9-4 summarizes the expected usage of the 3-bit output IDs, and their settings at reset.

Table 9-4: ID Intended Usage

Output ID	Reset State	Intended Use
7	Dynamic	Dynamic mapping only

Output ID	Reset State	Intended Use
6	Dynamic	Fixed or dynamic, programmed by software.
5		
4		
3		
2	Statistically assigned to DAP (ID 0x005)	Assigned to the input ID of the DAP at reset. After reset, can be either fixed or dynamic, programmed by software.
1	—	Not used by the ACP ID Mapper
0		

AXI User Sideband Override

For masters that cannot drive the AXI user sideband signal of incoming transactions, the ACP ID mapper can control overriding this signal. The ACP ID mapper can also control which 1 GB coherent window into memory is accessed by masters of the L3 interconnect. Each fixed mapping can be assigned a different user sideband signal and memory window to allow specific settings for different masters. All dynamic mappings share a common user sideband signal and memory window setting.

Transaction Capabilities

At any one time, the ACP ID mapper can accept and issue up to 15 transactions per ID mapping. Read and write ID mappings are managed in separate lists, allowing more unique input IDs to be remapped at any given time. If a master issues a series of reads and writes with the same input ID, there are no ordering restrictions.

Because there are only six output IDs available, there can be no more than six read and six write transactions with unique IDs in progress at any one time. The write acceptance of the ACP slave is five transactions, and the read acceptance is 13 transactions. Only four coherent read transactions per ID mapping can be outstanding at one time.

Dynamic Mapping Mode

In dynamic mode, every unique input ID that is received from the L3 master port is assigned to an unused output ID. The new output ID is applied to the transaction as it is issued to the ACP slave of the SCU. Any transaction that arrives to the ACP ID mapper with an input ID that matches an already-in-progress transaction is mapped to the same output ID. Once all transactions on an ID mapping have completed, that output ID is released and can be used again for other input IDs.

Fixed Mapping Mode

In fixed mode, output IDs 2 through 6 can be assigned by software to a specific 12-bit input ID. This ability makes it possible to use the lock-by-master feature of the L2 cache controller, because the input transaction ID from the master is always assigned to a specific output ID. Unlike dynamic mode, ID 7 is not available for fixed mapping because it is reserved for dynamic mode only to avoid system deadlocks.

The ACP ID mapper has two banks of registers to control the behavior of the mappings, namely, a request bank and a read-only status bank. Both banks contain the same number of registers. To change the settings for a particular mapping (either a specific fixed ID, or all dynamic mappings), software should write to the appropriate register in the request bank. The hardware examines the request, and only applies the change when safe to do so, which is when there are no outstanding transactions with the output ID. When the change is applied, the status register is updated. Software should check that the change has actually taken place by polling the corresponding status register.

HPS Peripheral Master Input IDs

Table 9-5: HPS Peripheral Master Input IDs

The input IDs issued from the interconnect for each HPS peripheral master that can access the ACP ID mapper

Interconnect Master	ID [15]
L2M0	`0xxx xxxx x010`
DMA	`0000 0xxx x001`
EMAC0	`1000 0xxx x001`
EMAC1	`1000 0xxx x010`
USB0	`1000 0000 0011`
USB1	`1000 0000 0110`
NAND	`1xxx xxxx x100`
ETR	`1000 0000 0000`
DAP	`0000 0000 0100`
SD/MMC	`1000 0000 0101`
FPGA-to-HPS bridge	`0xxx xxxx x000`

Control of the AXI User Sideband Signals

The ACP ID mapper module allows control of the AXI user sideband signal values. Not all masters drive these signals, so the ACP ID mapper makes it possible to drive the 5-bit user sideband signal with either a default value (in dynamic mode) or specific values (in fixed mode). The sideband signals are controlled

[15] Values are in binary. The letter *x* denotes variable ID bits each master passes with each transaction.

through the ACP port. Sideband signals `AWUSER[4:1]` convey the inner cache attributes and `AWUSER[0]` determines whether the transaction is a shared cache access.

There are registers available to configure the default values of the user sideband signals for all transactions, and fixed values of these signals for particular transactions in fixed mapping mode. In dynamic mode, the user sideband signals of incoming transactions are mapped with the default values stored in the register. In fixed mapping mode, the input ID of the transaction is mapped to the 3-bit output ID and the user sideband signals of the transaction are mapped with the values stored in the register that corresponds to the output ID. One important exception, however, is that the ACP ID mapper always allows user sideband signals from the FPGA-to-HPS bridge to pass through to the ACP regardless of the user sideband value associated with the ID.

Memory Region Remap

The ACP ID mapper has 1 GB of address space, which is by default a view into the bottom 1 GB of SDRAM. The mapper also allows transactions to be routed to different 1 GB-sized memory regions, called pages, in both dynamic and fixed modes. The two most significant bits of incoming 32-bit AXI address signals are replaced with the 2-bit user-configured address page decode information. The page decoder uses the values shown in **Table 9-6**.

Table 9-6: Page Decoder Values

Page	Address Range
0	`0x00000000–0x3FFFFFFF`
1	`0x40000000–0x7FFFFFFF`
2	`0x80000000–0xBFFFFFFF`
3	`0xC0000000–0xFFFFFFFF`

With this page decode information, a master can read or write to any 1 GB region of the 4 GB memory space while maintaining cache coherency with the MPU subsystem.

Using this feature, a debugger can have a coherent view into main memory, without having to stop the processor. For example, at reset the DAP input ID (0x001) is mapped to output ID 2, so the debugger can vary the 1 GB window that the DAP accesses without affecting any other traffic flow to the ACP.

ACP ID Mapper Address Map and Register Definitions

This section lists the ACP ID Mapper register address map and describes the registers.

ACP ID Mapper Registers Address Map

Registers in the ACP ID Mapper module

Base Address: `0xFF707000`

ACP ID Mapper Registers

Register	Offset	Width	Acces s	Reset Value	Description
vid2rd on page 9-32	0x0	32	RW	0x80040010	Read AXI Master Mapping Register for Fixed Virtual ID 2
vid2wr on page 9-33	0x4	32	RW	0x80040010	Write AXI Master Mapping Register for Fixed Virtual ID 2
vid3rd on page 9-33	0x8	32	RW	0x0	Read AXI Master Mapping Register for Fixed Virtual ID 3
vid3wr on page 9-34	0xC	32	RW	0x0	Write AXI Master Mapping Register for Fixed Virtual ID 3
vid4rd on page 9-35	0x10	32	RW	0x0	Read AXI Master Mapping Register for Fixed Virtual ID 4
vid4wr on page 9-36	0x14	32	RW	0x0	Write AXI Master Mapping Register for Fixed Virtual ID 4
vid5rd on page 9-37	0x18	32	RW	0x0	Read AXI Master Mapping Register for Fixed Virtual ID 5
vid5wr on page 9-37	0x1C	32	RW	0x0	Write AXI Master Mapping Register for Fixed Virtual ID 5
vid6rd on page 9-38	0x20	32	RW	0x0	Read AXI Master Mapping Register for Fixed Virtual ID 6
vid6wr on page 9-39	0x24	32	RW	0x0	Write AXI Master Mapping Register for Fixed Virtual ID 6
dynrd on page 9-40	0x28	32	RW	0x0	Read AXI Master Mapping Register for Dynamic Virtual ID Remap
dynwr on page 9-40	0x2C	32	RW	0x0	Write AXI Master Mapping Register for Dynamic Virtual ID Remap
vid2rd_s on page 9-41	0x30	32	RO	0x80040010	Read AXI Master Mapping Status Register for Fixed Virtual ID 2
vid2wr_s on page 9-42	0x34	32	RO	0x80040010	Write AXI Master Mapping Status Register for Fixed Virtual ID 2
vid3rd_s on page 9-43	0x38	32	RO	0x0	Read AXI Master Mapping Status Register for Fixed Virtual ID 3
vid3wr_s on page 9-43	0x3C	32	RO	0x0	Write AXI Master Mapping Status Register for Fixed Virtual ID 3
vid4rd_s on page 9-44	0x40	32	RO	0x0	Read AXI Master Mapping Status Register for Fixed Virtual ID 4
vid4wr_s on page 9-45	0x44	32	RO	0x0	Write AXI Master Mapping Status Register for Fixed Virtual ID 4
vid5rd_s on page 9-46	0x48	32	RO	0x0	Read AXI Master Mapping Status Register for Fixed Virtual ID 5

Register	Offset	Width	Access	Reset Value	Description
vid5wr_s on page 9-46	0x4C	32	RO	0x0	Write AXI Master Mapping Status Register for Fixed Virtual ID 5
vid6rd_s on page 9-47	0x50	32	RO	0x0	Read AXI Master Mapping Status Register for Fixed Virtual ID 6
vid6wr_s on page 9-48	0x54	32	RO	0x0	Write AXI Master Mapping Status Register for Fixed Virtual ID 6
dynrd_s on page 9-49	0x58	32	RO	0x0	Read AXI Master Mapping Status Register for Dynamic Virtual ID Remap
dynwr_s on page 9-49	0x5C	32	RO	0x0	Write AXI Master Mapping Status Register for Dynamic Virtual ID Remap

vid2rd

The Read AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707000

Offset: 0x0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RW 0x1</td><td colspan="3">Reserved</td><td colspan="12">mid
RW 0x4</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RW 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RW 0x1</td><td colspan="4">Reserved</td></tr>
</table>

vid2rd Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x1
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x4

Bit	Name	Description	Access	Reset
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	`RW`	`0x0`
8:4	`user`	This value is propagated to SCU as ARUSERS.	`RW`	`0x1`

vid2wr

The Write AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707004`

Offset: `0x4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x1	Reserved			mid RW 0x4											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x1					Reserved			

vid2wr Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	`RW`	`0x1`
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	`RW`	`0x4`
13:12	`page`	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	`RW`	`0x0`
8:4	`user`	This value is propagated to SCU as AWUSERS.	`RW`	`0x1`

vid3rd

The Read AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x0	Reserved			mid RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

vid3rd Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RW	0x0

vid3wr

The Write AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF70700C

Offset: 0xC

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RW 0x0</td><td colspan="3">Reserved</td><td colspan="12">mid
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RW 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RW 0x0</td><td colspan="4">Reserved</td></tr>
</table>

vid3wr Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	`page`	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	`user`	This value is propagated to SCU as AWUSERS.	RW	0x0

vid4rd

The Read AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707010`

Offset: `0x10`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RW 0x0</td><td colspan="3">Reserved</td><td colspan="12">mid
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RW 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RW 0x0</td><td colspan="4">Reserved</td></tr>
</table>

vid4rd Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	`user`	This value is propagated to SCU as ARUSERS.	RW	0x0

vid4wr

The Write AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707014`

Offset: `0x14`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x0	Reserved			mid RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

vid4wr Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0

Bit	Name	Description	Access	Reset
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RW	0x0

vid5rd

The Read AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707018

Offset: 0x18

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RW 0x0</td><td colspan="3">Reserved</td><td colspan="12">mid
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RW 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RW 0x0</td><td colspan="4">Reserved</td></tr>
</table>

vid5rd Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RW	0x0

vid5wr

The Write AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF70701C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x0	Reserved			mid RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

vid5wr Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RW	0x0

vid6rd

The Read AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x0	Reserved			mid RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

vid6rd Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	`user`	This value is propagated to SCU as ARUSERS.	RW	0x0

vid6wr

The Write AXI Master Mapping Register contains the USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707024

Offset: `0x24`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RW 0x0	Reserved			mid RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

vid6wr Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RW	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RW	0x0
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RW	0x0

dynrd

The Read AXI Master Mapping Register contains the USER, and ADDR page signals mapping values for transaction that dynamically remapped to one of the available 3-bit virtual IDs.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707028

Offset: 0x28

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RW 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RW 0x0</td><td colspan="4">Reserved</td></tr>
</table>

dynrd Fields

Bit	Name	Description	Access	Reset
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RW	0x0

dynwr

The Write AXI Master Mapping Register contains the USER, and ADDR page signals mapping values for transaction that dynamically remapped to one of the available 3-bit virtual IDs.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF70702C

Offset: 0x2C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RW 0x0		Reserved			user RW 0x0					Reserved			

dynwr Fields

Bit	Name	Description	Access	Reset
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RW	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RW	0x0

vid2rd_s

The Read AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707030

Offset: 0x30

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x1	Reserved			mid RO 0x4											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x1					Reserved			

vid2rd_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x1

Bit	Name	Description	Access	Reset
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x4
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RO	0x1

vid2wr_s

The Write AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707034

Offset: 0x34

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x1	Reserved			mid RO 0x4											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x1					Reserved			

vid2wr_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x1
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x4
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RO	0x1

vid3rd_s

The Read AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707038

Offset: 0x38

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid3rd_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RO	0x0

vid3wr_s

The Write AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF70703C

Offset: 0x3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid3wr_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RO	0x0

vid4rd_s

The Read AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707040

Offset: 0x40

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid4rd_s Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	`user`	This value is propagated to SCU as ARUSERS.	RO	0x0

vid4wr_s

The Write AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707044`

Offset: `0x44`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RO 0x0</td><td colspan="3">Reserved</td><td colspan="12">mid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RO 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RO 0x0</td><td colspan="4">Reserved</td></tr>
</table>

vid4wr_s Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0

Bit	Name	Description	Access	Reset
13:12	`page`	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	`RO`	`0x0`
8:4	`user`	This value is propagated to SCU as AWUSERS.	`RO`	`0x0`

vid5rd_s

The Read AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707048`

Offset: `0x48`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>force
RO 0x0</td><td colspan="3">Reserved</td><td colspan="12">mid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RO 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RO 0x0</td><td colspan="4">Reserved</td></tr>
</table>

vid5rd_s Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	`RO`	`0x0`
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	`RO`	`0x0`
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	`RO`	`0x0`
8:4	`user`	This value is propagated to SCU as ARUSERS.	`RO`	`0x0`

vid5wr_s

The Write AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF70704C

Offset: 0x4C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid5wr_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RO	0x0

vid6rd_s

The Read AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707050

Offset: 0x50

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid6rd_s Fields

Bit	Name	Description	Access	Reset
31	force	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	mid	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	page	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as ARUSERS.	RO	0x0

vid6wr_s

The Write AXI Master Mapping Status Register contains the configured USER, ADDR page, and ID signals mapping values for particular transaction with 12-bit ID which locks the fixed 3-bit virtual ID.

Module Instance	Base Address	Register Address
acpidmap	0xFF707000	0xFF707054

Offset: 0x54

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
force RO 0x0	Reserved			mid RO 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

vid6wr_s Fields

Bit	Name	Description	Access	Reset
31	`force`	Set to 1 to force the mapping between the 12-bit ID and 3-bit virtual ID N. Set to 0 to allow the 3-bit ID N to be dynamically allocated.	RO	0x0
27:16	`mid`	The 12-bit ID of the master to remap to 3-bit virtual ID N, where N is the 3-bit ID to use.	RO	0x0
13:12	`page`	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	`user`	This value is propagated to SCU as AWUSERS.	RO	0x0

dynrd_s

The Read AXI Master Mapping Status Register contains the configured USER, and ADDR page signals mapping values for transaction that dynamically remapped to one of the available 3-bit virtual IDs.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF707058`

Offset: `0x58`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">page
RO 0x0</td><td colspan="3">Reserved</td><td colspan="5">user
RO 0x0</td><td colspan="4">Reserved</td></tr>
</table>

dynrd_s Fields

Bit	Name	Description	Access	Reset
13:12	`page`	ARADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	`user`	This value is propagated to SCU as ARUSERS.	RO	0x0

dynwr_s

The Write AXI Master Mapping Status Register contains the configured USER, and ADDR page signals mapping values for transaction that dynamically remapped to one of the available 3-bit virtual IDs.

Module Instance	Base Address	Register Address
`acpidmap`	`0xFF707000`	`0xFF70705C`

Offset: `0x5C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		page RO 0x0		Reserved			user RO 0x0					Reserved			

dynwr_s Fields

Bit	Name	Description	Access	Reset
13:12	page	AWADDR remap to 1st, 2nd, 3rd, or 4th 1GB memory region.	RO	0x0
8:4	user	This value is propagated to SCU as AWUSERS.	RO	0x0

L2 Cache

The MPU subsystem includes a secondary 512 KB L2 shared, unified cache memory.

Functional Description

The L2 cache is much larger than the L1 cache. The L2 cache has significantly lower latency than external memory. The L2 cache is up to eight-way associative, configurable down to one-way (direct mapped). Like the L1 cache, the L2 cache can be locked by cache line, locked by way, or locked by bus master.

The L2 cache implements error correction codes (ECCs) and ECC error reporting. The cache can report a number of events to the processor and operating system.

Cache Controller Configuration

The L2 cache consists of the ARM L2C-310 L2 cache controller configured as follows:

- 512 KB total memory
- Eight-way associativity
- Physically addressed, physically tagged
- Line length of 32 bytes
- Critical first word linefills
- Support for all AXI cache modes
    - Write-through
    - Write-back
    - Read allocate
    - Write allocate
    - Read and write allocate

- Single event upset (SEU) protection
  - Parity on Tag RAM
  - ECC on L2 Data RAM
- Two slave ports mastered by the SCU
- Two master ports connected to the following slave ports:
  - SDRAM controller, 64-bit slave port width
  - L3 interconnect, 64-bit slave port width
- Cache lockdown capabilities as follows:
  - Line lockdown
  - Lockdown by way
  - Lockdown by master (both processors and ACP masters)
- TrustZone support
- Cache event monitoring

Related Information

- **L2 Cache Event Monitoring** on page 9-53
- **System Manager** on page 5-1
  For more information about SEU errors, refer to the *System Manager* chapter.
- **http://infocenter.arm.com/**
  For more information about AXI user sideband signals, refer to the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, which you can download from the ARM Infocenter website.

L2 Cache Address Filtering

The L2 cache can access either the L3 interconnect fabric or the SDRAM. The L2 cache address filtering determines how much address space is allocated to the HPS-to-FPGA bridge and how much is allocated to SDRAM, depending on the configuration of the memory management unit.

Related Information

- **Cortex-A9 MPU Subsystem with L3 Interconnect** on page 9-2
This block diagram shows a dual-core MPU subsystem in the context of the HPS, with the L2 cache. The L2 cache can access either the level 3 (L3) interconnect fabric or the SDRAM.
- **Memory Management Unit** on page 9-9
  Information about how the address space is set based on L2 cache address filtering.

ECC Support

The L2 cache has the option of using ECCs to protect against SEU errors in the cache RAM.

Enabling ECCs does not affect the performance of the L2 cache. The ECC bits are calculated only for writes to the data RAM that are 64 bits wide (8 bytes, or one-quarter of the cache line length). The ECC logic does not perform a read-modify-write when calculating the ECC bits.

The ECC protection bits are not valid in the following cases:

- Data is written that is not 64-bit aligned in memory
- Data is written that is less than 64 bits in width

In these cases the Byte Write Error interrupt is asserted. Cache data is still written when such an error occurs. However, the ECC error detection and correction continues to function. Therefore, the cache data is likely to be incorrect on subsequent reads.

To use ECCs, the software and system must meet the following requirements:

- L1 and L2 cache must be configured as write-back and write-allocate for any cacheable memory region
- Level 3 interconnect masters using the ACP must only perform the following types of data writes:
  - 64-bit aligned in memory
  - 64-bit wide accesses

  Note that L3 interconnect masters can include masters in the FPGA accessing the FPGA-to-HPS bridge.

If a correctable ECC error occurs, the ECC corrects the read data in parallel to asserting a correctable error signal on the AXI bus. If an uncorrectable error occurs in the L2 cache, the uncorrected data remains in the L2 cache and an AXI slave error (SLVERR) is sent to the L1 memory system. In addition, interrupts can be enabled for correctable and uncorrectable ECC errors through the GIC.

Related Information

System Manager on page 5-1
For more information about SEU errors, refer to the *System Manager* chapter.

Implementation Details

The following table shows the parameter settings for the cache controller.

Table 9-7: Cache Controller Configuration

Feature	Meaning
Cache way size	64 KB
Number of cache ways	8 ways
Tag RAM write latency	1
Tag RAM read latency	1
Tag RAM setup latency	1
Data RAM write latency	1
Data RAM read latency	2
Data RAM setup latency	1
Parity logic	Parity logic enabled
Lockdown by master	Lockdown by master enabled

Feature	Meaning
Lockdown by line	Lockdown by line enabled
AXI ID width on slave ports	6 AXI ID bits on slave ports
Address filtering	Address filtering logic enabled
Speculative read	Logic for supporting speculative read enabled
Presence of ARUSERMx and AWUSERMx sideband signals	Sideband signals enabled

Related Information

http://infocenter.arm.com/

For further information about cache controller configurable options, refer to the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, available on the ARM Infocenter website.

L2 Cache Lockdown Capabilities

The L2 cache has three methods to lock data in the cache RAMs:

- Lockdown by line—Used to lock lines in the cache. This is commonly used for loading critical sections of software into the cache temporarily.
- Lockdown by way—Allows any or all of the eight cache ways to be locked. This is commonly used for loading critical data or code into the cache.
- Lockdown by master—Allows cache ways to be dedicated to a single master port. This allows a large cache to look like smaller caches to multiple master ports. The L2 cache can be mastered by CPU0, CPU1, or the six ACP masters, for a total of eight possible master ports.

Related Information

http://infocenter.arm.com/

For more information about L2 cache lockdown capabilities, refer to “Cache operation” in the *Functional Overview* chapter of the *CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual*, available on the ARM Infocenter website.

L2 Cache Event Monitoring

The L2 cache supports the built-in cache event monitoring signals shown in the table below. The L2 cache can count two of the events at any one time.

Table 9-8: L2 Cache Events

Event	Description
`CO`	Eviction (cast out) of a line from the L2 cache.
`DRHIT`	Data read hit in the L2 cache.

Event	Description
DRREQ	Data read lookup to the L2 cache. Subsequently results in a hit or miss.
DWHIT	Data write hit in the L2 cache.
DWREQ	Data write lookup to the L2 cache. Subsequently results in a hit or miss.
DWTREQ	Data write lookup to the L2 cache with write-through attribute. Subsequently results in a hit or miss.
EPFALLOC	Prefetch hint allocated into the L2 cache.
EPFHIT	Prefetch hint hits in the L2 cache.
EPFRCVDS0	Prefetch hint received by slave port S0.
EPFRCVDS1	Prefetch hint received by slave port S1.
IPFALLOC	Allocation of a prefetch generated by L2 cache controller into the L2 cache.
IRHIT	Instruction read hit in the L2 cache.
IRREQ	Instruction read lookup to the L2 cache. Subsequently results in a hit or miss.
SPNIDEN	Secure privileged non-invasive debug enable.
SRCONFS0	Speculative read confirmed in slave port S0.
SRCONFS1	Speculative read confirmed in slave port S1.
SRRCVDS0	Speculative read received by slave port S0.
SRRCVDS1	Speculative read received by slave port S1.
WA	Allocation into the L2 cache caused by a write (with write-allocate attribute) miss.

In addition, the L2 cache events can be captured and timestamped using dedicated debugging circuitry.

Related Information

http://infocenter.arm.com/
For more information about L2 event capture, refer to the *Debug* chapter of the Cortex-A9 MPCore Technical Reference Manual, available on the ARM Infocenter website.

Debugging Modules

The MPU subsystem includes debugging resources through ARM CoreSight on-chip debugging and trace. The following functionality is included:

- Individual program trace for each processor
- Event trace for the Cortex-A9 MPCore
- Cross triggering between processors and other HPS debugging features

Program Trace

Each processor has an independent PTM that provides real-time instruction flow trace. The PTM is compatible with a number of third-party debugging tools.

The PTM provides trace data in a highly compressed format. The trace data includes tags for specific points in the program execution flow, called waypoints. Waypoints are specific events or changes in the program flow.

The PTM recognizes and tags the waypoints listed in **Table 9-9**.

Table 9-9: Waypoints Supported by the PTM

Type	Additional Waypoint Information
Indirect branches	Target address and condition code
Direct branches	Condition code
Instruction barrier instructions	—
Exceptions	Location where the exception occurred
Changes in processor instruction set state	—
Changes in processor security state	—
Context ID changes	—
Entry to and return from debug state when Halting debug mode is enabled	—

The PTM optionally provides additional information for waypoints, including the following.

- Processor cycle count between waypoints
- Global timestamp values
- Target addresses for direct branches

Related Information

- **CoreSight Debug and Trace** on page 10-1

- **http://infocenter.arm.com/**
  For more information about the PTM, refer to the CoreSight PTM-A9 Technical Reference Manual, available on the ARM Infocenter website.

Event Trace

Events from each processor can be used as inputs to the PTM. The PTM can use these events as trace and trigger conditions.

Related Information

- **Performance Monitoring Unit** on page 9-12
- **http://infocenter.arm.com/**
  For more information about the trigger and trace capabilities, refer to the CoreSight PTM-A9 Technical Reference Manual, Revision r1p0, available on the ARM Infocenter website.

Cross-Triggering

The PTM can export trigger events and perform actions on trigger inputs. The cross-trigger signals interface with other HPS debugging components including the FPGA fabric. Also, a breakpoint in one processor can trigger a break in the other.

Related Information

CoreSight Debug and Trace on page 10-1
For detailed information about cross-triggering and about debugging hardware in the MPU, refer to the *CoreSight Debug and Trace* chapter.

Clocks

Four synchronous clocks and two debug clocks are provided to the MPU subsystem.

Table 9-10: MPU Subsystem Clocks

System Clock Name	Use
`mpu_clk`	Main clock for the MPU subsystem
`mpu_periph_clk`	Clock for peripherals inside the MPU subsystem
`mpu_l2_ram_clk`	Clock for the L2 cache and Accelerator Coherency Port (ACP) ID mapper
`l4_mp_clk`	Clock for the ACP ID mapper control slave
`dbg_at_clk`	Trace bus clock
`dbg_clk`	Debug clock

Cortex-A9 MPU Subsystem Register Implementation

The following configurations are available through registers in the Cortex-A9 subsystem:

- All processor-related controls, including the MMU and L1 caches, are controlled using the Coprocessor 15 (CP15) registers of each individual processor.
- All SCU registers, including control for the timers and GIC, are memory mapped.
- All L2 cache registers are memory-mapped.

Related Information

- **Functional Description—HPS Memory Controller** on page 11-1
  For an address map of peripheral slave ports, including the SCU and L2 cache, refer to the *Introduction to the Hard Processor System* chapter.
- **http://infocenter.arm.com/**
  For detailed definitions of the registers for the Altera Cortex-A9 MPU subsystem, refer to the Cortex-A9 MPCore Technical Reference Manual, Revision r3p0 and the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, Revision r3p3, available on the ARM Infocenter website.

Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Added Reset Section to Cortex-A9 Processor • Updated HPS Peripheral Master Input IDs table • Added ACP ID Mapper Address Map and Register Definitions • Added information in ECC Support section regarding ECC errors • Minor clarifications regarding MPU description and module revision numbers
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Correct SDRAM region address in MPCore Address Map
November 2012	1.2	Minor updates.
May 2012	1.1	• Add description of the ACP ID mapper • Consolidate redundant information

Date	Version	Changes
January 2012	1.0	Initial release.

CoreSight Debug and Trace 10

2014.07.31

cv_54007 Subscribe Send Feedback

The hard processor system (HPS) debug infrastructure provides visibility and control of the HPS modules, the ARM® Cortex®-A9 microprocessor unit (MPU) subsystem, and user logic implemented in the FPGA fabric. The debug system design incorporates ARM® CoreSight™ components.

Details of the ARM CoreSight debug components can be viewed by clicking on the related information links below.

Related Information

- Debug Access Port (DAP) on page 10-4
- System Trace Macrocell (STM) on page 10-4
- Trace Funnel on page 10-5
- Embedded Trace FIFO (ETF) on page 10-5
- AMBA Trace Bus Replicator (Replicator) on page 10-5
- Embedded Trace Router (ETR) on page 10-5
- Trace Port Interface Unit (TPIU) on page 10-5
- Embedded Cross Trigger (ECT) System on page 10-6
- Program Trace Macrocell (PTM) on page 10-11

Features of CoreSight Debug and Trace

The CoreSight debug and trace system offers the following features:

- Real-time program flow instruction trace through a separate PTM for each processor
- Host debugger JTAG interface
- Connections for cross-trigger and STM-to-FPGA interfaces, which enable soft IP generation of triggers and system trace messages
- Instruction trace interface through TPIU for trace analysis tools
- Custom message injection through STM into trace stream for delivery to host debugger

- STM and PTM trace sources multiplexed into a single stream through the Trace Funnel
- Capability to route trace data to any slave accessible to the ETR AXI master connected to the level 3 (L3) interconnect
- Capability for the following SoC modules to trigger each other through the embedded cross-trigger system:
  - FPGA fabric
  - A9-0 processor
  - A9-1 processor
  - PTM-0
  - PTM-1
  - STM
  - ETF
  - ETR
  - TPIU
  - csCTI
  - CTI-0
  - CTI-1
  - FPGA-CTI
  - csCTM
  - CTM

ARM CoreSight Documentation

The following ARM CoreSight specifications and documentation provide a more thorough description of the ARM CoreSight components in the HPS debug system:

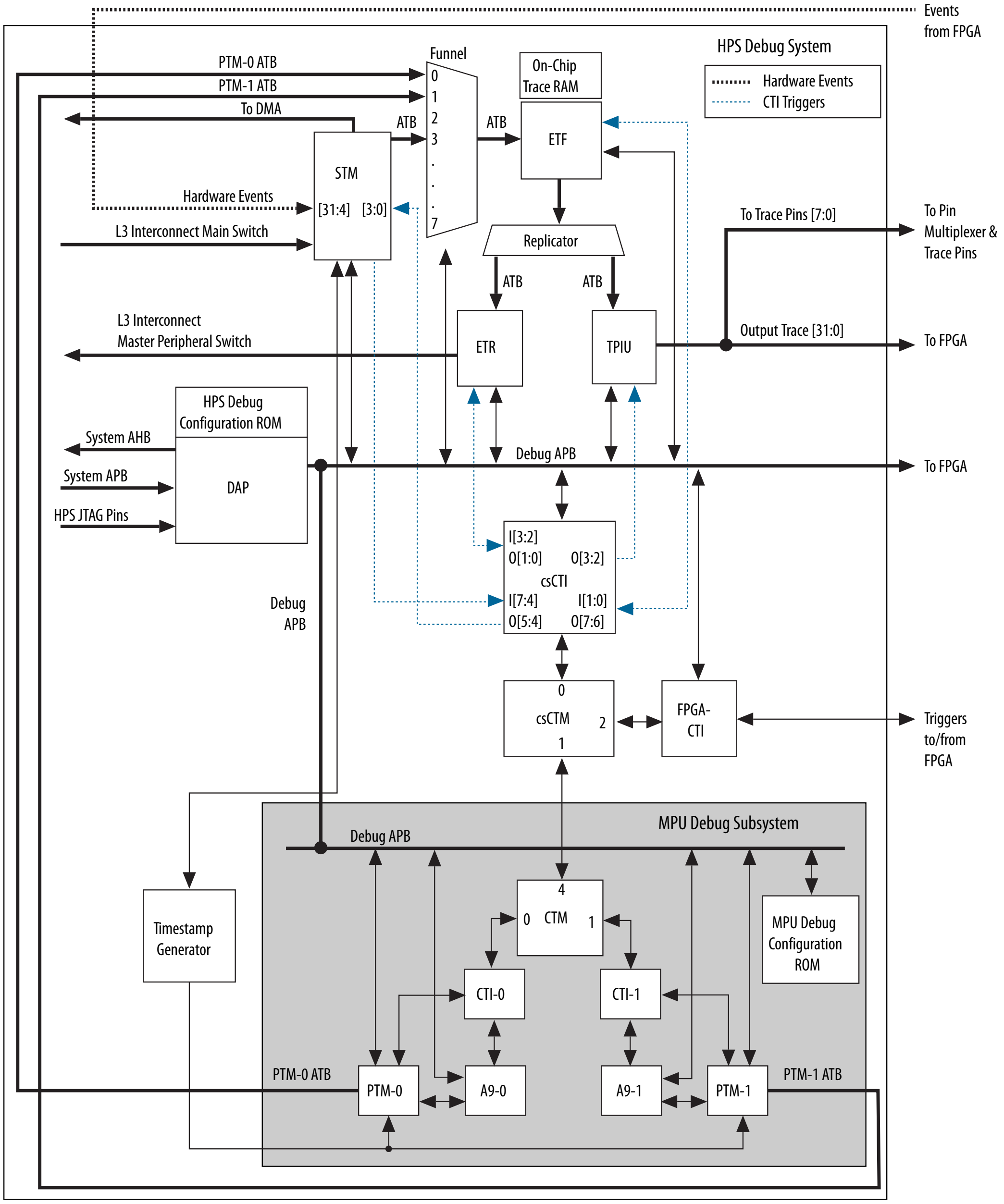
- *CoreSight Technology, System Design Guide, ARM DGI 0012D*
- *CoreSight Architecture Specification, ARM IHI 0029B*
- *ARM Debug Interface v5, Architecture Specification, ARM IHI 0031A*
- *Embedded Cross Trigger Technical Reference Manual, ARM DDI 0291A*
- *CoreSight Components Technical Reference Manual, ARM DDI 0314H*
- *CoreSight Program Flow Trace, Architecture Specification, ARM IHI 0035A*
- *CoreSight PTM-A9 Technical Reference Manual, ARM DDI 0401B*
- *CoreSight System Trace Macrocell Technical Reference Manual, ARM DDI 0444A*
- *System Trace Macrocell, Programmers' Model Architecture Specification, ARM IHI 0054*
- *CoreSight Trace Memory Controller Technical Reference Manual, ARM DDI 0461B*

Related Information

http://infocenter.arm.com/
For more information, refer to the *CoreSight Components* Technical Reference Manual and the *CoreSight Technology System* Design Guide.

CoreSight Debug and Trace Block Diagram and System Integration

Figure 10-1: HPS CoreSight Debug and Trace System Block Diagram



Functional Description of CoreSight Debug and Trace

CoreSight systems provide all the infrastructure you require to debug, monitor, and optimize the performance of a complete HPS design. CoreSight technology addresses the requirement for a multicore debug and trace solution with high bandwidth for whole systems beyond the processor core.

CoreSight technology provides the following features:

- Cross-trigger support between SoC subsystems
- High data compression
- Multisource trace in a single stream
- Standard programming models for standard tool support

Related Information

http://infocenter.arm.com/
You can download the documents from the ARM info center website.

Debug Access Port (DAP)

The DAP provides the necessary ports for a host debugger to connect to and communicate with the HPS through a JTAG interface connected to dedicated HPS pins that is independent of the JTAG for the FPGA. The JTAG interface provided with the DAP allows a host debugger to access various modules inside the HPS. Additionally, a debug monitor executing on either processor can access different HPS components by interfacing with the system Advanced Microcontroller Bus Architecture (AMBA®) Advanced Peripheral Bus (APB™) slave port of the DAP. The system APB slave port occupies 2 MB of address space in the HPS. Both the JTAG port and system APB port have access to the debug APB master port of the DAP. As shown in **Figure 10-1**, all CoreSight components are connected to the debug APB.

A host debugger can access any HPS memory-mapped resource in the system via the DAP system master port. Requests made over the DAP system master port are impacted by reads and writes to peripheral registers.

Note: The HPS JTAG interface does not support boundary scan tests (BST). To perform boundary scan testing on HPS I/Os, use the FPGA JTAG pins.

Related Information

http://infocenter.arm.com/
For more information, refer to the *CoreSight Components* Technical Reference Manual on the ARM info center website.

System Trace Macrocell (STM)

The STM allows messages to be injected into the trace stream for delivery to the host debugger receiving the trace data. These messages can be sent via stimulus ports or the hardware event interface. The STM allows the messages to be time stamped.

The STM provides an AMBA Advanced eXtensible Interface (AXI™) slave interface used to create trace events. The interface can be accessed by the MPU subsystem, direct memory access (DMA) controller, and masters implemented as soft logic in the FPGA fabric via the FPGA-to-HPS bridge. The AXI slave interface supports three address segments, where each address segment is 16 MB and each segment supports up to 65536 channels. Each channel occupies 256 bytes of address space.

The STM also provides 32 hardware event pins. The higher-order 28 pins (31:4) are connected to the FPGA fabric, allowing logic inside FPGA to insert messages into the trace stream. When the STM detects a rising edge on an event pin, a message identifying the event is inserted into the stream. The lower four event pins (3:0) are connected to csCTI.

Related Information

- **HPS-FPGA Bridges** on page 8-1

- **http://infocenter.arm.com/**
  For more information, refer to the *CoreSight System Trace Macrocell* Technical Reference Manual on the ARM info center website.

Trace Funnel

The Trace Funnel multiplexes three trace sources into a single trace stream. Port 0 of the Trace Funnel is connected to the PTM for CPU 0. Port 1 of the Trace Funnel is connected to the PTM for CPU 1. Port 3 of the Trace Funnel is connected to the STM. Port 2 and Port 4 through Port 7 are not used.

Related Information

http://infocenter.arm.com/
For more information, refer to the *CoreSight Components* Technical Reference Manual on the ARM info center website.

Embedded Trace FIFO (ETF)

The output of the Trace Funnel is sent to the ETF. The ETF is used as an elastic buffer between trace generators (STM, PTM) and trace destinations. The ETF stores up to 32 KB of trace data in the on-chip trace RAM.

AMBA Trace Bus Replicator (Replicator)

The Replicator broadcasts trace data from the ETF to the embedded trace router (ETR) and trace port interface unit (TPIU).

Related Information

http://infocenter.arm.com/
For more information, refer to the *CoreSight Components* Technical Reference Manual on the ARM info center website.

Embedded Trace Router (ETR)

The ETR can route trace data to the HPS on-chip RAM, the HPS SDRAM, and any memory in the FPGA fabric connected to the HPS-to-FPGA bridge. The ETR receives trace data from the Replicator. By default, the buffer to receive the trace data resides in SDRAM at offset 0x00100000 and is 32 KB. You can override this default configuration by programming registers in the ETR.

Related Information

HPS-FPGA Bridges on page 8-1

Trace Port Interface Unit (TPIU)

The TPIU is a bridge between on-chip trace sources and an off-chip trace port. The TPIU receives trace data from the Replicator and drives the trace data to a trace port analyzer.

The trace output from the TPIU is software programmable and can be set to either 8 or 32 bits wide. The trace output is routed to an 8-bit HPS I/O interface and a 32-bit interface to the FPGA fabric. The trace data sent to the FPGA fabric can be transported off-chip using available serializer/deserializer (SERDES) resources in the FPGA.

Related Information

http://infocenter.arm.com/

For more information, refer to the *CoreSight Components* Technical Reference Manual on the ARM info center website.

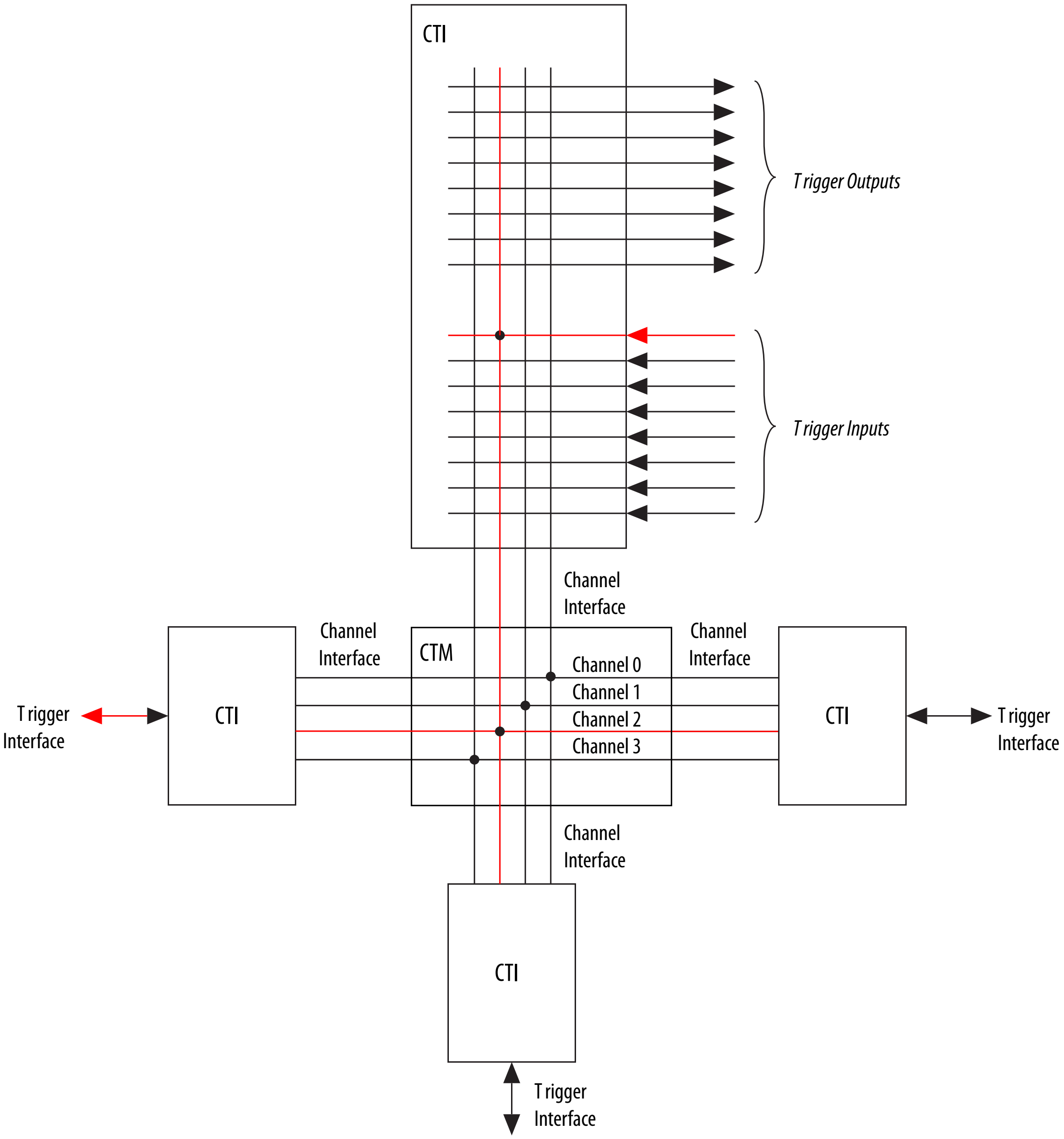
Embedded Cross Trigger (ECT) System

The ECT system provides a mechanism for HPS modules to trigger each other. The ECT consists of the following modules:

- Cross Trigger Interface (CTI)
- Cross Trigger Matrix (CTM)

Figure 10-2: Generic ECT System

The following figure shows how CTIs and CTMs are used in a generic ECT setup. The red line depicts an trigger input to one CTI generating a trigger output in another CTI. Though the signal travels throughout channel 2, it only enters and exits through trigger inputs and outputs you configure.



Related Information

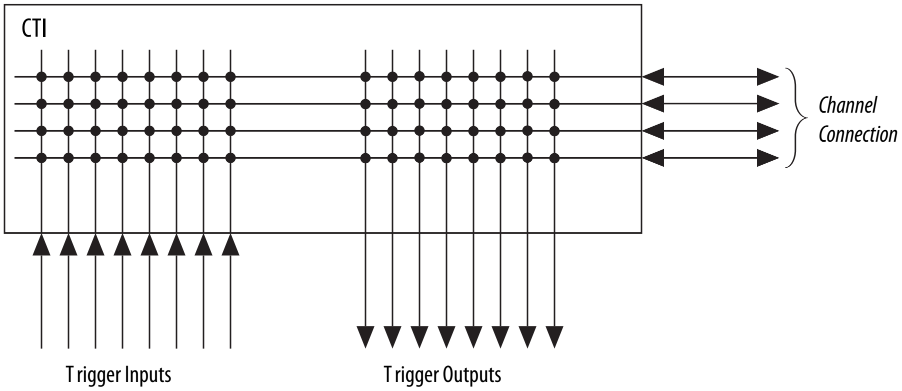
- Cross Trigger Interface (CTI) on page 10-8
- Cross Trigger Matrix (CTM) on page 10-8

Cross Trigger Interface (CTI)

CTIs allow trigger sources and sinks to interface with the ECT. Each CTI supports up to eight trigger inputs and eight trigger outputs, and is connected to a CTM. **Figure 10-2** shows the relationship of trigger inputs, trigger outputs, and CTM channels of a CTI.

Figure 10-3: CTI Connections

The following figure shows trigger input and trigger output connections in detail.



The HPS debug system contains the following four CTIs:

- csCTI—performs cross triggering between the STM, ETF, ETR, and TPIU.
- FPGA-CTI—exposes the cross-triggering system to the FPGA fabric.
- CTI-0 and CTI-1—reside in the MPU debug subsystem. Each CTI is associated with a processor and the processor's associated PTM.

Cross Trigger Matrix (CTM)

A CTM is a transport mechanism for triggers traveling from one CTI to one or more CTIs or CTMs. The HPS contains two CTMs. One CTM connects csCTI and FPGA-CTI; the other connects CTI-0 and CTI-1. The two CTMs are connected together, allowing triggers to be transmitted between the MPU debug subsystem, the debug system, and the FPGA fabric.

Each CTM has four ports and each port has four channels. Each CTM port can be connected to a CTI or another CTM.

Figure 10-4: CTM Channel Structure

The following figure shows the structure of a CTM channel. Paths inside the CTM are purely combinatorial.

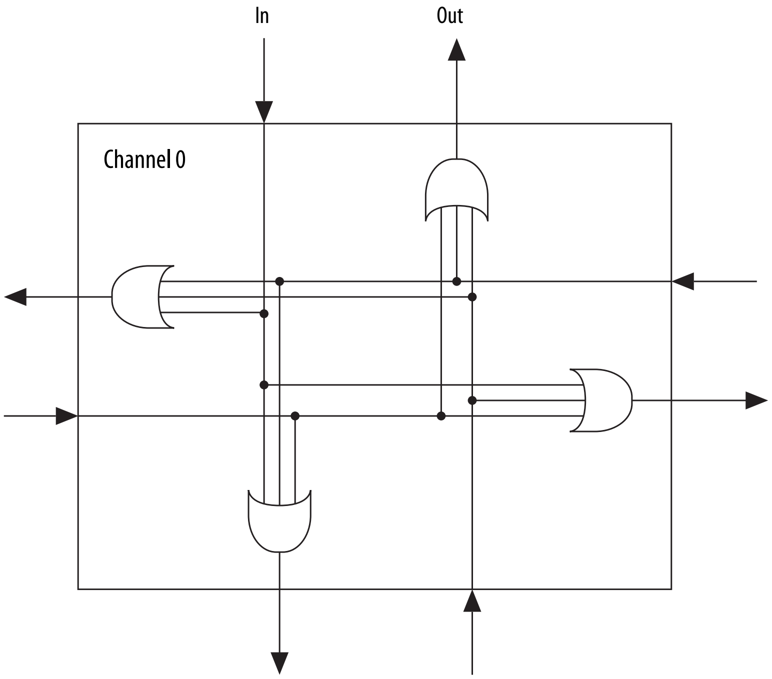
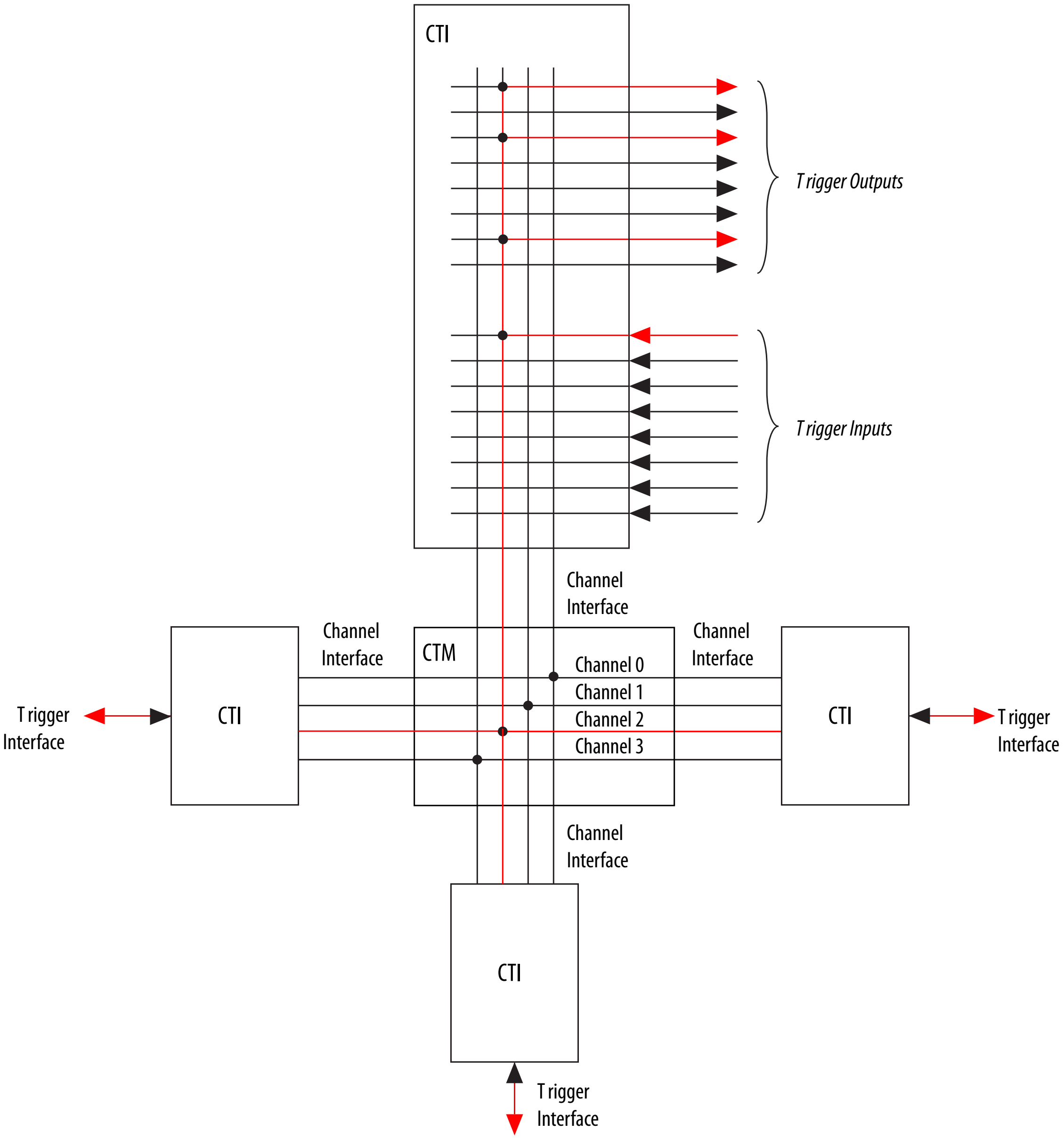


Figure 10-5: CTI Trigger Connections

Each CTI trigger input can be connected through a CTM to one or more trigger outputs under control by a debugger. The following figure shows a pictorial representation of CTI trigger connections. The red lines depict the impact one trigger input can have on the entire system.



Related Information

http://infocenter.arm.com/

For more information, refer to the *CoreSight Components* Technical Reference Manual on the ARM info center website.

Program Trace Macrocell (PTM)

The PTM performs real-time program flow instruction tracing and provides a variety of filters and triggers that can be used to trace specific portions of code.

The HPS contains two PTMs. Each PTM is paired with a processor and CTI. Trace data generated from the PTM can be transmitted off-chip using HPS pins, or to the FPGA fabric, where it can be pre-processed and transmitted off-chip using high-speed FPGA pins.

Related Information

http://infocenter.arm.com/
For more information, refer to the *CoreSight PTM-A9* Technical Reference Manual.

HPS Debug APB Interface

The HPS can extend the CoreSight debug control bus into the FPGA fabric. The debug interface is an APB-compatible interface with built-in clock crossing.

Related Information

HPS Component Interfaces on page 28-1

CoreSight Debug and Trace Programming Model

This section describes programming model details specific to Altera's implementation of the ARM CoreSight technology.

The debug components can be configured to cause triggers when certain events occur. For example, soft logic in the FPGA fabric can signal an event which triggers an STM message injection into the trace stream. CoreSight components are configured through memory-mapped registers, located at offsets relative to the CoreSight component base address. CoreSight component base addresses are accessible through a ROM table.

Related Information

http://infocenter.arm.com/
Programming interface details of each CoreSight component.

ROM Table

Table 10-1: DAP ROM Table

The following table contains entries found in the ROM table portion of the DAP.

ROM Entry	Offset[30:12]	Description
0x0	0x00001	ETF
0x1	0x00002	CTI
0x2	0x00003	TPIU
0x3	0x00004	Trace Funnel

ROM Entry	Offset[30:12]	Description
0x4	0x00005	STM
0x5	0x00006	ETR
0x6	0x00007	FPGA-CTI
0x7	0x00100	A9ROM
0x8	0x00080	FPGAROM
0x9	0x00000	End of ROM

A host debugger can access this table at 0x8000_0000 through the DAP. HPS masters can access this ROM at 0xFF00_0000. Registers for a particular CoreSight component are accessed by adding the register offset to the CoreSight component base address, and adding that total to the base address of the ROM table.

The base address of the ROM table is different when accessed from the debugger (at 0x8000_0000) than when accessed from any HPS master (at 0xFF00_0000). For example, the CTI output enable register, `CTIOUTEN[2]` at offset 0xA8, can be accessed by the host debugger at 0x8000_20A8. To derive that value, add the host debugger access address to the ROM table of 0x8000_0000, to the CTI component base address of 0x0000_2000, to the `CTIOUTEN[2]` register offset of 0xA8.

STM Channels

The STM AXI slave is connected to the MPU, DMA, and FPGA-to-HPS bridge masters. Each master has up to 65536 channels where each channel occupies 256 bytes of address space, for a total of 16 MB per master. The HPS address map allocates 48 MB of consecutive address space to the STM AXI slave port, divided in three 16 MB segments.

Table 10-2: STM AXI Slave Port Address Allocation

Segment	Start Address	End Address
0	0xFC00_0000	0xFCFF_FFFF
1	0xFD00_0000	0xFDFF_FFFF
2	0xFE00_0000	0xFEFF_FFFF

Each of the three masters can access any one of the three address segments. Your software design determines which master uses which segment, based on the value of bits 24 and 25 in the write address, `AWADDRS[25:24]`. Software must restrict each master to use only one of the three segments.

Table 10-3: STM AXI Address Fields

AXI Signal Fields	Description
`AWADDRS[7:0]`	These bits index the 256 bytes of the stimulus port.

AXI Signal Fields	Description
`AWADDRS[23:8]`	These bits identify the 65536 stimulus ports associated with a master.
`AWADDRS[25:24]`	These bits identify the three masters. Only 0, 1, and 2 are valid values.
`AWADDRS[31:26]`	Always 0x3F. Bits 24 to 31 combine to access 0xFC00_0000 through 0xFEFF_FFFF.

Each STM message contains a master ID that tells the host debugger which master is associated with the message. The STM master ID is determined by combining a portion of the `AWADDRS` signal and the `AWPROT` protection bit.

Table 10-4: STM Master ID Calculation

Master ID Bits	AXI Signal Bits	Notes
Master ID[5:0]	`AWADDRS[29:24]`	The lowest two bits are sufficient to determine which master, but CoreSight uses a seven-bit master ID.
Master ID[6]	`AWPROT[1]`	0 indicates secure; 1 indicates nonsecure.

In addition to access through STM channels, the higher-order 28 (31:4) of the 32 event signals are attached to the FPGA through the FPGA-CTI. These event signals allow the FPGA fabric to send additional messages using the STM.

Related Information

- **HPS-FPGA Bridges** on page 8-1
- **http://infocenter.arm.com/**
  For more information, refer to the System Trace Macrocell in the *Programmers' Model Architecture* Specification.

FPGA Interface

The following components connect to the FPGA fabric. This section lists the signals from debug system to the FPGA.

DAP

The DAP uses the system APB port to connect to the FPGA.

Table 10-5: DAP

The following table shows the signal description between DAP and FPGA.

Segment	Description
h2f_dbg_apb_PADDR[18]	Address bus to system APB port

Segment	Description
h2f_dbg_apb_PADDR31	Address bus to system APB port
h2f_dbg_apb_PENABLE	Enable signal from system APB port
h2f_dbg_apb_PRDATA[32]	32-bit system APB port read data bus
h2f_dbg_apb_PREADY	Ready signal to system APB port
h2f_dbg_apb_PSEL	Select signal from system APB port
h2f_dbg_apb_PSLVERR	Error signal to system APB port
h2f_dbg_apb_PSLVERR	32-bit system APB port write data bus
h2f_dbg_apb_PWRITE	Select whether read or write to system APB port • 0 - System APB port read from DAP • 1 - System APB Port write to DAP

STM

The STM has 28 event pins, f2h_stm_hw_events[28], for FPGA to trigger events to STM.

FPGA-CTI

The FPGA-CTI allows the FPGA to send and receive triggers from the debug system.

Table 10-6: FPGA-CTI

The following table lists the signal descriptions between the FPGA-CTI and FPGA.

Signal	Description
h2f_cti_trig_in[8]	Trigger input from FPGA
h2f_cti_trig_in_ack[8]	ACK signal to FPGA
h2f_cti_trig_out[8]	Trigger output to FGPA
h2f_cti_trig_out_ack[8]	ACK signal from FPGA
h2f_cti_clk	Clock input from FPGA
h2f_cti_fpga_clk_en	Clock enable driven by FPGA
h2f_cti_asicctl[8]	Signal from FPGA

Related Information

http://infocenter.arm.com/

TPIU

Table 10-7: TPIU Signals

The following table lists the signal descriptions between the TPIU and FPGA.

Signal	Description
h2f_tpiu_clk_ctl	Selects whether trace data is captured using the internal TPIU clock or an external clock provided as an input to the TPIU from the FPGA. 0 - use h2f_tpiu_clock_in 1 - use internal clock Note: When the FPGA is powered down or not configured the TPIU uses the internal clock.
h2f_tpiu_data[32]	32 bit trace data bus to the FPGA. Trace data changes on both edges of h2f_tpiu_clock. Note: When the FPGA is powered down or not configured, the TPIU sends the lower 8-bits trace data to I/Os.
h2f_tpiu_clock_in	Clock from the FPGA used to capture trace data.
h2f_tpiu_clock	Clock output from TPIU

CTI Trigger Connections to Outside the Debug System

The following CTIs in the HPS debug system connect to outside the debug system:

- csCTI
- FPGA-CTI

csCTI

This section lists the trigger input, output, and output acknowledge pin connections implemented for csCTI in the debug system. The trigger input acknowledge signals are not connected to pins.

Table 10-8: csCTI Trigger Input Signals

The following table lists the trigger input pin connections implemented for csCTI.

Number	Signal	Source
7	ASYNCOUT	STM
6	TRIGOUTHETE	STM
5	TRIGOUTSW	STM
4	TRIGOUTSPTE	STM

Number	Signal	Source
3	ACQCOMP	ETR
2	FULL	ETR
1	ACQCOMP	ETF
0	FULL	ETF

Table 10-9: csCTI Trigger Output Signals

The following table lists the trigger output pin connections implemented for csCTI.

Number	Signal	Destination
7	TRIGIN	ETF
6	FLUSHIN	ETF
5	HWEVENTS[3:2]	STM
4	HWEVENTS[1:0]	STM
3	TRIGIN	TPIU
2	FLUSHIN	TPIU
1	TRIGIN	ETR
0	FLUSHIN	ETR

Table 10-10: csCTI Trigger Output Acknowledge Signals

The following table lists the trigger output pin acknowledge connections implemented for csCTI.

Number	Signal	Source
7	0	—
6	0	—
5	0	—
4	0	—
3	TRIGINACK	TPIU
2	FLUSHINACK	TPIU

Number	Signal	Source
1	0	—
0	0	—

FPGA-CTI

FPGA-CTI connects the debug system to the FPGA fabric. FPGA-CTI has all of its triggers available to the FPGA fabric.

Configuring Embedded Cross-Trigger Connections

CTI interfaces are programmable through a memory-mapped register interface.

The specific registers are described in the CoreSight Components Technical Reference Manual, which you can download from the ARM website (infocenter.arm.com).

To access registers in any CoreSight component through the debugger, the register offsets must be added to the CoreSight component's base address. That combined value must then be added to the address at which the ROM table is visible to the debugger (0x80000000).

Each CTI has two interfaces, the trigger interface and the channel interface. The trigger interface is the interface between the CTI and other components. It has eight trigger signals, which are hardwired to other components. The channel interface is the interface between a CTI and its CTM, with four bidirectional channels. The mapping of trigger interface to channel interface (and vice versa) in a CTI is dynamically configured. You can enable or disable each CTI trigger output and CTI trigger input connection individually.

For example, you can configure trigger input 0 in the FPGA-CTI to route to channel 3, and configure trigger output 3 in the FPGA-CTI and trigger output 7 in CTI-0 in the MPU debug subsystem to route from channel 3. This configuration causes a trigger at trigger input 0 in FPGA-CTI to propagate to trigger output 3 in the FPGA-CTI and trigger output 7 in CTI-0. Propagation can be single-to-single, single-to-multiple, multiple-to-single, and multiple-to-multiple.

A particular soft logic signal in the FPGA connected to a trigger input in the FPGA-CTI can be configured to trigger a flush of trace data to the TPIU. For example, you can configure channel 0 to trigger output 2 in csCTI. Then configure trigger input T3 to channel 0 in FPGA-CTI. Trace data is flushed to the TPIU when a trigger is received at trigger output 2 in csCTI.

Another soft logic signal in the FPGA connected to trigger input T2 in FPGA-CTI can be configured to trigger an STM message. csCTI output triggers 4 and 5 are wired to the STM CoreSight component in the HPS. For example, configure channel 1 to trigger output 4 in csCTI. Then configure trigger input T2 to channel 1 in FPGA-CTI.

Another soft logic signal in the FPGA fabric connected to trigger input T1 in FPGA-CTI can be configured to trigger a breakpoint on CPU 1. Trigger output 1 in CTI-1 is wired to the debug request (EDBGRQ) signal of CPU-1. For example, configure channel 2 to trigger output 1 in CTI-1. Then configure trigger input T1 to channel 2 in FPGA-CTI.

Related Information

http://infocenter.arm.com/

Debug Clocks

The CoreSight system uses several different clocks. Port Name is the name of the clock signal inputs described for individual CoreSight debug components in the ARM documentation. Signal Name is the name of the clock signal used with other HPS components.

Table 10-11: CoreSight Clocks

Port Name	Clock Source	Signal Name	Description
ATCLK	Clock manager	dbg_at_clk	Trace bus clock.
CTICLK(for csCTI)	Clock manager	dbg_at_clk	Cross trigger interface clock for csCTI. It can be synchronous or asynchronous to CTMCLK.
CTICLK (for FPGA-CTI)	FPGA fabric	fpga_cti_clk	Cross trigger interface clock for FPGA-CTI.
CTICLK (for CTI-0 and CTI-1)	Clock manager	mpu_clk	Cross trigger interface clock for CTI-0 and CTI-1. It can be synchronous or asynchronous to CTMCLK.
CTMCLK(for csCTM)	Clock manager	dbg_clk	Cross trigger matrix clock for csCTM. It can be synchronous or asynchronous to CTICLK.
CTMCLK(for CTM)	Clock manager	mpu_clk	Cross trigger matrix clock for CTM. It can be synchronous or asynchronous to CTICLK.
DAPCLK	Clock manager	dbg_clk	DAP internal clock. It must be equivalent to PCLKDBG.
PCLKDBG	Clock manager	dbg_clk	Debug APB (DAPB) clock.
HCLK	Clock manager	dbg_clk	Used by the AHB-Lite master inside the DAP. It is asynchronous to DAPCLK. In the HPS, the AHB-Lite port uses same clock as DAPCLK.
PCLKSYS	Clock manager	l4_mp_clk	Used by the APB slave port inside the DAP. It is asynchronous to DAPCLK.

Port Name	Clock Source	Signal Name	Description
SWCLKTCK	JTAG interface	dap_tck	The SWJ-DP clock driven by the external debugger through either the JTAG interface or the FPGA fabric. It is asynchronous to DAPCLK. When through the JTAG interface, this clock is the same as TCK of the JTAG interface.
	FPGA fabric	tpiu_traceclkin	
TRACECLKIN	Clock manager	dbg_trace_clk	TPIU trace clock input. It is asynchronous to ATCLK. In the HPS, this clock can come from the clock manager or the FPGA fabric.

Related Information

http://infocenter.arm.com/
For more information about the CoreSight port names, refer to the *CoreSight Technology System* Design Guide, which you can download from the ARM info center website.

Debug Resets

The CoreSight system uses several resets. Port Name is the name of the clock signal inputs described for individual CoreSight debug components in the ARM documentation. Signal Name is the name of the clock signal used with other HPS components.

Table 10-12: CoreSight Resets

Port Name	Clock Source	Signal Name	Description
ATRESETn	Reset manager	dbg_rst_n	Trace bus reset. It resets all registers in ATCLK domain.
nCTIRESET	Reset manager	dbg_rst_n	CTI reset signal. It resets all registers in CTICLK domain. In the HPS, there are four instances of CTI. All four use the same reset signal.
DAPRESETn	Reset manager	dbg_rst_n	DAP internal reset. It is connected to PRESETDBGn.
PRESETDBGn	Reset manager	dbg_rst_n	Debug APB reset. Resets all registers clocked by PCLKDBG.
HRESETn	Reset manager	sys_dbg_rst_n	SoC-provided reset signal that resets all of the AMBA on-chip interconnect. Use this signal to reset the DAP AHB-Lite master port.
PRESETSYSn	Reset manager	sys_dbg_rst_n	Resets system APB slave port of DAP.

Port Name	Clock Source	Signal Name	Description
nCTMRESET	Reset manager	dbg_rst_n	CTM reset signal. It resets all signals clocked by CTMCLK.
nPOTRST	Reset manager	tap_cold_rst_n	True power on reset signal to the DAP SWJ-DP. It must only reset at power-on.
nTRST	JTAG interface	nTRST pin	Resets the DAP TAP controller inside the SWJ-DP. This signal is driven by the host using the JTAG connector.
TRESETn	Reset manager	dbg_rst_n	Reset signal for TPIU. Resets all registers in the TRACECLKIN domain.

The ETR stall enable field (etrstallen) of the ctrl register in the reset manager controls whether the ETR is requested to stall its AXI master interface to the L3 interconnect before a warm or debug reset.

The level 4 (L4) watchdog timers can be paused during debugging to prevent reset while the processor is stopped at a breakpoint.

Related Information

- **Reset Manager** on page 3-1
- **Watchdog Timer** on page 24-1
- **http://infocenter.arm.com/**
  For more information about the CoreSight port names, refer to the *CoreSight Technology System* Design Guide.

CoreSight Debug and Trace Address Map and Register Definitions

The address map and register definitions for CoreSight Debug and Trace consist of the following regions:

- STM Module Address Space
- DAP Module Address Space
- MPU SCU Module Address Space
- MPU L2 Cache Controller Module Address Space

System Trace Macrocell (STM) Module Address Map on page 10-21
This address space holds the registers used for Coresight System Trace Macrocell. For detailed information about the STM module and register descriptions, **click here** to access the ARM documentation for the CoreSight STM-101.

Debug Access Port (DAP) Module Address Map on page 10-21
This address space is allocated to the Debug Access Port (DAP). For detailed information about the use of this address space, **click here** to access the ARM documentation for the DAP.

MPU Address Map on page 10-23
This address space is allocated to the MPU. For detailed information about the use of this address space, **click here** to access the ARM documentation for the Cortex-A9 MPCore.

MPU L2 Cache Controller (L2C-310) Module Address Map on page 10-24
This address space is allocated to the MPU L2 cache controller. For detailed information about the use of this address space, **click here** to access the ARM documentation for the L2C-310.

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

System Trace Macrocell (STM) Module Address Map

This address space holds the registers used for Coresight™ System Trace Macrocell. For detailed information about the STM module and register descriptions, **click here** to access the ARM documentation for the CoreSight STM-101.

Table 10-13: STM Module Address Range

Module Instance	Start Address	End Address
STM	0xFC000000	0xFEFFFFFF

Debug Access Port (DAP) Module Address Map

This address space is allocated to the Debug Access Port (DAP). For detailed information about the use of this address space, **click here** to access the ARM documentation for the DAP.

Table 10-14: DAP Module Address Range

Module Instance	Start Address	End Address
DAP	0xFF000000	0xFF1FFFFF

Table 10-15: DAP Module Register Space

Register Group	Description	Start Address	End Address
`DAP ROM`	This address space is allocated for the Debug Access Port (DAP) ROM.	`0xFF000000`	`0xFF000FFF`
`ETF`	This address space is allocated for the Embedded Trace FIFO.	`0xFF001000`	`0xFF001FFF`
`CTI`	This address space is allocated for the Cross-Trigger Interface (CTI).	`0xFF002000`	`0xFF002FFF`
`TPIU`	This address space is allocated for the Trace Port Interface Unit.	`0xFF003000`	`0xFF003FFF`
`Trace Funnel`	This is the address space is allocated for the Trace Funnel.	`0xFF004000`	`0xFF004FFF`
`STM`	This address space is allocated for the System Trace Macrocell (STM).	`0xFF005000`	`0xFF005FFF`
`ETR`	This address space is allocated for the Embedded Trace Router.	`0xFF006000`	`0xFF006FFF`
`CTI FPGA`	This address space is allocated for the Cross-Trigger Interface of the FPGA.	`0xFF007000`	`0xFF007FFF`
`FPGA ROM`	This address space is allocated for the FPGA ROM.	`0xFF080000`	`0xFF080FFF`
`FPGA CoreSight Components`	This address space is allocated for the FPGA CoreSight Components.	`0xFF081000`	`0xFF0FF000`

Register Group	Description	Start Address	End Address
Cortex-A9 ROM	This address space is allocated for the Cortex-A9 ROM.	0xFF100000	0xFF10FFFF
CPU0 Debug	This address space is allocated for CPU0 Debug.	0xFF110000	0xFF110FFF
CPU0 PMU	This address space is allocated for the CPU0 PMU.	0xFF111000	0xFF111FFF
CPU1 Debug	This address space is allocated for CPU1 Debug.	0xFF112000	0xFF112FFF
CPU1 PMU	This address space is allocated for the CPU1 PMU.	0xFF113000	0xFF117FFF
CTI0	This address space is allocated for Cross-Trigger Interface 0 (CTI0).	0xFF118000	0xFF118FFF
CTI1	This address space is allocated for Cross-Trigger Interface 1 (CTI1)	0xFF119000	0xFF11BFFF
PTM0	This addres space is allocated for Program Trace Macrocell 0 (PTM0)	0xFF11C000	0xFF11CFFF
PTM1	This address space is allocated for Program Trace Macrocell 1 (PTM1).	0xFF11D000	0xFF11DFFF

MPU Address Map

This address space is allocated to the MPU. For detailed information about the use of this address space, click here to access the ARM documentation for the Cortex-A9 MPCore.

Table 10-16: MPU Module Address Range

Module Instance	Start Address	End Address
MPU	0xFFFEC000	0xFFFEEFFF

Table 10-17: MPU Module Register Space

Module Instance	Description	Start Address	End Address
SCU	This address space is allocated for the Snoop Control Unit registers.	0xFFFEC000	0xFFFEC0FF
GIC	This address space is allocated for the General Interrupt Controller (GIC) registers.	0xFFFEC100	0xFFFEC1FF
Global Timer	This address space is allocated for the Global Timer registers.	0xFFFC200	0xFFFEC2FF
Reserved	This address space is reserved.	0xFFFEC300	0xFFFEC5FF
Private Timers and Watchdog Timers	This is the address space is allocated for private timers and watchdog timers.	0xFFFEC600	0xFFFEC6FF
Reserved	This address space is reserved. Caution: Any access to this region causes a SLVERR abort exception.	0xFFFEC700	0xFFFEC7FF
Interrupt Distributor	This address space is allocated for the interrupt distributor.	0xFFFED000	0xFFFEDFFF
Reserved	This address space is reserved.	0xFFFEE000	0xFFFEEFFF

MPU L2 Cache Controller (L2C-310) Module Address Map

This address space is allocated to the MPU L2 cache controller. For detailed information about the use of this address space, click here to access the ARM documentation for the L2C-310.

Table 10-18: MPU L2 Cache Controller Address Range

Module Instance	Start Address	End Address
MPUL2	0xFFFEF000	0xFFFEFFFF

Table 10-19: MPU L2 Cache Controller Register Range

Register Group	Description	Start Address	End Address
Cache ID and Cache Type	This address space is allocated for the cache ID and cache type registers.	0xFFFEF000	0xFFFEF0FF
Control	This is the address space for the cache control registers.	0xFFFEF100	0xFFFEF1FF
Interrupt/Counter Control	This address space is allocated for the Interrupt/Counter control registers.	0xFFFEF200	0xFFFEF2FF
Reserved	This address space is reserved.	0xFFFEF300	0xFFFEF6FF
Cache Maintenance Operations	This is the address space is allocated for the cache maintenance operation registers.	0xFFFEF700	0xFFFEF7FF
Reserved	This address space is reserved.	0xFFFEF800	0xFFFEF8FF
Cache Lockdown	This addres space is allocated for cache lockdown registers.	0xFFFEF900	0xFFFEF9FF
Reserved	This address space is reserved.	0xFFFEFA00	0xFFFEFBFF
Address Filtering	This address space is allocated for address filtering registers.	0xFFFEFC00	0xFFFEFCFF
Reserved	This address space is reserved.	0xFFFEFD00	0xFFFEFEFF
Debug, Prefetch, Power	This address space is allocated for debug, prefetch and power registers.	0xFFFEFF00	0xFFFEFFFF

Document Revision History

Table 10-20: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added the address map and register definitions.
February 2014	2014.02.28	Maintenance release.
December 2013	2013.12.30	Maintenance release.
November 2012	1.2	Minor updates.
January 2012	1.1	Added functional description, programming model, and address map and register definition sections.
January 2012	1.0	Initial release.

Functional Description—HPS Memory Controller 11

2014.06.30

cv_54008 Subscribe Send Feedback

The hard processor system (HPS) SDRAM controller subsystem provides efficient access to external SDRAM for the ARM® Cortex™-A9 microprocessor unit (MPU) subsystem, the level 3 (L3) interconnect, and the FPGA fabric. The SDRAM controller provides an interface between the FPGA fabric and HPS. The interface accepts Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) and Avalon® Memory-Mapped (Avalon-MM) transactions, converts those commands to the correct commands for the SDRAM, and manages the details of the SDRAM access.

Features of the SDRAM Controller Subsystem

The SDRAM controller subsystem offers programming flexibility, port and bus configurability, error correction, and power management for external memories up to 4 GB.

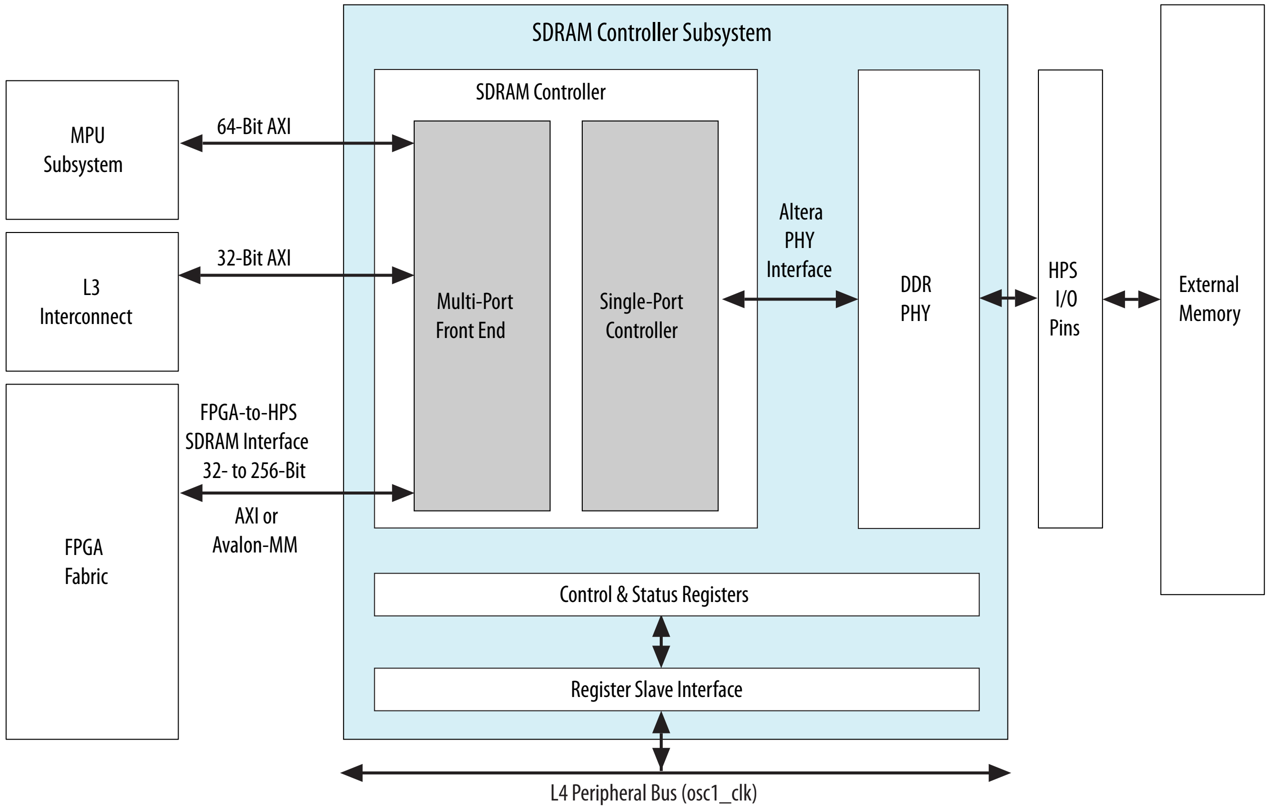
- Support for double data rate 2 (DDR2), DDR3, and low-power DDR2 (LPDDR2) SDRAM
- Flexible row and column addressing with the ability to support up to 4 GB of memory in various interface configurations
- Optional 8-bit integrated error correction code (ECC) for 16- and 32-bit data widths[16]
- User-configurable memory width of 8, 16, 16+ECC, 32, 32+ECC
- User-configurable timing parameters
- Two chip selects
- Command reordering (look-ahead bank management)
- Data reordering (out of order transactions)
- User-controllable bank policy on a per port basis for either closed page or conditional open page accesses
- User-configurable priority support with both absolute and weighted round-robin scheduling
- Flexible FPGA fabric interface with up to 6 ports that can be combined for a data width up to 256 bits wide using Avalon-MM and AXI interfaces.
- Power management supporting self refresh, partial array self refresh (PASR), power down, and LPDDR2 deep power down

[16] The level of ECC support is package dependent.

SDRAM Controller Subsystem Block Diagram

The SDRAM controller subsystem connects to the MPU subsystem, the main switch of the L3 interconnect, and the FPGA fabric. The memory interface consists of the SDRAM controller, the physical layer (PHY), control and status registers (CSRs), and their associated interfaces.

Figure 11-1: SDRAM Controller Subsystem High-Level Block Diagram



SDRAM Controller

The SDRAM controller provides high performance data access and run-time programmability. The controller reorders data to reduce row conflicts and bus turn-around time by grouping read and write transactions together, allowing for efficient traffic patterns and reduced latency.

The SDRAM controller consists of a multiport front end (MPFE) and a single-port controller. The MPFE provides multiple independent interfaces to the single-port controller. The single-port controller communicates with and manages each external memory device.

The MPFE FPGA-to-HPS SDRAM interface port has an asynchronous FIFO buffer followed by a synchronous FIFO buffer. Both the asynchronous and synchronous FIFO buffers have a read and write data FIFO depth of 8, and a command FIFO depth of 4. The MPU sub-system 64-bit AXI and L3 interconnect 32-bit AXI have asynchronous FIFO buffers with read and write data FIFO depth of 8, and command FIFO depth of 4.

DDR PHY

The DDR PHY provides a physical layer interface for read and write memory operations between the memory controller and memory devices. The DDR PHY has dataflow components, control components, and calibration logic that handle the calibration for the SDRAM interface timing.

Related Information

Memory Controller Architecture on page 11-6

SDRAM Controller Memory Options

Bank selects, and row and column address lines can be configured to work with SDRAMs of various technology and density combinations.

Table 11-1: SDRAM Controller Interface Memory Options

Memory Type [17]	Mbits	Column Address Bit Width	Bank Select Bit Width	Row Address Bit Width	MBytes	Page Size
DDR2	256	10	2	13	32	1024
	512	10	2	14	64	1024
	1024 (1 Gb)	10	3	14	128	1024
	2048 (2 Gb)	10	3	15	256	1024
	4096 (4 Gb)	10	3	16	512	1024
DDR3	512	10	3	13	64	1024
	1024 (1 Gb)	10	3	14	128	1024
	2048 (2 Gb)	10	3	15	256	1024
	4096 (4 Gb)	10	3	16	512	1024

[17] For all memory types shown in this table, the DQ width is 8.

Memory Type [17]	Mbits	Column Address Bit Width	Bank Select Bit Width	Row Address Bit Width	MBytes	Page Size
LPDDR2	64	9	2	12	8	512
	128	10	2	12	16	1024
	256	10	2	13	32	1024
	512	11	2	13	64	2048
	1024 (1 Gb) - S2[18]	11	2	14	128	2048
	1024 (1 Gb) - S4 [19]	11	3	13	128	2048
	2048 (2 Gb) - S2 [18]	11	2	15	256	2048
	2048 (2 Gb) -S4 [19]	11	3	14	256	2048
	4096 (4 Gb)	12	3	14	512	4096

SDRAM Controller Subsystem Interfaces

MPU Subsystem Interface

The SDRAM controller connects to the MPU subsystem with a dedicated 64-bit AXI interface, operating on the `mpu_l2_ram_clk` clock domain.

L3 Interconnect Interface

The SDRAM controller interfaces to the L3 interconnect with a dedicated 32-bit AXI interface, operating on the `l3_main_clk` clock domain.

Related Information

System Interconnect on page 7-1

[17] For all memory types shown in this table, the DQ width is 8.

[18] S2 signifies a 2n prefetch size

[19] S4 signifies a 4n prefetch size

CSR Interface

The CSR interface connects to the level 4 (L4) bus and operates on the `l4_sp_clk` clock domain. The MPU subsystem uses the CSR interface to configure the controller and PHY, for example, setting the memory timing parameter values or placing the memory to a low power state. The CSR interface also provides access to the status registers in the controller and PHY.

FPGA-to-HPS SDRAM Interface

The FPGA-to-HPS SDRAM interface provides masters implemented in the FPGA fabric access to the SDRAM controller subsystem in the HPS. The interface has three ports types that are used to construct the following AXI or Avalon-MM interfaces:

- Command ports—issue read and/or write commands, and for receive write acknowledge responses
- 64-bit read data ports—receive data returned from a memory read
- 64-bit write data ports—transmit write data

The FPGA-to-HPS SDRAM interface supports six command ports, allowing up to six Avalon-MM interfaces or three AXI interfaces. Each command port can be used to implement either a read or write command port for AXI, or be used as part of an Avalon-MM interface. The AXI and Avalon-MM interfaces can be configured to support 32-, 64-, 128-, and 256-bit data.

Table 11-2: FPGA-to-HPS SDRAM Controller Port Types

Port Type	Available Number of Ports
Command	6
64-bit read data	4
64-bit write data	4

The FPGA-to-HPS SDRAM controller interface can be configured with the following characteristics:

- Avalon-MM interfaces and AXI interfaces can be mixed and matched as required by the fabric logic, within the bounds of the number of ports provided to the fabric.
- Because the AXI protocol allows simultaneous read and write commands to be issued, two SDRAM control ports are required to form an AXI interface.
- Because the data ports are natively 64-bit, they must be combined if wider data paths are required for the interface.
- Each Avalon-MM or AXI interface of the FPGA-to-HPS SDRAM interface operates on an independent clock domain.
- The FPGA-to-HPS SDRAM interfaces are configured during FPGA configuration.

The following table shows the number of ports needed to configure different bus protocols, based on type and data width.

Table 11-3: FPGA-to-HPS SDRAM Port Utilization

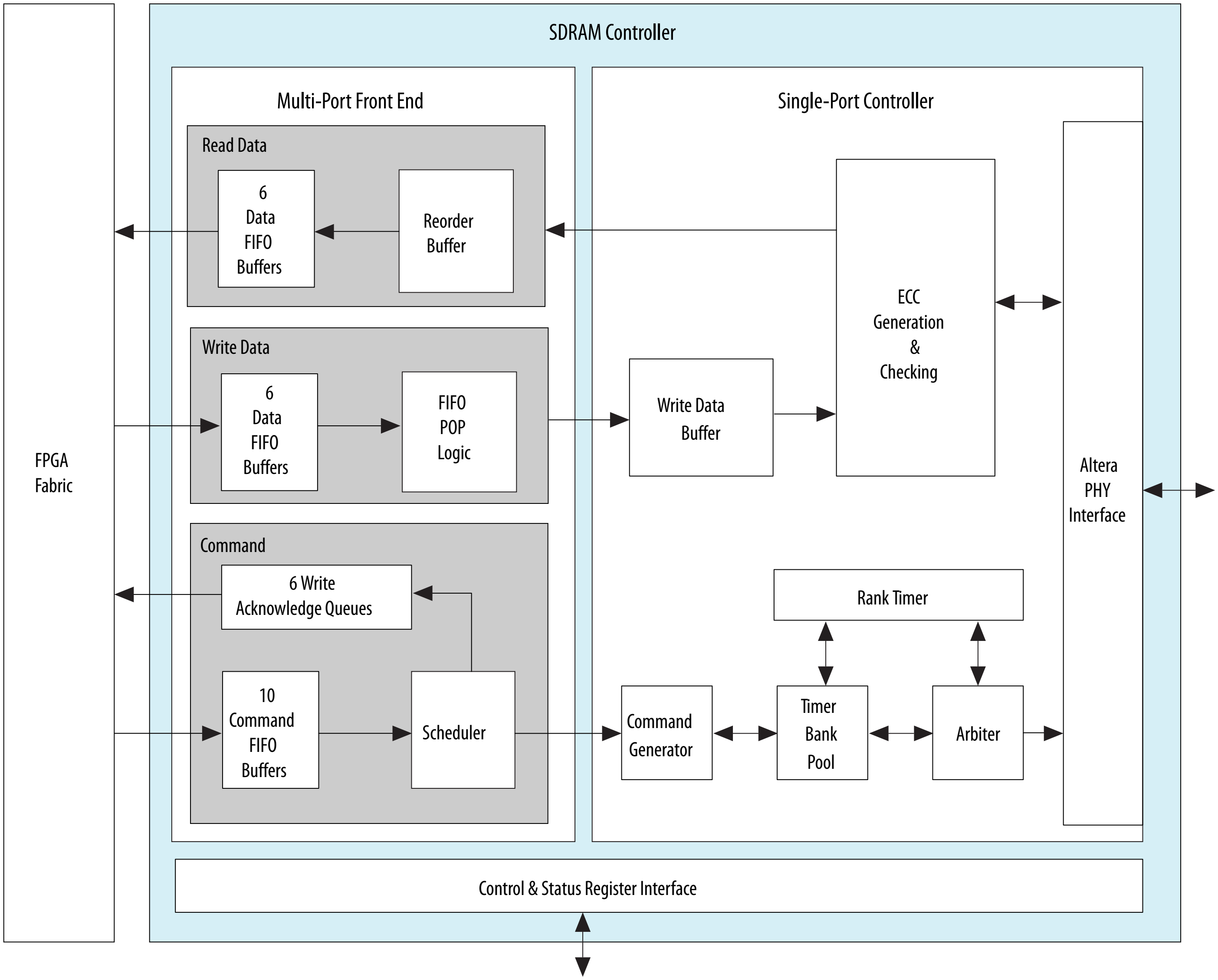
Bus Protocol	Command Ports	Read Data Ports	Write Data Ports
32- or 64-bit AXI	2	1	1

Bus Protocol	Command Ports	Read Data Ports	Write Data Ports
128-bit AXI	2	2	2
256-bit AXI	2	4	4
32- or 64-bit Avalon-MM	1	1	1
128-bit Avalon-MM	1	2	2
256-bit Avalon-MM	1	4	4
32- or 64-bit Avalon-MM write-only	1	0	1
128-bit Avalon-MM write-only	1	0	2
256-bit Avalon-MM write-only	1	0	4
32- or 64-bit Avalon-MM read-only	1	1	0
128-bit Avalon-MM read-only	1	2	0
256-bit Avalon-MM read-only	1	4	0

Memory Controller Architecture

The SDRAM controller consists of an MPFE, a single-port controller, and an interface to the CSRs.

Figure 11-2: SDRAM Controller Block Diagram



Multi-Port Front End

The Multi-Port Front End (MPFE) is responsible for scheduling pending transactions from the configured interfaces and sending the scheduled memory transactions to the single-port controller. The MPFE handles all functions related to individual ports.

The MPFE consists of three primary sub-blocks.

Command Block

The command block accepts read and write transactions from the FPGA fabric and the HPS. When the command FIFO buffer is full, the command block applies backpressure by deasserting the ready signal. For each pending transaction, the command block calculates the next SDRAM burst needed to progress on that transaction. The command block schedules pending SDRAM burst commands based on the user-supplied configuration, available write data, and unallocated read data space.

Write Data Block

The write data block transmits data to the single-port controller. The write data block maintains write data FIFO buffers and clock boundary crossing for the write data. The write data block informs the command block of the amount of pending write data for each transaction so that the command block can calculate eligibility for the next SDRAM write burst.

Read Data Block

The read data block receives data from the single-port controller. Depending on the port state, the read data block either buffers the data in its internal buffer or passes the data straight to the clock boundary crossing FIFO buffer. The read data block reorders out-of-order data for Avalon-MM ports.

In order to prevent the read FIFO buffer from overflowing, the read data block informs the command block of the available buffer area so the command block can pace read transaction dispatch.

Single-Port Controller

The single-port logic is responsible for following actions:

- Queuing the pending SDRAM bursts
- Choosing the most efficient burst to send next
- Keeping the SDRAM pipeline full
- Ensuring all SDRAM timing parameters are met

Transactions passed to the single-port logic for a single page in SDRAM are guaranteed to be executed in order, but transactions can be reordered between pages. Each SDRAM burst read or write is converted to the appropriate Altera PHY interface (AFI) command to open a bank on the correct row for the transaction (if required), execute the read or write command, and precharge the bank (if required).

The single-port logic implements command reordering (looking ahead at the command sequence to see which banks can be put into the correct state to allow a read or write command to be executed) and data reordering (allowing data transactions to be dispatched even if the data transactions are executed in an order different than they were received from the multi-port logic).

The single-port controller consists of eight sub-modules.

Command Generator

The command generator accepts commands from the MPFE and from the internal ECC logic, and provides those commands to the timer bank pool.

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

Timer Bank Pool

The timer bank pool is a parallel queue that operates with the arbiter to enable data reordering. The timer bank pool tracks incoming requests, ensures that all timing requirements are met, and, on receiving write-data-ready notifications from the write data buffer, passes the requests to the arbiter

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

Arbiter

The arbiter determines the order in which requests are passed to the memory device. When the arbiter receives a single request, that request is passed immediately. When multiple requests are received, the arbiter uses arbitration rules to determine the order to pass requests to the memory device.

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

Rank Timer

The rank timer performs the following functions:

- Maintains rank-specific timing information
- Ensures that only four activates occur within a specified timing window
- Manages the read-to-write and write-to-read bus turnaround time
- Manages the time-to-activate delay between different banks

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

Write Data Buffer

The write data buffer receives write data from the MPFE and passes the data to the PHY, on approval of the write request.

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

ECC Block

The ECC block consists of an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC block can correct single- bit errors and detect double-bit errors resulting from noise or other impairments during data transmission.

Note: The level of ECC support is package dependent.

Related Information

- Memory Controller Architecture on page 11-6
  For more information, refer to the SDRAM Controller Block diagram.
- **http://www.altera.com/literature/hb/cyclone-v/cv_52006.pdf**

AFI Interface

The AFI interface provides communication between the controller and the PHY.

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

CSR Interface

The CSR interface is accessible from the L4 bus. The interface allows code executing in the HPS MPU and FPGA fabric to configure and monitor the SDRAM controller.

Related Information

Memory Controller Architecture on page 11-6
For more information, refer to the SDRAM Controller Block diagram.

Functional Description of the SDRAM Controller Subsystem

MPFE Operation Ordering

Operation ordering is defined and enforced within a port, but not between ports. All transactions received on a single port for overlapping addresses execute in order. Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

Avalon-MM does not support write acknowledgement. When a port is configured to support Avalon-MM, you should read from the location that was previously written to ensure that the write operation has completed. When a port is configured to support AXI, the master accessing the port can safely issue a read operation to the same address as a write operation as soon as the write has been acknowledged. To keep write latency low, writes are acknowledged as soon as the transaction order is guaranteed—meaning that any operations received on any port to the same address as the write operation are executed after the write operation.

To reduce read latency, the single-port logic can return read data out of order to the multi-port logic. The returned data is rearranged to its initial order on a per port basis by the multi-port logic and no traffic reordering occurs between individual ports.

Read Data Handling

The MPFE contains a read buffer shared by all ports. If a port is capable of receiving returned data then the read buffer is bypassed. If the size of a read transaction is smaller than twice the memory interface width, the buffer RAM cannot be bypassed. The lowest memory latency occurs when the port is ready to receive data and the full width of the interface is utilized.

MPFE Multi-port Scheduling

Multi-port scheduling is governed by two factors: the absolute priority of a transaction and the weighting of a port.

The evaluation of absolute priority ensures that ports carrying higher-priority traffic are served ahead of ports carrying lower-priority traffic. The scheduler recognizes eight priority levels (0-7), with higher values representing higher priorities. For example, any transaction with priority seven is scheduled before transactions of priority six or lower. The absolute priority for each port is configured by programming the `userpriority` field of the `mppriority` register.

When ports carry traffic of the same absolute priority, relative priority is determined based on port weighting. The static weight of a port and the sum of static weights are configured by programming the `mpweight_*_*` registers. Port weighting is a five-bit value (0-31), and is determined by a deficit-weighted round robin (DWRR) algorithm, which corrects for past over-servicing or under-servicing of a port. Each port has an associated weight which is updated every cycle, with a user-configured weight added to it and the amount of traffic served subtracted from it. The port with the highest weighting is considered the most eligible.

To ensure that high-priority traffic is served quickly and that long and short bursts are effectively interleaved between ports, incoming transactions longer than a single SDRAM burst are scheduled as a series of SDRAM bursts, with each burst arbitrated separately.

To ensure that lower priority ports do not build up large running weights while higher priority ports monopolize bandwidth, the controller's DWRR weights are updated only when a port matches the scheduled priority. Therefore, if three ports are being accessed, two being priority seven and one being priority four, the weights for both ports at priority seven are updated but the port with priority four remains unchanged.

Multi-port scheduling is performed between all of the ports connected to the FPGA fabric and internally in the HPS to determine which transaction is serviced next. Arbitration is performed on a SDRAM burst basis to ensure that a long transaction does not lock other transactions or cause latency to significantly increase for high-priority ports.

Arbitration supports both absolute and relative priority. Absolute priority is intended for applications where one master should always get priority above or below others. Relative priority is supported through a programmable weight field which controls scheduling between ports at the same priority.

The scheduler is work-conserving. Write operations can only be scheduled when enough data for the SDRAM burst has been received. Read operations can only be scheduled when sufficient internal memory is free and the port is not occupying too much of the read buffer.

The multi-port scheduling configuration can be updated while transactions are still active in the memory controller. Both priority and weight for a port can be updated without interrupting traffic on a port. Updates are used in scheduling decisions within 10 memory clock cycles of being updated, so priority can be updated frequently if needed. Multi-port priority is programmed in the `mppriority` register and port weight is configured in the `mpweight_*_*` registers.

Note: In addition to the `mppriority` register and `mpweight_*_*` registers, the `remappriority` register adds another level of priority to the port scheduling. By programming bit N in the `priorityremap` field of the `remappriority` register, any port with an absolute priority N is sent to the front of the single port command queue and is serviced before any other transaction. Please refer to the `remappriority` register for more details.

MPFE SDRAM Burst Scheduling

SDRAM burst scheduling recognizes addresses that access the same row/bank combination, known as open page accesses. Operations to a page are served in the order in which they are received by the single-port controller. Selection of SDRAM operations is a two-stage process. First, each pending transaction must wait for its timers to be eligible for execution. Next, the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High-priority operations take precedence over lower-priority operations
- If multiple operations are in arbitration, read operations have precedence over write operations
- If multiple operations still exist, the oldest is served first

A high-priority transaction in the SDRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction's chip select, row, or column fields of the address do not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the chip select, row, and column fields match an earlier transaction, the high-priority transaction yields until the earlier transaction is completed.

Clocking

The FPGA fabric ports of the MPFE can be clocked at different frequencies. Synchronization is maintained by clock-domain crossing logic in the MPFE. Command ports can operate on different clock domains, but the data ports associated with a given command port must be attached to the same clock as that command port.

Note: A command port paired with a read and write port to form an Avalon-MM interface must operate at the same clock frequency as the data ports associated with it.

Single-Port Controller Operation

The single-port controller increases the performance of memory transactions through command and data re-ordering, enforcing bank policies, combining write operations and allowing burst transfers. Correction of single-bit errors and detection of double-bit errors is handled in the ECC module of the single-port Controller.

SDRAM Interface

The SDRAM interface is up to 40 bits wide and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations, depending on the device package. The SDRAM interface supports LPDDR2, DDR2, and DDR3 memory protocols.

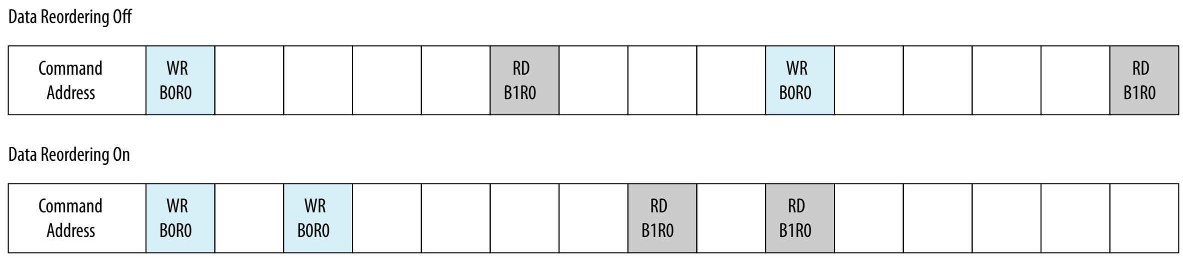
Command and Data Reordering

The heart of the SDRAM controller is a command and data reordering engine. Command reordering allows banks for future transactions to be opened before the current transaction finishes.

Data reordering allows transactions to be serviced in a different order than they were received when that new order allows for improved utilization of the SDRAM bandwidth. Operations to the same bank and row are performed in order to ensure that operations which impact the same address preserve the data integrity.

The following figure shows the relative timing for a write/read/write/read command sequence performed in order and then the same command sequence performed with data reordering. Data reordering allows the write and read operations to occur in bursts, without bus turnaround timing delay or bank reassignment.

Figure 11-3: Data Reordering Effect



Data Reordering Off

Command Address	WR B0R0					RD B1R0				WR B0R0					RD B1R0

Data Reordering On

Command Address	WR B0R0		WR B0R0					RD B1R0		RD B1R0					

The SDRAM controller schedules among all pending row and column commands every clock cycle.

Bank Policy

The bank policy of the SDRAM controller allows users to request that a transaction's bank remain open after an operation has finished so that future accesses do not delay in activating the same bank and row combination. The controller supports only eight simultaneously-opened banks, so an open bank might get closed if the bank resource is needed for other operations.

Open bank resources are allocated dynamically as SDRAM burst transactions are scheduled. Bank allocation is requested automatically by the controller when an incoming transaction spans multiple SDRAM bursts or by the extended command interface. When a bank must be reallocated, the least-recently-used open bank is used as the replacement.

If the controller determines that the next pending command will cause the bank request to not be honored, the bank might be held open or closed depending on the pending operation. A request to close a bank with a pending operation in the timer bank pool to the same row address causes the bank to remain open. A request to leave a bank open with a pending command to the same bank but a different row address causes a precharge operation to occur.

Write Combining

The SDRAM controller combines write operations from successive bursts on a port where the starting address of the second burst is one greater than the ending address of the first burst and the resulting burst length does not overflow the 11-bit burst-length counters.

Write combining does not occur if the previous bus command has finished execution before the new command has been received.

Burst Length Support

The controller supports burst lengths of 2, 4, 8, and 16, and data widths of 8, 16, and 32 bits for non-ECC operation, and widths of 24 and 40 operations with ECC enabled. The following table shows the type of SDRAM for each burst length.

Table 11-4: SDRAM Burst Lengths

Burst Length	SDRAM
4	LPDDR2, DDR2
8	DDR2, DDR3, LPDDR2
16	LPDDR2

Width Matching

The SDRAM controller automatically performs data width conversion.

ECC

The single-port controller supports memory ECC calculated by the controller.
The controller ECC employs standard Hamming logic to detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

Note: The level of ECC support is package dependent.

Functions the controller ECC provides are:

- Byte Writes
- ECC Write Backs
- Notification of ECC Errors

Byte Writes

The memory controller performs a read-modify-write operation to ensure that the ECC data remains valid when a subset of the bits of a word is being written.

Byte writes with ECC enabled are executed as a read-modify-write. Typical operations only use a single entry in the timer bank pool. Controller ECC enabled sub-word writes use two entries. The first operation is a read and the second operation is a write. These two operations are transferred to the timer bank pool with an address dependency so that the write cannot be performed until the read data has returned. This approach ensures that any subsequent operations to the same address (from the same port) are executed after the write operation, because they are ordered on the row list after the write operation.

If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

ECC Write Backs

If the controller ECC is enabled and a read operation results in a correctable ECC error, the controller corrects the location in memory, if write backs are enabled. The correction results in scheduling a new read-modify-write.

A new read is performed at the location to ensure that a write operation modifying the location is not overwritten. The actual ECC correction operation is performed as a read-modify-write operation. ECC write backs are enabled and disabled through the `cfg_enable_ecc_code_overwrites` field in the `ctrlcfg` register.

User Notification of ECC Errors

When an ECC error occurs, an interrupt signal notifies the MPU subsystem, and the ECC error information is stored in the status registers. The memory controller provides interrupts for single-bit and double-bit errors.

The status of interrupts and errors are recorded in status registers, as follows:

- The `dramsts` register records interrupt status.
- The `dramintr` register records interrupt masks.
- The `sbecount` register records the single-bit error count.
- The `dbecount` register records the double-bit error count.
- The `erraddr` register records the address of the most recent error.

Interleaving Options

The controller supports the following address-interleaving options:

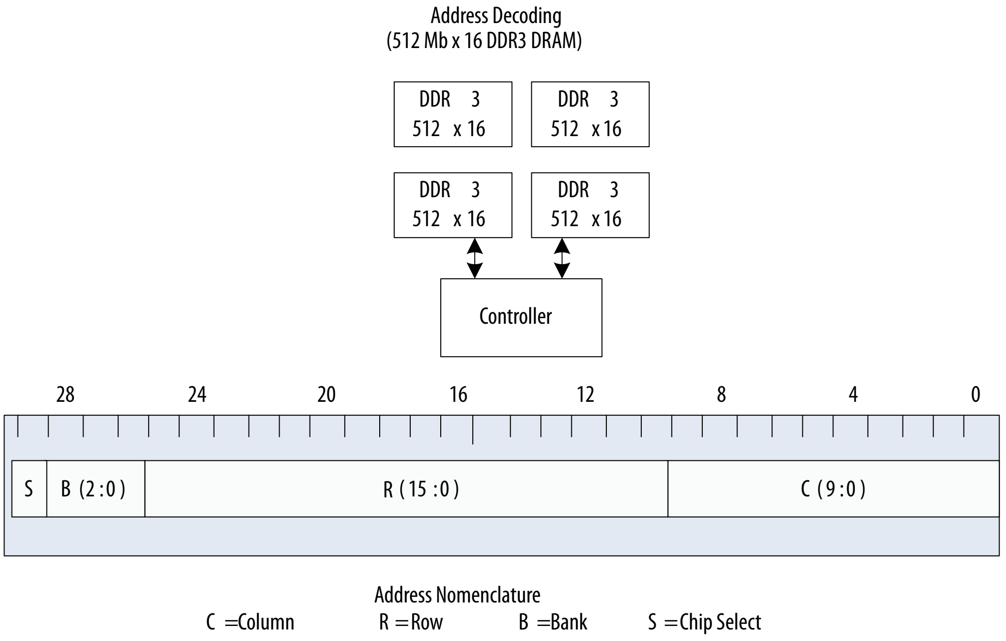
- Non-interleaved
- Bank interleave without chip select interleave
- Bank interleave with chip select interleave

The following interleaving examples use 512 megabits (Mb) x 16 DDR3 chips and are documented as byte addresses. For RAMs with smaller address fields, the order of the fields stays the same but the widths may change.

Non-interleaved

RAM mapping is non-interleaved.

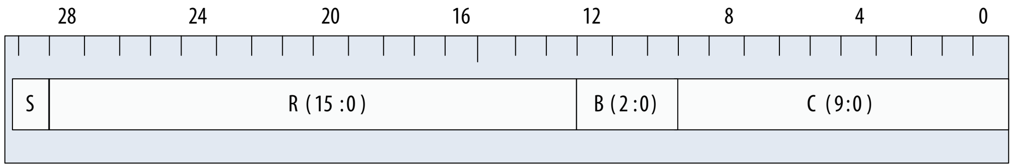
Figure 11-4: Non-interleaved Address Decoding



Bank Interleave Without Chip Select Interleave

Bank interleave without chip select interleave swaps row and bank from the non-interleaved address mapping. This interleaving allows smaller data structures to spread across all banks in a chip.

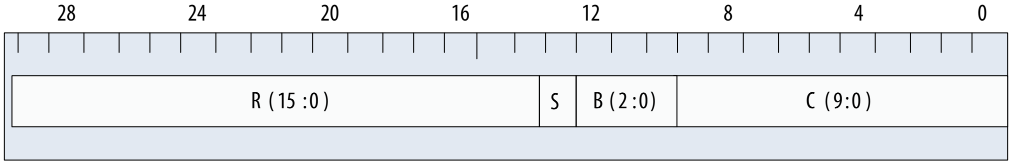
Figure 11-5: Bank Interleave Without Chip Select Interleave Address Decoding



Bank Interleave with Chip Select Interleave

Bank interleave with chip select interleave moves the row address to the top, followed by chip select, then bank, and finally column address. This interleaving allows smaller data structures to spread across multiple banks and chips (giving access to 16 total banks for multithreaded access to blocks of memory). Memory timing is degraded when switching between chips.

Figure 11-6: Bank Interleave With Chip Select Interleave Address Decoding



AXI-Exclusive Support

The single-port controller supports AXI-exclusive operations. The controller implements a table shared across all masters, which can store up to 16 pending writes. Table entries are allocated on an exclusive read and table entries are deallocated on a successful write to the same address by any master.

Any exclusive write operation that is not present in the table returns an exclusive fail as acknowledgement to the operation. If the table is full when the exclusive read is performed, the table replaces a random entry.

Note: When using AXI-exclusive operations, accessing the same location from Avalon-MM interfaces can result in unpredictable results.

Memory Protection

The single-port controller has address protection to allow the software to configure basic protection of memory from all masters in the system. If the system has been designed exclusively with AMBA masters, TrustZone® is supported. Ports that use Avalon-MM can be configured for port level protection.

Memory protection is based on physical addresses in memory. The single-port controller can configure up to 20 rules to allow or prevent masters from accessing a range of memory based on their AxIDs, level of security and the memory region being accessed. If no rules are matched in an access, then default settings take effect.

The rules are stored in an internal protection table and can be accessed through indirect addressing offsets in the `protruledwr` register in the CSR. To read a specific rule, set the `readrule` bit and write the appropriate offset in the `ruleoffset` field of the `protruledwr` register.

To write a new rule, three registers in the CSR must be configured:

1. The `protportdefault` register is programmed to control the default behavior of memory accesses when no rules match. When a bit is clear, all default accesses from that port pass. When a bit is set, all default accesses from that port fails. The bits are assigned as follows:

Table 11-5: `protportdefault` register

Bits	Description
`31:10`	reserved
`9`	When this bit is set to 1, deny CPU writes during a default transaction. When this bit is clear, allow CPU writes during a default transaction.

Bits	Description
8	When this bit is set to 1, deny L3 writes during a default transaction. When this bit is clear, allow L3 writes during a default transaction.
7	When this bit is set to 1, deny CPU reads during a default transaction. When this bit is clear, allow CPU reads during a default transaction.
6	When this bit is set to 1, deny L3 reads during a default transaction. When this bit is clear, allow L3 reads during a default transaction.
5:0	When this bit is set to 1, deny accesses from FPGA-to-SDRAM ports 0 through 5 during a default transaction. When this bit is clear, allow accesses from FPGA-to-SDRAM ports 0 through 5 during a default transaction.

2. The `protruleid` register gives the bounds of the AxID value that allows an access
3. The `protruledata` register configures the specific security characteristics for a rule.

Once the registers are configured, they can be committed to the internal protection table by programming the `ruleoffset` field and setting the `writerule` bit in the `protruledwr` register.

Secure and non-secure regions are specified by rules containing a starting address and ending address with 1 MB boundaries for both addresses. You can override the port defaults and allow or disallow all transactions.

The following table lists the fields that you can specify for each rule.

Table 11-6: Fields for Rules in Memory Protection Table

Field	Width	Description
Valid	1	Set to 1 to activate the rule. Set to 0 to deactivate the rule.
Port Mask [20]	10	Specifies the set of ports to which the rule applies, with one bit representing each port, as follows: bits 0 to 5 correspond to FPGA fabric ports 0 to 5, bit 6 corresponds to AXI L3 switch read, bit 7 is the CPU read, bit 8 is L3 switch write, and bit 9 is the CPU write.

[20] Although AxID and Port Mask could be redundant, including both in the table allows possible compression of rules. If masters connected to a port do not have contiguous AxIDs, a port-based rule might be more efficient than an AxID-based rule, in terms of the number of rules needed.

Field	Width	Description
AxID_ low[20]	12	Low transfer AxID of the rules to which this rule applies. Incoming transactions match if they are greater than or equal to this value. Ports with smaller AxIDs have the AxID shifted to the lower bits and zero padded at the top.
AxID_ high [20]	12	High transfer AxID of the rules to which this rule applies. Incoming transactions match if they are less than or equal to this value.
Address_ low	12	Points to a 1MB block and is the lower address. Incoming addresses match if they are greater than or equal to this value.
Address_ high	12	Upper limit of address. Incoming addresses match if they are less than or equal to this value.
Protection	2	A value of 0x0 indicates that the protection bit is not set; a value of 0x1 sets the protection bit. Values 0x2 and 0x3 set the region to shared, meaning both secure and non-secure accesses are valid.
Fail/allow	1	Set this value to 1 to force the operation to fail or succeed.

A port has a default access status of either allow or fail, and rules with the opposite allow/fail value can override the default. The system evaluates each transaction against every rule in the memory protection table. A transaction received on a port which by default allows access, would fail only if a rule with the fail bit matches the transaction. Conversely, a port which by default prevents access, would allow access only if a rule allows that transaction to pass.

Exclusive transactions are security checked on the read operation only. A write operation can occur only if a valid read is marked in the internal exclusive table. Consequently, a master performing an exclusive read followed by a write, can write to memory only if the exclusive read was successful.

Related Information

ARM TrustZone

For more information about TrustZone refer to the ARM web page.

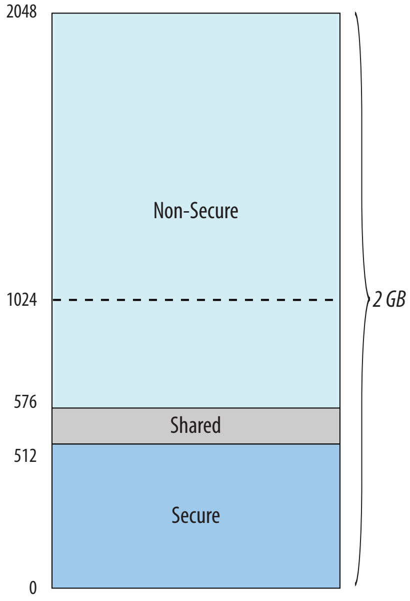
Example of Configuration for TrustZone

For a TrustZone configuration, memory isTrustZone divided into a range of memory accessible by secure masters and a range of memory accessible by non-secure masters. The two memory address ranges may have a range of memory that overlaps.

This example implements the following memory configuration:

- 2 GB total RAM size
- 0—512 MB dedicated secure area
- 513—576 MB shared area
- 577—2048 MB dedicated non-secure area

Figure 11-7: Example Memory Configuration



In this example, each port is configured by default to disallow all accesses. The following table shows the two rules programmed into the memory protection table.

Table 11-7: Rules in Memory Protection Table for Example Configuration

Rule #	Port Mask	AxID Low	AxID High	Address Low	Address High	Prot	Fail/Allow
1	0x3FF (1023)	0x000	0xFFF (4095)	0	576	1 (non-secure)	allow
2	0x3FF (1023)	0x000	0xFFF (4095)	512	2047	0 (secure)	allow

The port mask value, AxID Low, and AxID High, apply to all ports and all transfers within those ports. Each access request is evaluated against the memory protection table, and will fail unless there is a rule match allowing a transaction to complete successfully.

The following table shows the result for a sample set of transactions.

Table 11-8: Result for a Sample Set of Transactions

Operation	Source	Address Accesses	Security Access Type	Result	Comments
Read	CPU	4096	secure	Allow	Matches rule 1.
Write	CPU	536, 870, 912	secure	Allow	Matches rule 1.

Operation	Source	Address Accesses	Security Access Type	Result	Comments
Write	L3 attached masters	605, 028, 350	secure	Fail	Does not match rule 1 (out of range of the address field), does not match rule 2 (protection bit incorrect).
Read	L3 attached masters	4096	non-secure	Fail	Does not match rule 1 (prot value wrong), does not match rule 2 (not in address range).
Write	CPU	536, 870, 912	non-secure	Allow	Matches rule 2.
Write	L3 attached masters	605, 028, 350	non-secure	Allow	Matches rule 2.

If you did not want any overlap between the memory blocks, you could specify the address ranges in the two rules of the Memory Protection Table to be mutually exclusive. Depending on your desired TrustZone configuration, you can add rules to the memory protection table to create multiple blocks of protected or unprotected space.

SDRAM Power Management

The SDRAM controller subsystem supports the following power saving features in the SDRAM:

- Partial array self-refresh (PASR)
- Power down
- Deep power down for LPDDR2

To enable self-refresh for the memories of one or both chip selects, program the `selfshreq` bit and the `sefrfshmask` bit in the `lowpwreq` register.

Power-saving mode initiates either due to a user command or from inactivity. The number of idle clock cycles after which a memory can be put into power-down mode is programmed through the `autopdycycles` field of the `lowpwrtiming` register.

Power-down mode forces the SDRAM burst-scheduling bank-management logic to close all banks and issue the power down command. The SDRAM automatically reactivates when an active SDRAM command is received.

To enable deep power down request for the LPDDR2 memories of one or both chip selects, program the `deeppwrdnreq` bit and the `deepwrdnmask` field of the `lowpwreq` register.

Other power-down modes are performed only under user control.

DDR PHY

The DDR PHY connects the memory controller and external memory devices in the speed critical command path.

The DDR PHY implements the following functions:

- Calibration—the DDR PHY supports the JEDEC-specified steps to synchronize the memory timing between the controller and the SDRAM chips. The calibration algorithm is implemented in software.
- Memory device initialization—the DDR PHY performs the mode register write operations to initialize the devices. The DDR PHY handles re-initialization after a deep power down.
- Single-data-rate to double-data-rate conversion.

Clocks

All clocks are assumed to be asynchronous with respect to the `ddr_dqs_clk` memory clock. All transactions are synchronized to memory clock domain.

Table 11-9: SDRAM Controller Subsystem Clock Domains

Clock Name	Description
`ddr_dq_clk`	Clock for PHY
`ddr_dqs_clk`	Clock for MPFE, single-port controller, CSR access, and PHY
`ddr_2x_dqs_clk`	Clock for PHY that provides up to 2 times `ddr_dq_clk` frequency
`l4_sp_clk`	Clock for CSR interface
`mpu_l2_ram_clk`	Clock for MPU interface
`l3_main_clk`	Clock for L3 interface
`f2h_sdram_clk[5:0]`	Six separate clocks used for the FPGA-to-HPS SDRAM ports to the FPGA fabric

In terms of clock relationships, the FPGA fabric connects the appropriate clocks to write data, read data, and command ports for the constructed ports.

Related Information

Clock Manager on page 2-1

Resets

The SDRAM controller subsystem supports a full reset (cold reset) and a warm reset. The SDRAM controller can be configured to preserve memory contents during a warm reset.

To preserve memory contents, the reset manager can request that the single-port controller place the SDRAM in self-refresh mode prior to issuing the warm reset. If self-refresh mode is enabled before the warm reset to preserve memory contents, the PHY and the memory timing logic is not reset, but the rest of the controller is reset.

Related Information

Reset Manager on page 3-1

Port Mappings

The memory interface controller has a set of command, read data, and write data ports that support AXI3, AXI4 and Avalon-MM. Tables are provided to identify port assignments and functions.

Table 11-10: Command Port Assignments

Command Port	Allowed Functions
0, 2, 4	FPGA fabric AXI read command ports FPGA fabric Avalon-MM read or write command ports
1, 3, 5	FPGA fabric AXI read command ports FPGA fabric Avalon-MM read or write command ports
6	L3 AXI read command port
7	MPU AXI read command port
8	L3 AXI write command port
9	MPU AXI write command port

Table 11-11: Read Port Assignments

Read Port	Allowed Functions
0, 1, 2, 3	64-bit read data from the FPGA fabric. When 128-bit data read ports are created, then read data ports 0 and1 get paired as well as 2 and 3.
4	32-bit L3 read data port
5	64-bit MPU read data port

Table 11-12: Write Port Assignments

Write Port	Allowed Functions
0, 1, 2, 3	64-bit write data from the FPGA fabric. When 128-bit data write ports are created, then write data ports 0 and 1 get paired as well as 2 and 3.
4	32-bit L3 write data port
5	64-bit MPU write data port

Initialization

The SDRAM controller subsystem has control and status registers (CSRs) which control the operation of the controller including DRAM type, DRAM timing parameters and relative port priorities. It also has a small set of bits which depend on the FPGA fabric to configure ports between the memory controller and the FPGA fabric; these bits are set for you when you configure your implementation using the HPS GUI in Qsys.

The CSRs are configured using a dedicated slave interface, which provides access to the registers. This region controls all SDRAM operation, MPFE scheduler configuration, and PHY calibration.

The FPGA fabric interface configuration is programmed into the FPGA fabric and the values of these register bits can be read by software. The ports can be configured without software developers needing to know how the FPGA-to-HPS SDRAM interface has been configured.

FPGA-to-SDRAM Protocol Details

The following topics summarize signals for the Avalon-MM Bidirectional port, Avalon-MM Write Port, Avalon-MM Read Port, and AXI port.

Note: If your device has multiple FPGA hardware images, then the same FPGA-to-SDRAM port configuration should be used across all designs.

Avalon-MM Bidirectional Port

The Avalon-MM bidirectional ports are standard Avalon-MM ports used to dispatch read and write operations.

Each configured Avalon-MM bidirectional port consists of the signals listed in the following table.

Table 11-13: Avalon-MM Bidirectional Port Signals

Name	Bit Width	Input/Output Direction	Function
`clk`	1	In	Clock for the Avalon-MM interface
`read`	1	In	Indicates read transaction [21]

[21] The Avalon-MM protocol does not allow read and write transactions to be posted concurrently.

Name	Bit Width	Input/Output Direction	Function
`write`	1	In	Indicates write transaction [21]
`address`	32	In	Address of the transaction
`readdata`	32, 64, 128, or 256	Out	Read data return
`readdatavalid`	1	Out	Indicates the `readdata` signal contains valid data in response to a previous read request.
`writedata`	32, 64, 128, or 256	In	Write data for a transaction
`byteenable`	4, 8, 16, 32	In	Byte enables for each write byte lane
`waitrequest`	1	Out	Indicates need for additional cycles to complete a transaction
`burstcount`	11	In	Transaction burst length. The value of the maximum `burstcount` parameter must be a power of 2.

The read and write interfaces are configured to the same size. The byte-enable size scales with the data bus size.

Related Information

Avalon Interface Specifications
Information about the Avalon-MM protocol

Avalon-MM Write-Only Port

The Avalon-MM write-only ports are standard Avalon-MM ports used to dispatch write operations. Each configured Avalon-MM write port consists of the signals listed in the following table.

Table 11-14: Avalon-MM Write-Only Port Signals

Name	Bits	Direction	Function
`reset`	1	In	Reset
`clk`	1	In	Clock
`write`	1	In	Indicates write transaction

Name	Bits	Direction	Function
address	32	In	Address of the transaction
writedata	32, 64, 128, or 256	In	Write data for a transaction
byteenable	4, 8, 16, 32	In	Byte enables for each write byte
waitrequest	1	Out	Indicates need for additional cycles to complete a transaction
burstcount	11	In	Transaction burst length

Related Information

Avalon Interface Specifications

Information about the Avalon-MM protocol

Avalon-MM Read Port

The Avalon-MM read ports are standard Avalon-MM ports used only to dispatch read operations. Each configured Avalon-MM read port consists of the signals listed in the following table.

Table 11-15: Avalon-MM Read Port Signals

Name	Bits	Direction	Function
reset	1	In	Reset
clk	1	In	Clock
read	1	In	Indicates read transaction
address	32	In	Address of the transaction
readdata	32, 64, 128, or 256	Out	Read data return
readdatavalid	1	Out	Flags valid cycles for read data return
waitrequest	1	Out	Indicates the need for additional cycles to complete a transaction. Needed for read operations when delay is needed to accept the read command.
burstcount	11	In	Transaction burst length

Related Information

Avalon Interface Specifications

Information about the Avalon-MM protocol

AXI Port

The AXI port uses an AXI-3 interface. Each configured AXI port consists of the signals listed in the following table. Each AXI interface signal is independent of the other interfaces for all signals, including clock and reset.

Table 11-16: AXI Port Signals

Name	Bits	Direction	Function
`ARESETn`	1	In	Reset
`ACLK`	1	In	Clock
Name	**Bits**	**Direction**	**Function**
Write Address Channel Signals			
`AWID`	4	In	Write identification tag
`AWADDR`	32	In	Write address
`AWLEN`	4	In	Write burst length
`AWSIZE`	3	In	Width of the transfer size
`AWBURST`	2	In	Burst type
`AWLOCK`	2	In	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
`AWCACHE`	4	In	Cache policy type
`AWPROT`	3	In	Protection-type signal used to indicate whether a transaction is secure or non-secure

Name	Bits	Direction	Function
AWREADY	1	Out	Indicates ready for a write command
AWVALID	1	In	Indicates valid write command.
Name	Bits	Direction	Function
Write Data Channel Signals			
WID	4	In	Write data transfer ID
WDATA	32, 64, 128 or 256	In	Write data
WSTRB	4, 8, 16, 32	In	Byte-based write data strobe. Each bit width corresponds to 8 bit wide transfer for 32-bit wide to 256-bit wide transfer.
WLAST	1	In	Last transfer in a burst
WVALID	1	In	Indicates write data and strobes are valid
WREADY	1	Out	Indicates ready for write data and strobes
Name	Bits	Direction	Function
Write Response Channel Signals			
BID	4	Out	Write response transfer ID
BRESP	2	Out	Write response status
BVALID	1	Out	Write response valid signal
BREADY	1	In	Write response ready signal
Name	Bits	Direction	Function
Read Address Channel Signals			

Name	Bits	Direction	Function
ARID	4	In	Read identification tag
ARADDR	32	In	Read address
ARLEN	4	In	Read burst length
ARSIZE	3	In	Width of the transfer size
ARBURST	2	In	Burst type
ARLOCK	2	In	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
ARCACHE	4	In	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
ARPROT	3	In	Protection-type signal used to indicate whether a transaction is secure or non-secure
ARREADY	1	Out	Indicates ready for a read command
ARVALID	1	In	Indicates valid read command
Name	Bits	Direction	Function
Read Data Channel Signals			
RID	4	Out	Read data transfer ID
RDATA	32, 64, 128 or 256	Out	Read data
RRESP	2	Out	Read response status

Name	Bits	Direction	Function
RLAST	1	Out	Last transfer in a burs
RVALID	1	Out	Indicates read data is valid
RREADY	1	In	Read data channel ready signal

Related Information

ARM AMBA Open Specification

AMBA Open Specifications, including information about the AXI-3 interface

SDRAM Controller Subsystem Programming Model

SDRAM controller configuration occurs through software programming of the configuration registers using the CSR interface.

HPS Memory Interface Architecture

The configuration and initialization of the memory interface by the ARM processor is a significant difference compared to the FPGA memory interfaces, and results in several key differences in the way the HPS memory interface is defined and configured.

Boot-up configuration of the HPS memory interface is handled by the initial software boot code, not by the FPGA programmer, as is the case for the FPGA memory interfaces. The Quartus II software is involved in defining the configuration of I/O ports which is used by the boot-up code, as well as timing analysis of the memory interface. Therefore, the memory interface must be configured with the correct PHY-level timing information. Although configuration of the memory interface in Qsys is still necessary, it is limited to PHY- and board-level settings.

HPS Memory Interface Configuration

To configure the external memory interface components of the HPS, open the HPS interface by selecting the Arria V/Cyclone V Hard Processor System component in Qsys. Within the HPS interface, select the EMIF tab to open the EMIF parameter editor.

The EMIF parameter editor contains four additional tabs: PHY Settings, Memory Parameters, Memory Timing, and Board Settings. The parameters available on these tabs are similar to those available in the parameter editors for non-SoC device families.

There are significant differences between the EMIF parameter editor for the Hard Processor System and the parameter editors for non-SoC devices, as follows:

- Because the HPS memory controller is not configurable through the Quartus II software, the Controller and Diagnostic tabs, which exist for non-SoC devices, are not present in the EMIF parameter editor for the hard processor system.
- Unlike the protocol-specific parameter editors for non-SoC devices, the EMIF parameter editor for the Hard Processor System supports multiple protocols, therefore there is an SDRAM Protocol parameter, where you can specify your external memory interface protocol. By default, the EMIF parameter editor assumes the DDR3 protocol, and other parameters are automatically populated with DDR3-appropriate values. If you select a protocol other than DDR3, change other associated parameter values appropriately.
- Unlike the memory interface clocks in the FPGA, the memory interface clocks for the HPS are initialized by the boot-up code using values provided by the configuration process. You may accept the values provided by UniPHY, or you may use your own PLL settings. If you choose to specify your own PLL settings, you must indicate that the clock frequency that UniPHY should use is the requested clock frequency, and not the achieved clock frequency calculated by UniPHY.

Note: The HPS does not support EMIF synthesis generation, compilation, or timing analysis. The HPS hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

HPS Memory Interface Simulation

Qsys provides a complete simulation model of the HPS memory interface controller and PHY, providing cycle-level accuracy, comparable to the simulation models for the FPGA memory interface.

The simulation model supports only the skip-cal simulation mode; quick-cal and full-cal are not supported. An example design is not provided, however you can create a test design by adding the traffic generator component to your design using Qsys. Also, the HPS simulation model does not use external memory pins to connect to the DDR memory model; instead, the memory model is incorporated directly into the HPS SDRAM interface simulation modules. The memory instance incorporated into the HPS model is in the simulation model hierarchy at: **hps_0/fpga_interfaces/f2sdram/hps_sdram_inst/mem/**

Simulation of the FPGA-to-SDRAM interfaces requires that you first bring the interfaces out of reset, otherwise transactions cannot occur. Connect the H2F reset to the F2S port resets and add a stage to your testbench to assert and deassert the H2F reset in the HPS. Appropriate Verilog code is shown below:

```
initial
begin
   // Assert reset
   <base name>.hps.fpga_interfaces.h2f_reset_inst.reset_assert();
   // Delay
   #1
   // Deassert reset
   <base name>.hps.fpga_interfaces.h2f_reset_inst.reset_deassert();
end
```

Generating a Preloader Image for HPS with EMIF

To generate a Preloader image for an HPS-based external memory interface, you must complete the following tasks:

- Create a Qsys project.
- Create a top-level file and add constraints.
- Create a Preloader BSP file.
- Create a Preloader image.

Creating a Qsys Project in Preparation for Generating a Preloader Image

This topic describes creating a Qsys project in preparation for generating a Preloader image.

1. On the **Tools** menu in the Quartus II software, click **Qsys**.
2. Under **Component library**, expand **Embedded Processor System**, select **Hard Processor System** and click **Add**.
3. Specify parameters for the **FPGA Interfaces**, **Peripheral Pin Multiplexing**, and **HPS Clocks**, based on your design requirements.
4. On the **SDRAM** tab, select the SDRAM protocol for your interface.
5. Populate the necessary parameter fields on the **PHY Settings**, **Memory Parameters**, **Memory Timing**, and **Board Settings** tabs.
6. Add other Qsys components in your Qsys design and make the appropriate bus connections.
7. Save the Qsys project.
8. Click **Generate** on the **Generation** tab, to generate the Qsys design.

Creating a Top-Level File and Adding Constraints

This topic describes adding your Qsys system to your top-level design and adding constraints to your design.

1. Add your Qsys system to your top-level design.
2. Add the Quartus II IP files (**.qip**) generated in step 2, to your Quartus II project.
3. Perform analysis and synthesis on your design.
4. Constrain your EMIF design by running the **<*variation_name*>_p0_pin_assignments.tcl** pin constraints script.
5. Add other necessary constraints—such as timing constraints, location assignments, and pin I/O standard assignments—for your design.
6. Compile your design to generate an SRAM object file (**.sof**) and the hardware handoff files necessary for creating a preloader image.

   Note: You must regenerate the hardware handoff files whenever the HPS configuration changes; for example, due to changes in Peripheral Pin Multiplexing or I/O standard for HPS pins.

Related Information

Altera SoC Embedded Design Suite User's Guide

For more information on how to create a preloader BSP file and image.

Debugging HPS SDRAM in the Preloader

To assist in debugging your design, tools are available at the preloader stage.

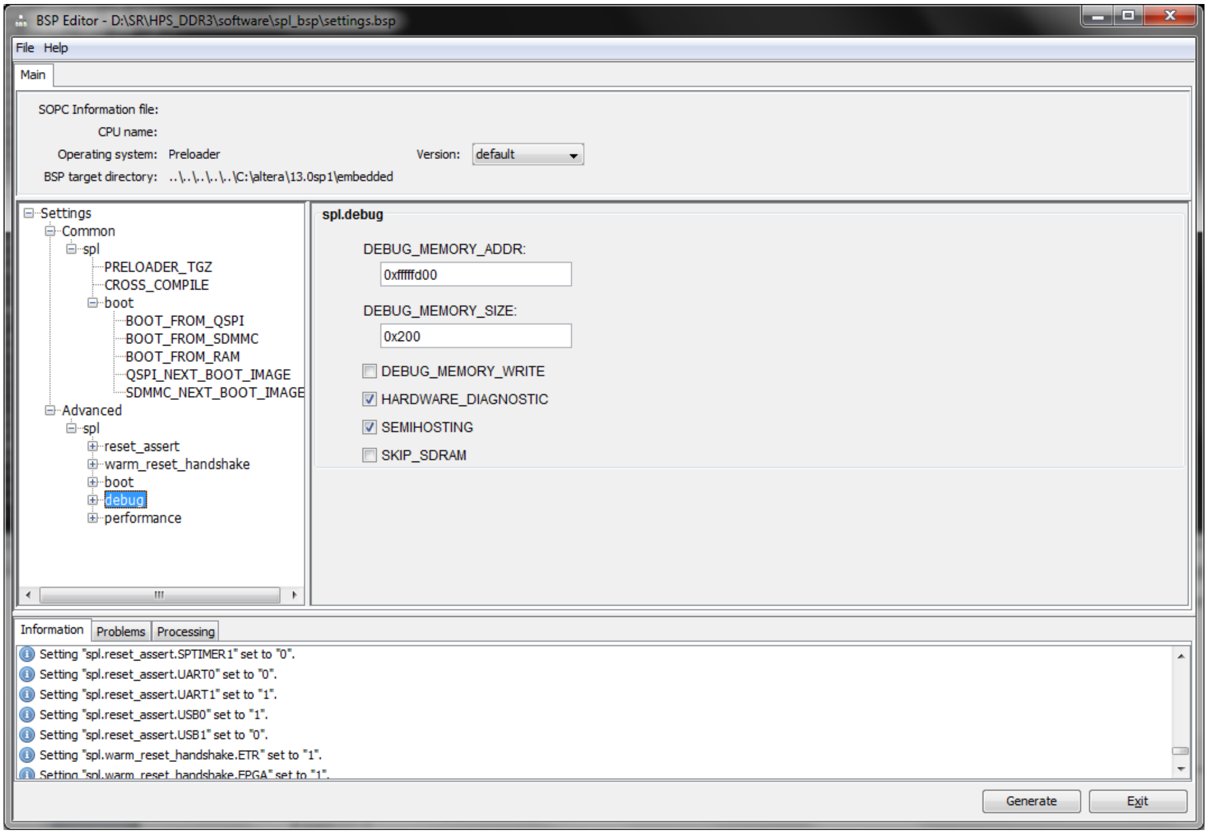
- UART or semihosting printout
- Simple memory test
- Debug report
- Predefined data patterns

The following topics provide procedures for implementing each of the above tools.

Enabling UART or Semihosting Printout

UART printout is enabled by default. If UART is not available on your system, you can use semihosting together with the debugger tool. To enable semihosting in the Preloader, follow these steps:

1. When you create the **.bsp** file in the BSP Editor, select **SEMIHOSTING** in the **spl.debug** window.



SOPC Information file:
CPU name:
Operating system: Preloader
Version: default
BSP target directory: ..\..\..\..\..\C:\altera\13.0sp1\embedded

spl.debug

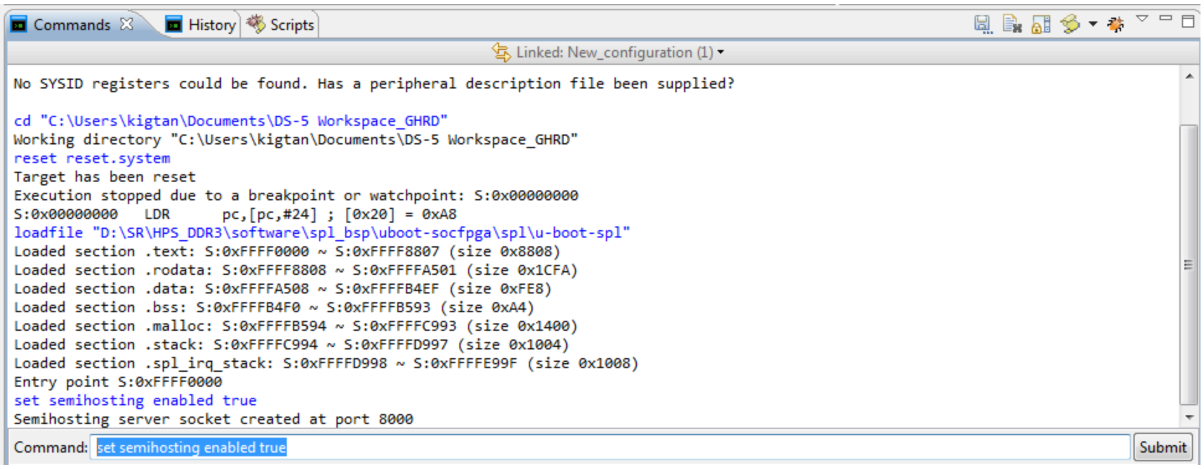
DEBUG_MEMORY_ADDR: 0xffffd00

DEBUG_MEMORY_SIZE: 0x200

☐ DEBUG_MEMORY_WRITE
☑ HARDWARE_DIAGNOSTIC
☑ SEMIHOSTING
☐ SKIP_SDRAM

```
Setting "spl.reset_assert.SPTIMER1" set to "0".
Setting "spl.reset_assert.UART0" set to "0".
Setting "spl.reset_assert.UART1" set to "1".
Setting "spl.reset_assert.USB0" set to "1".
Setting "spl.reset_assert.USB1" set to "0".
Setting "spl.warm_reset_handshake.ETR" set to "1".
Setting "spl.warm_reset_handshake.FPGA" set to "1".
```

2. Enable semihosting in the debugger, by typing `set semihosting enabled true` at the command line in the debugger.



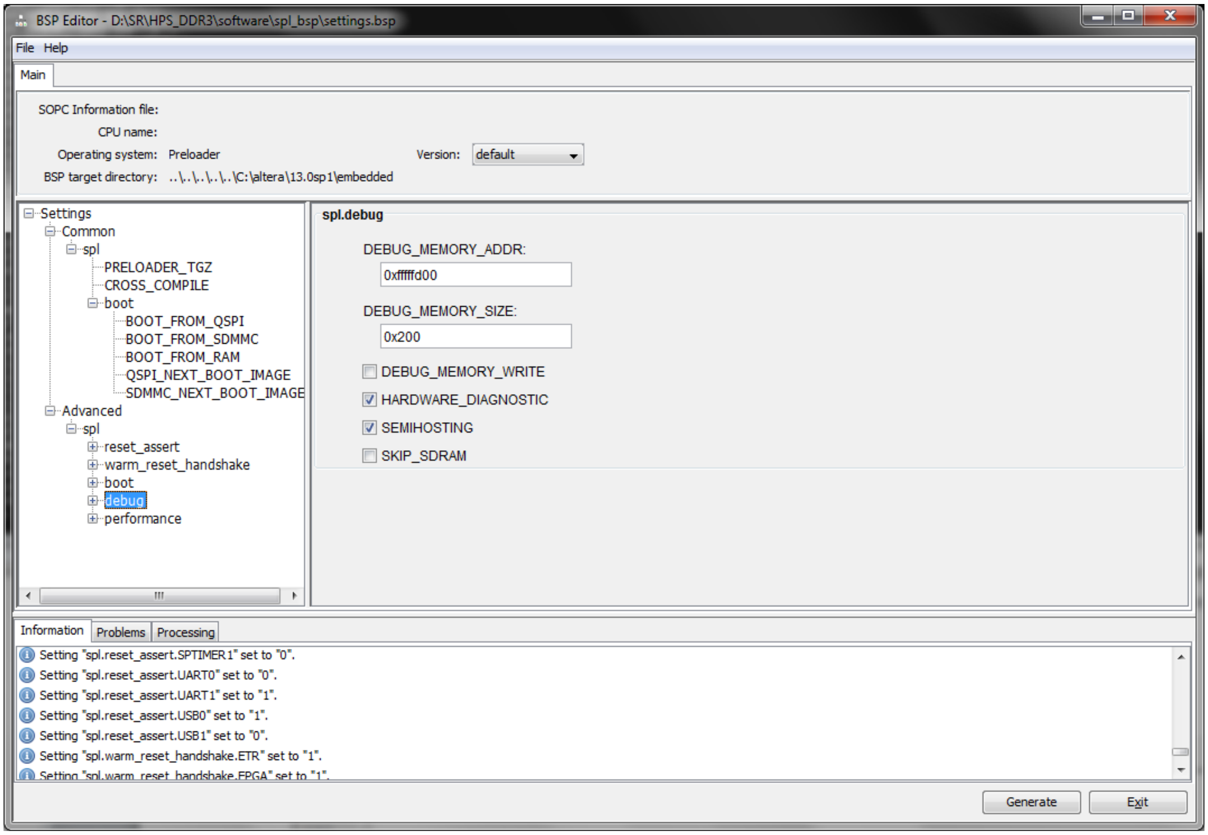
```
No SYSID registers could be found. Has a peripheral description file been supplied?

cd "C:\Users\kigtan\Documents\DS-5 Workspace_GHRD"
Working directory "C:\Users\kigtan\Documents\DS-5 Workspace_GHRD"
reset reset.system
Target has been reset
Execution stopped due to a breakpoint or watchpoint: S:0x00000000
S:0x00000000   LDR      pc,[pc,#24] ; [0x20] = 0xA8
loadfile "D:\SR\HPS_DDR3\software\spl_bsp\uboot-socfpga\spl\u-boot-spl"
Loaded section .text: S:0xFFFF0000 ~ S:0xFFFF8807 (size 0x8808)
Loaded section .rodata: S:0xFFFF8808 ~ S:0xFFFFA501 (size 0x1CFA)
Loaded section .data: S:0xFFFFA508 ~ S:0xFFFFB4EF (size 0xFE8)
Loaded section .bss: S:0xFFFFB4F0 ~ S:0xFFFFB593 (size 0xA4)
Loaded section .malloc: S:0xFFFFB594 ~ S:0xFFFFC993 (size 0x1400)
Loaded section .stack: S:0xFFFFC994 ~ S:0xFFFFD997 (size 0x1004)
Loaded section .spl_irq_stack: S:0xFFFFD998 ~ S:0xFFFFE99F (size 0x1008)
Entry point S:0xFFFF0000
set semihosting enabled true
Semihosting server socket created at port 8000
```

Command: set semihosting enabled true

Enabling Simple Memory Test

After the SDRAM is successfully calibrated, a simple memory test may be performed using the debugger.

1. When you create the **.bsp** file in the BSP Editor, select **HARDWARE_DIAGNOSTIC** in the **spl.debug** window..



SOPC Information file:
CPU name:
Operating system: Preloader Version: default
BSP target directory: ..\..\..\..\..\C:\altera\13.0sp1\embedded

spl.debug

DEBUG_MEMORY_ADDR: 0xffffd00

DEBUG_MEMORY_SIZE: 0x200

□ DEBUG_MEMORY_WRITE

☑ HARDWARE_DIAGNOSTIC

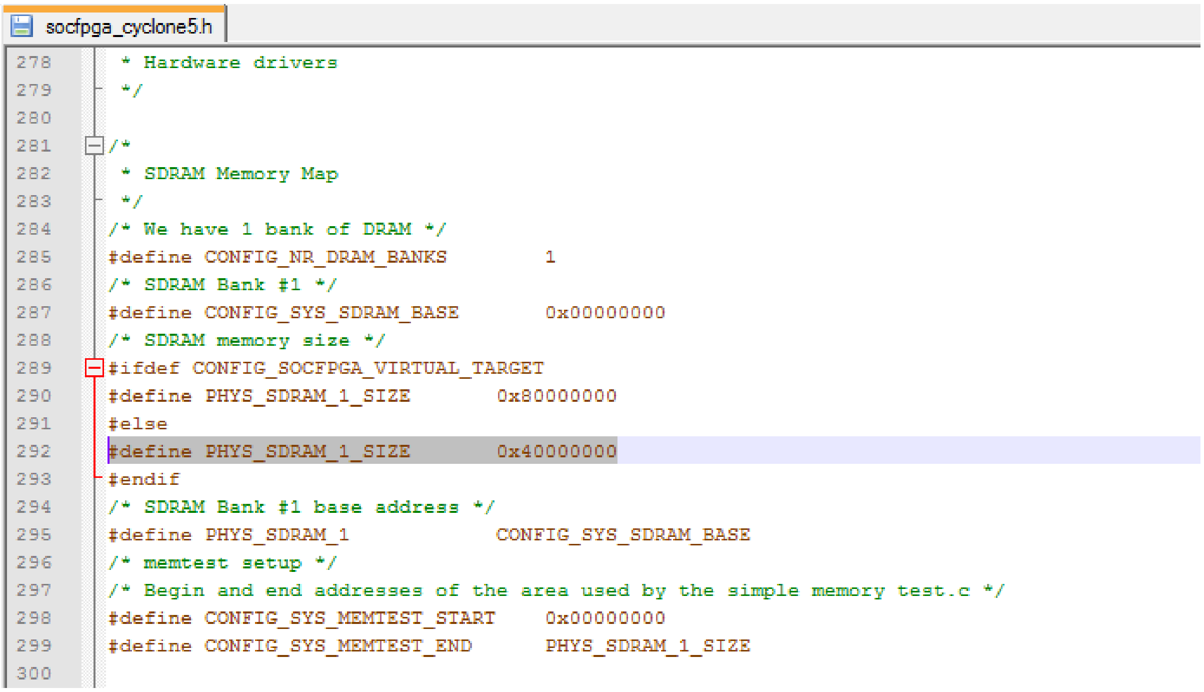
☑ SEMIHOSTING

□ SKIP_SDRAM

```
Setting "spl.reset_assert.SPTIMER1" set to "0".
Setting "spl.reset_assert.UART0" set to "0".
Setting "spl.reset_assert.UART1" set to "1".
Setting "spl.reset_assert.USB0" set to "1".
Setting "spl.reset_assert.USB1" set to "0".
Setting "spl.warm_reset_handshake.ETR" set to "1".
Setting "spl.warm_reset_handshake.FPGA" set to "1".
```

2. The simple memory test assumes SDRAM with a memory size of 1 GB. If your board contains a different SDRAM memory size, open the file **<*design folder*>\spl_bsp\uboot-socfpga\include\configs\**

socfpga_cyclone5.h in a text editor, and change the `PHYS_SDRAM_1_SIZE` parameter at line 292 to specify your actual memory size in bytes.



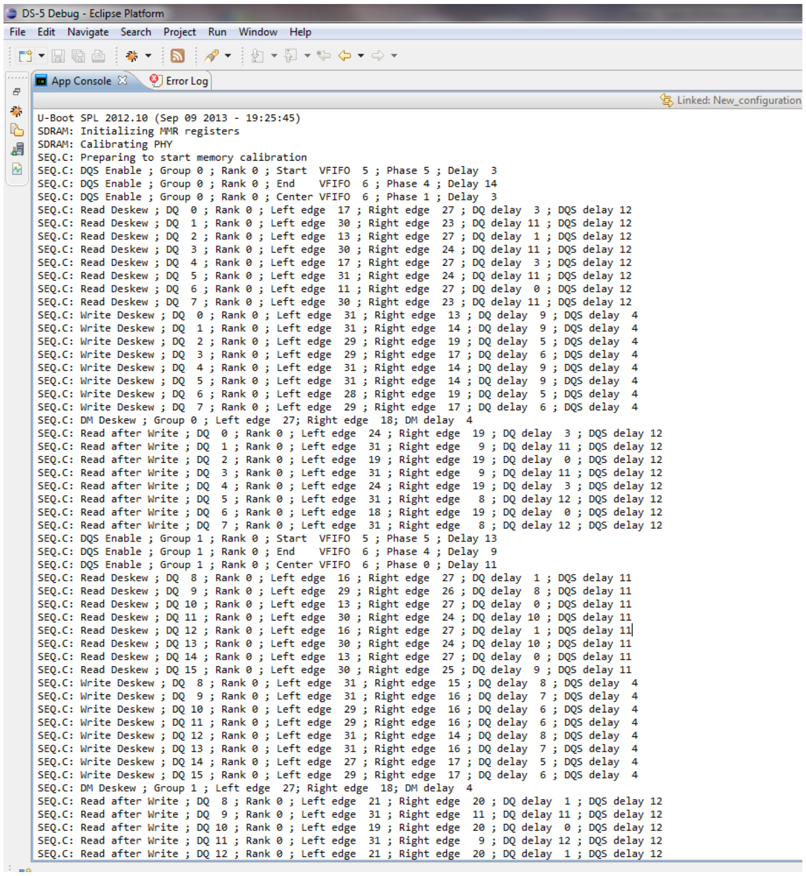
```
278  * Hardware drivers
279  */
280
281 /*
282  * SDRAM Memory Map
283  */
284 /* We have 1 bank of DRAM */
285 #define CONFIG_NR_DRAM_BANKS        1
286 /* SDRAM Bank #1 */
287 #define CONFIG_SYS_SDRAM_BASE       0x00000000
288 /* SDRAM memory size */
289 #ifdef CONFIG_SOCFPGA_VIRTUAL_TARGET
290 #define PHYS_SDRAM_1_SIZE       0x80000000
291 #else
292 #define PHYS_SDRAM_1_SIZE       0x40000000
293 #endif
294 /* SDRAM Bank #1 base address */
295 #define PHYS_SDRAM_1            CONFIG_SYS_SDRAM_BASE
296 /* memtest setup */
297 /* Begin and end addresses of the area used by the simple memory test.c */
298 #define CONFIG_SYS_MEMTEST_START    0x00000000
299 #define CONFIG_SYS_MEMTEST_END      PHYS_SDRAM_1_SIZE
300
```

Enabling the Debug Report

You can enable the SDRAM calibration sequencer to produce a debug report on the UART printout or semihosting output. To enable the debug report, follow these steps:

1. After you have enabled the UART or semihosting, open the file **<*project directory*>\hps_isw_handoff\sequencer_defines.h**in a text editor.
2. Locate the line `#define RUNTIME_CAL_REPORT 0` and change it to `#define RUNTIME_CAL_REPORT 1`.

Figure 11-8: Semihosting Printout With Debug Support Enabled


DS-5 Debug - Eclipse Platform
File Edit Navigate Search Project Run Window Help
App Console
Error Log
Linked: New_configuration

```
U-Boot SPL 2012.10 (Sep 09 2013 - 19:25:45)
SDRAM: Initializing MMR registers
SDRAM: Calibrating PHY
SEQ.C: Preparing to start memory calibration
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Start  VFIFO  5 ; Phase 5 ; Delay  3
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; End    VFIFO  6 ; Phase 4 ; Delay 14
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Center VFIFO  6 ; Phase 1 ; Delay  3
SEQ.C: Read Deskew ; DQ  0 ; Rank 0 ; Left edge  17 ; Right edge  27 ; DQ delay  3 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  1 ; Rank 0 ; Left edge  30 ; Right edge  23 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  2 ; Rank 0 ; Left edge  13 ; Right edge  27 ; DQ delay  1 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  3 ; Rank 0 ; Left edge  30 ; Right edge  24 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  4 ; Rank 0 ; Left edge  17 ; Right edge  27 ; DQ delay  3 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  5 ; Rank 0 ; Left edge  31 ; Right edge  24 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  6 ; Rank 0 ; Left edge  11 ; Right edge  27 ; DQ delay  0 ; DQS delay 12
SEQ.C: Read Deskew ; DQ  7 ; Rank 0 ; Left edge  30 ; Right edge  23 ; DQ delay 11 ; DQS delay 12
SEQ.C: Write Deskew ; DQ  0 ; Rank 0 ; Left edge  31 ; Right edge  13 ; DQ delay  9 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  1 ; Rank 0 ; Left edge  31 ; Right edge  14 ; DQ delay  9 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  2 ; Rank 0 ; Left edge  29 ; Right edge  19 ; DQ delay  5 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  3 ; Rank 0 ; Left edge  29 ; Right edge  17 ; DQ delay  6 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  4 ; Rank 0 ; Left edge  31 ; Right edge  14 ; DQ delay  9 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  5 ; Rank 0 ; Left edge  31 ; Right edge  14 ; DQ delay  9 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  6 ; Rank 0 ; Left edge  28 ; Right edge  19 ; DQ delay  5 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  7 ; Rank 0 ; Left edge  29 ; Right edge  17 ; DQ delay  6 ; DQS delay  4
SEQ.C: DM Deskew ; Group 0 ; Left edge  27; Right edge  18; DM delay  4
SEQ.C: Read after Write ; DQ  0 ; Rank 0 ; Left edge  24 ; Right edge  19 ; DQ delay  3 ; DQS delay 12
SEQ.C: Read after Write ; DQ  1 ; Rank 0 ; Left edge  31 ; Right edge   9 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ  2 ; Rank 0 ; Left edge  19 ; Right edge  19 ; DQ delay  0 ; DQS delay 12
SEQ.C: Read after Write ; DQ  3 ; Rank 0 ; Left edge  31 ; Right edge   9 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ  4 ; Rank 0 ; Left edge  24 ; Right edge  19 ; DQ delay  3 ; DQS delay 12
SEQ.C: Read after Write ; DQ  5 ; Rank 0 ; Left edge  31 ; Right edge   8 ; DQ delay 12 ; DQS delay 12
SEQ.C: Read after Write ; DQ  6 ; Rank 0 ; Left edge  18 ; Right edge  19 ; DQ delay  0 ; DQS delay 12
SEQ.C: Read after Write ; DQ  7 ; Rank 0 ; Left edge  31 ; Right edge   8 ; DQ delay 12 ; DQS delay 12
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; Start  VFIFO  5 ; Phase 5 ; Delay 13
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; End    VFIFO  6 ; Phase 4 ; Delay  9
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; Center VFIFO  6 ; Phase 0 ; Delay 11
SEQ.C: Read Deskew ; DQ  8 ; Rank 0 ; Left edge  16 ; Right edge  27 ; DQ delay  1 ; DQS delay 11
SEQ.C: Read Deskew ; DQ  9 ; Rank 0 ; Left edge  29 ; Right edge  26 ; DQ delay  8 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 10 ; Rank 0 ; Left edge  13 ; Right edge  27 ; DQ delay  0 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 11 ; Rank 0 ; Left edge  30 ; Right edge  24 ; DQ delay 10 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 12 ; Rank 0 ; Left edge  16 ; Right edge  27 ; DQ delay  1 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 13 ; Rank 0 ; Left edge  30 ; Right edge  24 ; DQ delay 10 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 14 ; Rank 0 ; Left edge  13 ; Right edge  27 ; DQ delay  0 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 15 ; Rank 0 ; Left edge  30 ; Right edge  25 ; DQ delay  9 ; DQS delay 11
SEQ.C: Write Deskew ; DQ  8 ; Rank 0 ; Left edge  31 ; Right edge  15 ; DQ delay  8 ; DQS delay  4
SEQ.C: Write Deskew ; DQ  9 ; Rank 0 ; Left edge  31 ; Right edge  16 ; DQ delay  7 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 10 ; Rank 0 ; Left edge  29 ; Right edge  16 ; DQ delay  6 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 11 ; Rank 0 ; Left edge  29 ; Right edge  16 ; DQ delay  6 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 12 ; Rank 0 ; Left edge  31 ; Right edge  14 ; DQ delay  8 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 13 ; Rank 0 ; Left edge  31 ; Right edge  16 ; DQ delay  7 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 14 ; Rank 0 ; Left edge  27 ; Right edge  17 ; DQ delay  5 ; DQS delay  4
SEQ.C: Write Deskew ; DQ 15 ; Rank 0 ; Left edge  29 ; Right edge  17 ; DQ delay  6 ; DQS delay  4
SEQ.C: DM Deskew ; Group 1 ; Left edge  27; Right edge  18; DM delay  4
SEQ.C: Read after Write ; DQ  8 ; Rank 0 ; Left edge  21 ; Right edge  20 ; DQ delay  1 ; DQS delay 12
SEQ.C: Read after Write ; DQ  9 ; Rank 0 ; Left edge  31 ; Right edge  11 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ 10 ; Rank 0 ; Left edge  19 ; Right edge  20 ; DQ delay  0 ; DQS delay 12
SEQ.C: Read after Write ; DQ 11 ; Rank 0 ; Left edge  31 ; Right edge   9 ; DQ delay 12 ; DQS delay 12
SEQ.C: Read after Write ; DQ 12 ; Rank 0 ; Left edge  21 ; Right edge  20 ; DQ delay  1 ; DQS delay 12
```

Analysis of Debug Report

The following analysis will help you interpret the debug report.

- The Read Deskew and Write Deskew results shown in the debug report are before calibration. (Before calibration results are actually from the window seen *during* calibration, and are most useful for debugging.)
- For each DQ group, the Write Deskew, Read Deskew, DM Deskew, and Read after Write results map to the before-calibration margins reported in the EMIF Debug Toolkit.

  Note: The Write Deskew, Read Deskew, DM Deskew, and Read after Write results are reported in delay steps (nominally 25ps, in Arria V and Cyclone V devices), not in picoseconds.
- DQS Enable calibration is reported as a VFIFO setting (in one clock period steps), a phase tap (in one-eighth clock period steps), and a delay chain step (in 25ps steps).

```
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Start  VFIFO  5 ; Phase 6 ; Delay  4
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; End    VFIFO  6 ; Phase 5 ; Delay  9
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Center VFIFO  6 ; Phase 2 ; Delay  1
```

  Analysis of DQS Enable results: A VFIFO tap is 1 clock period, a phase is 1/8 clock period (45 degrees) and delay is nominally 25ps per tap. The DQSen window is the difference between the start and end—for the above example, assuming a frequency of 400 MHz (2500ps), that calculates as follows: `start is 5*2500 + 6*2500/8 +4*25 = 14475ps`. By the same calculation, the end is 16788ps. Consequently, the DQSen window is 2313ps.
- The size of a read window or write window is equal to `(left edge + right edge) * delay chain step size`. Both the left edge and the right edge can be negative or positive.:

```
SEQ.C: Read Deskew  ; DQ  0 ; Rank 0 ; Left edge  18 ; Right edge  27 ; DQ
delay  0 ; DQS delay  8
SEQ.C: Write Deskew ; DQ  0 ; Rank 0 ; Left edge  30 ; Right edge  17 ; DQ
delay  6 ; DQS delay  4
```

  Analysis of DQ and DQS delay results: The DQ and DQS output delay (write) is the D5 delay chain. The DQ input delay (read) is the D1 delay chain, the DQS input delay (read) is the D4 delay chain.
- Consider the following example of latency results:

```
SEQ.C: LFIFO Calibration ; Latency 10
```

  Analysis of latency results: This is the calibrated PHY read latency. The EMIF Debug Toolkit does not report this figure. This latency is reported in clock cycles.
- Consider the following example of FOM results:

```
SEQ.C: FOM IN  = 83
SEQ.C: FOM OUT = 91
```

  Analysis of FOM results: The FOM IN value is a measure of the health of the read interface; it is calculated as the sum over all groups of the minimum margin on DQ plus the margin on DQS, divided by 2. The FOM OUT is a measure of the health of the write interface; it is calculated as the sum over all groups of the minimum margin on DQ plus the margin on DQS, divided by 2. You may refer to these values as indicators of improvement when you are experimenting with various termination schemes, assuming there are no individual misbehaving DQ pins.
- The debug report does not provide delay chain step size values. The delay chain step size varies with device speed grade. Refer to your device data sheet for exact incremental delay values for delay chains.

Related Information

Functional Description–UniPHY

For more information about calibration, refer to the *Calibration Stages* section in the *Functional Description-UniPHY* chapter of the *External Memory Interface Handbook*.

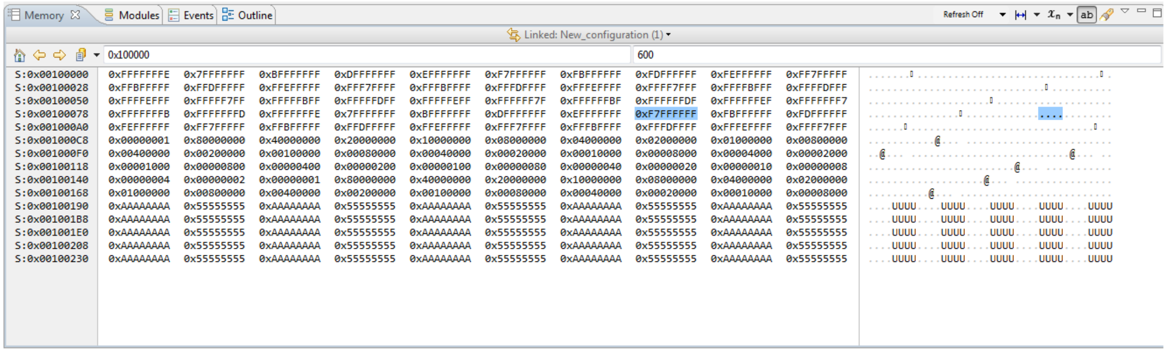
Writing a Predefined Data Pattern to SDRAM in the Preloader

You can include your own code to write a predefined data pattern to the SDRAM in the preloader for debugging purposes.

1. Include your code in the file: **<*project_folder*>\software\spl_bsp\uboot-socfpga\arch\arm\cpu\armv7\socfpga\spl.c** .
   Adding the following code to the **spl.c** file causes the controller to write walking 1s and walking 0s, repeated five times, to the SDRAM.

```
/*added for demo, place after the last #define statement in spl.c */
#define ROTATE_RIGHT(X) ( (X>>1) | (X&1?0X80000000:0) )
/*added for demo, place after the calibration code */
 test_data_walk0((long *)0x100000,PHYS_SDRAM_1_SIZE);
int test_data_walk0(long *base, long maxsize)
{
    volatile long *addr;
    long           cnt;
    ulong           data_temp[3];
        ulong           expected_data[3];
    ulong           read_data;
        int            i = 0;  //counter to loop different data pattern
        int           num_address;

       num_address=50;

       data_temp[0]=0XFFFFFFFE; //initial data for walking 0 pattern
       data_temp[1]=0X00000001; //initial data for walking 1 pattern
       data_temp[2]=0XAAAAAAAA; //initial data for A->5 switching

       expected_data[0]=0XFFFFFFFE; //initial data for walking 0 pattern
       expected_data[1]=0X00000001; //initial data for walking 1 pattern
       expected_data[2]=0XAAAAAAAA; //initial data for A->5 switching

       for (i=0;i<3;i++) {

    printf("\nSTARTED %08X DATA PATTERN !!!!\n",data_temp[i]);
    /*write*/
    for (cnt = (0+i*num_address); cnt < ((i+1)*num_address) ;  cnt++ ) {
        addr = base + cnt;    /* pointer arith! */
        sync ();
            *addr = data_temp[i];
        data_temp[i]=ROTATE_RIGHT(data_temp[i]);

        }

    /*read*/
    for (cnt = (0+i*num_address); cnt < ((i+1)*num_address) ; cnt = cnt++ ) {
        addr = base + cnt;    /* pointer arith! */
        sync ();
        read_data=*addr;
              printf("Address:%X  Expected: %08X    Read:%08X \n",addr,
expected_data[i],read_data);
               if (expected_data[i] !=read_data) {
                puts("!!!!!!FAILED!!!!!!\n\n");
        hang();
            }
                expected_data[i]=ROTATE_RIGHT(expected_data[i]);
}
}
}
====//End Of Code//=====
```

Figure 11-9: Memory Contents After Executing Example Code



```
0x100000                                                                                                        600
S:0x00100000  0xFFFFFFFE 0x7FFFFFFF 0xBFFFFFFF 0xDFFFFFFF 0xEFFFFFFF 0xF7FFFFFF 0xFBFFFFFF 0xFDFFFFFF 0xFEFFFFFF 0xFF7FFFFF  .......▯..............................▯.
S:0x00100028  0xFFBFFFFF 0xFFDFFFFF 0xFFFEFFFF 0xFFF7FFFF 0xFFFBFFFF 0xFFFDFFFF 0xFFFEFFFF 0xFFFF7FFF 0xFFFFBFFF 0xFFFFDFFF  .............................▯..........
S:0x00100050  0xFFFFEFFF 0xFFFFF7FF 0xFFFFFBFF 0xFFFFFDFF 0xFFFFFEFF 0xFFFFFF7F 0xFFFFFFBF 0xFFFFFFDF 0xFFFFFFEF 0xFFFFFFF7  ....................▯...................
S:0x00100078  0xFFFFFFFB 0xFFFFFFFD 0xFFFFFFFE 0x7FFFFFFF 0xBFFFFFFF 0xDFFFFFFF 0xEFFFFFFF 0xF7FFFFFF 0xFBFFFFFF 0xFDFFFFFF  ...............▯........................
S:0x001000A0  0xFEFFFFFF 0xFF7FFFFF 0xFFBFFFFF 0xFFDFFFFF 0xFFEFFFFF 0xFFF7FFFF 0xFFFBFFFF 0xFFFDFFFF 0xFFFEFFFF 0xFFFF7FFF  ......▯..............................▯..
S:0x001000C8  0x00000001 0x80000000 0x40000000 0x20000000 0x10000000 0x08000000 0x04000000 0x02000000 0x01000000 0x00800000  ...........@............................
S:0x001000F0  0x00400000 0x00200000 0x00100000 0x00080000 0x00040000 0x00020000 0x00010000 0x00008000 0x00004000 0x00002000  ..@..............................@......
S:0x00100118  0x00001000 0x00000800 0x00000400 0x00000200 0x00000100 0x00000080 0x00000040 0x00000020 0x00000010 0x00000008  ........................@...............
S:0x00100140  0x00000004 0x00000002 0x00000001 0x80000000 0x40000000 0x20000000 0x10000000 0x08000000 0x04000000 0x02000000  ...................@....................
S:0x00100168  0x01000000 0x00800000 0x00400000 0x00200000 0x00100000 0x00080000 0x00040000 0x00020000 0x00010000 0x00008000  ..........@.............................
S:0x00100190  0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555  ....UUUU....UUUU....UUUU....UUUU....UUUU
S:0x001001B8  0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555  ....UUUU....UUUU....UUUU....UUUU....UUUU
S:0x001001E0  0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555  ....UUUU....UUUU....UUUU....UUUU....UUUU
S:0x00100208  0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555  ....UUUU....UUUU....UUUU....UUUU....UUUU
S:0x00100230  0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555 0xAAAAAAAA 0x55555555  ....UUUU....UUUU....UUUU....UUUU....UUUU
```

SDRAM Controller Address Map and Register Definitions

This section lists the SDRAM register address map and describes the registers.

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  Base addresses of all HPS modules
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

SDRAM Controller Address Map

Address map for the SDRAM Interface registers

Base Address: `0xFFC20000`

SDRAM Controller Module

Register	Offset	Width	Access	Reset Value	Description
`ctrlcfg` on page 11-42	0x5000	32	RW	0x0	Controller Configuration Register
`dramtiming1` on page 11-44	0x5004	32	RW	0x0	DRAM Timings 1 Register
`dramtiming2` on page 11-45	0x5008	32	RW	0x0	DRAM Timings 2 Register
`dramtiming3` on page 11-46	0x500C	32	RW	0x0	DRAM Timings 3 Register
`dramtiming4` on page 11-46	0x5010	32	RW	0x0	DRAM Timings 4 Register
`lowpwrtiming` on page 11-47	0x5014	32	RW	0x0	Lower Power Timing Register
`dramodt` on page 11-48	0x5018	32	RW	0x0	ODT Control Register

Register	Offset	Width	Acces s	Reset Value	Description
dramaddrw on page 11-49	0x502C	32	RW	0x0	DRAM Address Widths Register
dramifwidth on page 11-49	0x5030	32	RW	0x0	DRAM Interface Data Width Register
dramsts on page 11-50	0x5038	32	RW	0x0	DRAM Status Register
dramintr on page 11-51	0x503C	32	RW	0x0	ECC Interrupt Register
sbecount on page 11-51	0x5040	32	RW	0x0	ECC Single Bit Error Count Register
dbecount on page 11-52	0x5044	32	RW	0x0	ECC Double Bit Error Count Register
erraddr on page 11-52	0x5048	32	RW	0x0	ECC Error Address Register
dropcount on page 11-53	0x504C	32	RW	0x0	ECC Auto-correction Dropped Count Register
dropaddr on page 11-53	0x5050	32	RW	0x0	ECC Auto-correction Dropped Address Register
lowpwreq on page 11-54	0x5054	32	RW	0x0	Low Power Control Register
lowpwrack on page 11-55	0x5058	32	RW	0x0	Low Power Acknowledge Register
staticcfg on page 11-55	0x505C	32	RW	0x0	Static Configuration Register
ctrlwidth on page 11-56	0x5060	32	RW	0x0	Memory Controller Width Register
portcfg on page 11-57	0x507C	32	RW	0x0	Port Configuration Register
fpgaportrst on page 11-58	0x5080	32	RW	0x0	FPGA Ports Reset Control Register
protportdefault on page 11-59	0x508C	32	RW	0x0	Memory Protection Port Default Register
protruleaddr on page 11-59	0x5090	32	RW	0x0	Memory Protection Address Register
protruleid on page 11-60	0x5094	32	RW	0x0	Memory Protection ID Register
protruledata on page 11-61	0x5098	32	RW	0x0	Memory Protection Rule Data Register
protrulerdwr on page 11-62	0x509C	32	RW	0x0	Memory Protection Rule Read-Write Register
mppriority on page 11-63	0x50AC	32	RW	0x0	Scheduler priority Register
remappriority on page 11-63	0x50E0	32	RW	0x0	Controller Command Pool Priority Remap Register

Port Sum of Weight Register

Register	Offset	Width	Acces s	Reset Value	Description
mpweight_0_4 on page 11-64	0x50B0	32	RW	0x0	Port Sum of Weight Register[1/4]
mpweight_1_4 on page 11-65	0x50B4	32	RW	0x0	Port Sum of Weight Register[2/4]
mpweight_2_4 on page 11-65	0x50B8	32	RW	0x0	Port Sum of Weight Register[3/4]
mpweight_3_4 on page 11-66	0x50BC	32	RW	0x0	Port Sum of Weight Register[4/4]

SDRAM Controller Module Register Descriptions

Address map for the SDRAM controller and multi-port front-end. All registers in this group reset to zero.

Offset: 0x5000

ctrlcfg on page 11-42
The Controller Configuration Register determines the behavior of the controller.

dramtiming1 on page 11-44
This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

dramtiming2 on page 11-45
This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

dramtiming3 on page 11-46
This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

dramtiming4 on page 11-46
This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

lowpwrtiming on page 11-47
This register controls the behavior of the low power logic in the controller.

dramodt on page 11-48
This register controls which ODT pin asserts with chip select 0 (CS0) assertion and which ODT pin asserts with chip select 1 (CS1) assertion.

dramaddrw on page 11-49
This register configures the width of the various address fields of the DRAM. The values specified in this register must match the memory devices being used.

dramifwidth on page 11-49
This register controls the interface width of the SDRAM controller.

dramsts on page 11-50
This register provides the status of the calibration and ECC logic.

dramintr on page 11-51
This register can enable, disable and clear the SDRAM error interrupts.

sbecount on page 11-51
This register tracks the single-bit error count.

dbecount on page 11-52
This register tracks the double-bit error count.

erraddr on page 11-52
This register holds the address of the most recent ECC error.

dropcount on page 11-53
This register holds the address of the most recent ECC error.

dropaddr on page 11-53
This register holds the last dropped address.

lowpwreq on page 11-54
This register instructs the controller to put the DRAM into a power down state. Note that some commands are only valid for certain memory types.

lowpwrack on page 11-55
This register gives the status of the power down commands requested by the Low Power Control register.

staticcfg on page 11-55
This register controls configuration values which cannot be updated during active transfers. First configure the `membl` and `eccn` fields and then re-write these fields while setting the `applycfg` bit. The `applycfg` bit is write only.

ctrlwidth on page 11-56
This register controls the width of the physical DRAM interface.

portcfg on page 11-57
Each bit of the `autopchen` field maps to one of the control ports. If a port executes mostly sequential memory accesses, the corresponding `autopchen` bit should be 0. If the port has highly random accesses, then its `autopchen` bit should be set to 1.

fpgaportrst on page 11-58
This register implements functionality to allow the CPU to control when the MPFE will enable the ports to the FPGA fabric.

protportdefault on page 11-59
This register controls the default protection assignment for a port. Ports which have explicit rules which define regions which are illegal to access should set the bits to pass by default. Ports which have explicit rules which define legal areas should set the bit to force all transactions to fail. Leaving this register to all zeros should be used for systems which do not desire any protection from the memory controller.

protruleaddr on page 11-59
This register is used to control the memory protection for port 0 transactions. Address ranges can either be used to allow access to memory regions or disallow access to memory regions. If TrustZone is being used, access can be enabled for protected transactions or disabled for unprotected transactions. The default state of this register is to allow all access. Address values used for protection are only physical addresses.

protruleid on page 11-60
This register configures the AxID for a given protection rule.

protruledata on page 11-61
This register configures the protection memory characteristics of each protection rule.

protrulerdwr on page 11-62
This register is used to perform read and write operations to the internal protection table.

mppriority on page 11-63
This register is used to configure the DRAM burst operation scheduling.

remappriority on page 11-63
This register applies another level of port priority after a transaction is placed in the single port queue.

Port Sum of Weight Register Register Descriptions on page 11-64
This register is used to configure the DRAM burst operation scheduling.

ctrlcfg

The Controller Configuration Register determines the behavior of the controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25000

Offset: 0x5000

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						bursttermen RW 0x0	burstintren RW 0x0	nodmpins RW 0x0	dqstrken RW 0x0	starvelimit RW 0x0					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reorderen RW 0x0	gendbe RW 0x0	gensbe RW 0x0	cfg_enable_ecc_code_overwrites RW 0x0	ecccorren RW 0x0	eccen RW 0x0	addrorder RW 0x0		membl RW 0x0					memtype RW 0x0		

ctrlcfg Fields

Bit	Name	Description	Access	Reset
25	bursttermen	Set to a one to enable the controller to issue burst terminate commands. This must only be set when the DRAM memory type is LPDDR2.	RW	0x0
24	burstintren	Set to a one to enable the controller to issue burst interrupt commands. This must only be set when the DRAM memory type is LPDDR2.	RW	0x0
23	nodmpins	Set to a one to enable DRAM operation if no DM pins are connected.	RW	0x0
22	dqstrken	Enables DQS tracking in the PHY.	RW	0x0
21:16	starvelimit	Specifies the number of DRAM burst transactions an individual transaction will allow to reorder ahead of it before its priority is raised in the memory controller.	RW	0x0
15	reorderen	This bit controls whether the controller can re-order operations to optimize SDRAM bandwidth. It should generally be set to a one.	RW	0x0
14	gendbe	Enable the deliberate insertion of double bit errors in data written to memory. This should only be used for testing purposes.	RW	0x0
13	gensbe	Enable the deliberate insertion of single bit errors in data written to memory. This should only be used for testing purposes.	RW	0x0
12	cfg_enable_ecc_code_ overwrites	Set to a one to enable ECC overwrites. ECC overwrites occur when a correctable ECC error is seen and cause a new read/modify/write to be scheduled for that location to clear the ECC error.	RW	0x0
11	ecccorren	Enable auto correction of the read data returned when single bit error is detected.	RW	0x0
10	eccen	Enable the generation and checking of ECC. This bit must only be set if the memory connected to the SDRAM interface is 24 or 40 bits wide. If you set this, you must clear the useeccasdata field in the staticcfg register.	RW	0x0

Bit	Name	Description	Access	Reset
9:8	addrorder	This bit field selects the order for address interleaving. Programming this field with different values gives different mappings between the AXI or Avalon-MM address and the SDRAM address. Program this field with the following binary values to select the ordering. **Value** **Description** 0x0 chip, row, bank, column 0x1 chip, bank, row, column 0x2 row, chip, bank, column 0x3 reserved	RW	0x0
7:3	membl	This bit field configures burst length as a static decimal RW 0x0 value. These values are valid for JEDEC allowed DRAMs configured by programming the `memtype` field. The membl field is programmed as follows: For DDR3, program `membl` to 0x8; for DDR2, `membl` can be 0x4 or 0x8, depending on the DRAM chip; for LPDDR2, `membl` can be programmed with 0x4, 0x8 or 0x10; for LPDDR, `membl` can be 0x2, 0x4 or 0x8.	RW	0x0
2:0	memtype	This bit field selects the memory type. This field can be programmed with the following binary values: **Value** **Description** 0x0 Memory type is DDR2 SDRAM 0x1 Memory type is DDR3 SDRAM 0x2 Memory type is LPDDR1 SDRAM 0x3 Memory type is LPDR2 SDRAM 0x4-0x7 These bits are reserved.	RW	0x0

dramtiming1

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25004

Offset: `0x5004`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
trfc RW 0x0								tfaw RW 0x0						trrd RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trrd RW 0x0		tcl RW 0x0					tal RW 0x0					tcwl RW 0x0			

dramtiming1 Fields

Bit	Name	Description	Access	Reset
31:24	trfc	The refresh cycle timing parameter.	RW	0x0
23:18	tfaw	The four-activate window timing parameter.	RW	0x0
17:14	trrd	The activate to activate, different banks timing parameter.	RW	0x0
13:9	tcl	Memory read latency.	RW	0x0
8:4	tal	Memory additive latency.	RW	0x0
3:0	tcwl	Memory write latency.	RW	0x0

dramtiming2

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25008

Offset: 0x5008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			twtr RW 0x0				twr RW 0x0				trp RW 0x0				trcd RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trcd RW 0x0			trefi RW 0x0												

dramtiming2 Fields

Bit	Name	Description	Access	Reset
28:25	twtr	The write to read timing parameter.	RW	0x0

Bit	Name	Description	Access	Reset
24:21	`twr`	The write recovery timing.	`RW`	`0x0`
20:17	`trp`	The precharge to activate timing parameter.	`RW`	`0x0`
16:13	`trcd`	The activate to read/write timing parameter.	`RW`	`0x0`
12:0	`trefi`	The refresh interval timing parameter.	`RW`	`0x0`

dramtiming3

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
`sdr`	`0xFFC20000`	`0xFFC2500C`

Offset: `0x500C`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`									`tccd` `RW 0x0`				`tmrd` `RW 0x0`		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
`tmrd` `RW 0x0`	`trc` `RW 0x0`						`tras` `RW 0x0`					`trtp` `RW 0x0`			

dramtiming3 Fields

Bit	Name	Description	Access	Reset
22:19	`tccd`	The CAS to CAS delay time.	`RW`	`0x0`
18:15	`tmrd`	Mode register timing parameter.	`RW`	`0x0`
14:9	`trc`	The activate to activate timing parameter.	`RW`	`0x0`
8:4	`tras`	The activate to precharge timing parameter.	`RW`	`0x0`
3:0	`trtp`	The read to precharge timing parameter.	`RW`	`0x0`

dramtiming4

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
`sdr`	`0xFFC20000`	`0xFFC25010`

Offset: 0x5010

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">minpwrsavecycles
RW 0x0</td><td colspan="4">pwrdownexit
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">pwrdownexit
RW 0x0</td><td colspan="10">selfrfshexit
RW 0x0</td></tr>
</table>

dramtiming4 Fields

Bit	Name	Description	Access	Reset
23:20	minpwrsavecycles	The minimum number of cycles to stay in a low power state. This applies to both power down and self-refresh and should be set to the greater of tPD and tCKESR.	RW	0x0
19:10	pwrdownexit	The power down exit cycles, tXPDLL.	RW	0x0
9:0	selfrfshexit	The self refresh exit cycles, tXS.	RW	0x0

lowpwrtiming

This register controls the behavior of the low power logic in the controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25014

Offset: 0x5014

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="12">Reserved</td><td colspan="4">clkdisablecycles
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">autopdcycles
RW 0x0</td></tr>
</table>

lowpwrtiming Fields

Bit	Name	Description	Access	Reset
19:16	clkdisablecycles	Set to a the number of clocks after the execution of an self-refresh to stop the clock. This register is generally set based on PHY design latency and should generally not be changed.	RW	0x0

Bit	Name	Description	Access	Reset
15:0	autopdcycles	The number of idle clock cycles after which the controller should place the memory into power-down mode.	RW	0x0

dramodt

This register controls which ODT pin asserts with chip select 0 (CS0) assertion and which ODT pin asserts with chip select 1 (CS1) assertion.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25018

Offset: 0x5018

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">cfg_read_odt_chip
RW 0x0</td><td colspan="4">cfg_write_odt_chip
RW 0x0</td></tr>
</table>

dramodt Fields

Bit	Name	Description	Access	Reset
7:4	cfg_read_odt_chip	This register controls which ODT pin is asserted during reads. Bits[5:4] select the ODT pin that asserts with CS0 and bits[7:6] select the ODT pin that asserts with CS1. For example, a value of 0x9 asserts ODT[0] for accesses CS0 and ODT[1] for accesses with CS1. This field can be set to 0x1 is there is only one chip select available.	RW	0x0
3:0	cfg_write_odt_chip	This register controls which ODT pin is asserted during writes. Bits[1:0] select the ODT pin that asserts with CS0 and bits[3:2] select the ODT pin that asserts with CS1. For example, a value of 0x9 asserts ODT[0] for accesses CS0 and ODT[1] for accesses with CS1. This field can be set to 0x1 is there is only one chip select available.	RW	0x0

dramaddrw

This register configures the width of the various address fields of the DRAM. The values specified in this register must match the memory devices being used.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2502C

Offset: 0x502C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
csbits RW 0x0			bankbits RW 0x0			rowbits RW 0x0					colbits RW 0x0				

dramaddrw Fields

Bit	Name	Description	Access	Reset
15:13	csbits	The number of chip select address bits for the memory devices in your memory interface.	RW	0x0
12:10	bankbits	The number of bank address bits for the memory devices in your memory interface.	RW	0x0
9:5	rowbits	The number of row address bits for the memory devices in your memory interface.	RW	0x0
4:0	colbits	The number of column address bits for the memory devices in your memory interface.	RW	0x0

dramifwidth

This register controls the interface width of the SDRAM controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25030

Offset: 0x5030

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ifwidth RW 0x0							

dramifwidth Fields

Bit	Name	Description	Access	Reset
7:0	`ifwidth`	This register controls the width of the SDRAM interface, including any bits used for ECC. For example, for a 32-bit interface with ECC, program this register to 0x28. The `ctrlwidth` register must also be programmed.	RW	0x0

dramsts

This register provides the status of the calibration and ECC logic.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25038

Offset: `0x5038`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											corrdrop RW 0x0	dbeerr RW 0x0	sbeerr RW 0x0	calfail RW 0x0	calsuccess RW 0x0

dramsts Fields

Bit	Name	Description	Access	Reset
4	`corrdrop`	This bit is set to 1 when any auto-corrections have been dropped.	RW	0x0
3	`dbeerr`	This bit is set to 1 when any ECC double bit errors are detected.	RW	0x0
2	`sbeerr`	This bit is set to 1 when any ECC single bit errors are detected.	RW	0x0
1	`calfail`	This bit is set to 1 when the PHY is unable to calibrate.	RW	0x0
0	`calsuccess`	This bit will be set to 1 if the PHY was able to successfully calibrate.	RW	0x0

dramintr

This register can enable, disable and clear the SDRAM error interrupts.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2503C

Offset: 0x503C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											intrclr RW 0x0	corrdropmask RW 0x0	dbemask RW 0x0	sbemask RW 0x0	intren RW 0x0

dramintr Fields

Bit	Name	Description	Access	Reset
4	intrclr	Writing to this self-clearing bit clears the interrupt signal. Writing to this bit also clears the error count and error address registers: sbecount, dbecount, dropcount, erraddr, and dropaddr.	RW	0x0
3	corrdropmask	Set this bit to a one to mask interrupts for an ECC correction write back needing to be dropped. This indicates a burst of memory errors in a short period of time.	RW	0x0
2	dbemask	Mask the double bit error interrupt.	RW	0x0
1	sbemask	Mask the single bit error interrupt.	RW	0x0
0	intren	Enable the interrupt output.	RW	0x0

sbecount

This register tracks the single-bit error count.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25040

Offset: 0x5040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								count RW 0x0							

sbecount Fields

Bit	Name	Description	Access	Reset
7:0	count	Reports the number of single bit errors that have occurred since the status register counters were last cleared.	RW	0x0

dbecount

This register tracks the double-bit error count.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25044

Offset: 0x5044

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								count RW 0x0							

dbecount Fields

Bit	Name	Description	Access	Reset
7:0	count	Reports the number of double bit errors that have occurred since the status register counters were last cleared.	RW	0x0

erraddr

This register holds the address of the most recent ECC error.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25048

Offset: 0x5048

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr RW 0x0															

erraddr Fields

Bit	Name	Description	Access	Reset
31:0	addr	The address of the most recent ECC error.	RW	0x0

dropcount

This register holds the address of the most recent ECC error.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2504C

Offset: 0x504C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								corrdropcount RW 0x0							

dropcount Fields

Bit	Name	Description	Access	Reset
7:0	corrdropcount	This gives the count of the number of ECC write back transactions dropped due to the internal FIFO overflowing.	RW	0x0

dropaddr

This register holds the last dropped address.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25050

Offset: 0x5050

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
corrdropaddr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
corrdropaddr RW 0x0															

dropaddr Fields

Bit	Name	Description	Access	Reset
31:0	corrdropaddr	This register gives the last address which was dropped.	RW	0x0

lowpwreq

This register instructs the controller to put the DRAM into a power down state. Note that some commands are only valid for certain memory types.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25054

Offset: 0x5054

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										selfrfshmask RW 0x0		selfrshreq RW 0x0	deeppwrdnmask RW 0x0		deeppwrdnreq RW 0x0

lowpwreq Fields

Bit	Name	Description	Access	Reset
5:4	selfrfshmask	Write a one to each bit of this field to have a self refresh request apply to both chips.	RW	0x0
3	selfrshreq	Write a one to this bit to request the RAM be put into a self refresh state. This bit is treated as a static value so the RAM will remain in self-refresh as long as this register bit is set to a one. This power down mode can be selected for all DRAMs supported by the controller.	RW	0x0

Bit	Name	Description	Access	Reset
2:1	deeppwrdnmask	Write ones to this register to select which DRAM chip selects will be powered down. Typical usage is to set both of these bits when deeppwrdnreq is set but the controller does support putting a single chip into deep power down and keeping the other chip running.	RW	0x0
0	deeppwrdnreq	Write a one to this bit to request a deep power down. This bit should only be written with LPDDR2 DRAMs, DDR3 DRAMs do not support deep power down.	RW	0x0

lowpwrack

This register gives the status of the power down commands requested by the Low Power Control register.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25058

Offset: 0x5058

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														selfrfshack RW 0x0	deeppwrdnack RW 0x0

lowpwrack Fields

Bit	Name	Description	Access	Reset
1	selfrfshack	This bit is a one to indicate that the controller is in a self-refresh state.	RW	0x0
0	deeppwrdnack	This bit is set to a one after a deep power down has been executed	RW	0x0

staticcfg

This register controls configuration values which cannot be updated during active transfers. First configure the membl and eccn fields and then re-write these fields while setting the applycfg bit. The applycfg bit is write only.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2505C

Offset: 0x505C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												applycfg RW 0x0	useeccasdata RW 0x0	membl RW 0x0	

staticcfg Fields

Bit	Name	Description	Access	Reset
3	applycfg	Write with this bit set to apply all the settings loaded in SDR registers to the memory interface. This bit is write-only and always returns 0 if read.	RW	0x0
2	useeccasdata	This field allows the FPGA ports to directly access the extra data bits that are normally used to hold the ECC code. The interface width must be set to 24 or 40 in the dramifwidth register. If you set this, you must clear the eccen field in the ctrlcfg register.	RW	0x0
1:0	membl	This bit field configures the DRAM burst length. For DDR3, program membl to 0x8; for DDR2, membl can be 0x4 or 0x8, depending on the DRAM chip; for LPDDR2, membl can be programmed with 0x4, 0x8 or 0x10; for LPDDR, membl can be 0x2, 0x4 or 0x8. If this field is programmed, the membl field in the ctrlcfg register must also be programmed.	RW	0x0

ctrlwidth

This register controls the width of the physical DRAM interface.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25060

Offset: 0x5060

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ctrlwidth RW 0x0	

ctrlwidth Fields

Bit	Name	Description	Access	Reset
1:0	ctrlwidth	This field specifies the SDRAM controller interface width: **Value** **Description** 0x0 8-bit interface width 0x1 16-bit (no ECC) or 24-bit (ECC enabled) interface width 0x2 32-bit (no ECC) or 40-bit (ECC enabled) interface width Additionally, you must program the dramifwidth register.	RW	0x0

portcfg

Each bit of the autopchen field maps to one of the control ports. If a port executes mostly sequential memory accesses, the corresponding autopchen bit should be 0. If the port has highly random accesses, then its autopchen bit should be set to 1.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2507C

Offset: 0x507C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												autopchen RW 0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
autopchen RW 0x0						Reserved									

portcfg Fields

Bit	Name	Description	Access	Reset
19:10	`autopchen`	Auto-Precharge Enable: One bit is assigned to each control port. For each bit, the encodings are as follows: **Value** **Description** 0x0 The controller requests an automatic precharge following a bus command completion (close the row automatically) 0x1 The controller attempts to keep a row open. All active ports with random dominated operations should set the `autopchen` bit to 1.	`RW`	`0x0`

fpgaportrst

This register implements functionality to allow the CPU to control when the MPFE will enable the ports to the FPGA fabric.

Module Instance	Base Address	Register Address
`sdr`	`0xFFC20000`	`0xFFC25080`

Offset: `0x5080`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		portrstn RW 0x0													

fpgaportrst Fields

Bit	Name	Description	Access	Reset
13:0	`portrstn`	This register should be written to with a 1 to enable the selected FPGA port to exit reset. Writing a bit to a zero will stretch the port reset until the register is written. Read data ports are connected to bits 3:0, with read data port 0 at bit 0 to read data port 3 at bit 3. Write data ports 0 to 3 are mapped to 4 to 7, with write data port 0 connected to bit 4 to write data port 3 at bit 7. Command ports are connected to bits 8 to 13, with command port 0 at bit 8 to command port 5 at bit 13. Expected usage would be to set all the bits at the same time but setting some bits to a zero and others to a one is supported.	`RW`	`0x0`

protportdefault

This register controls the default protection assignment for a port. Ports which have explicit rules which define regions which are illegal to access should set the bits to pass by default. Ports which have explicit rules which define legal areas should set the bit to force all transactions to fail. Leaving this register to all zeros should be used for systems which do not desire any protection from the memory controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2508C

Offset: 0x508C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						portdefault RW 0x0									

protportdefault Fields

Bit	Name	Description	Access	Reset
9:0	portdefault	Determines the default action for specified transactions. When a bit is zero, the specified access is allowed by default. When a bit is one, the specified access is denied by default. Bit 9 CPU write Bit 8 L3 write Bit 7 CPU read Bit 6 L3 read Bit 5 Access to HPGA-to-SDRAM port 5 Bit 4 Access to HPGA-to-SDRAM port 4 Bit 3 Access to HPGA-to-SDRAM port 3 Bit 2 Access to HPGA-to-SDRAM port 2 Bit 1 Access to HPGA-to-SDRAM port 1 Bit 0 Access to HPGA-to-SDRAM port 0	RW	0x0

protruleaddr

This register is used to control the memory protection for port 0 transactions. Address ranges can either be used to allow access to memory regions or disallow access to memory regions. If TrustZone is being used, access can be enabled for protected transactions or disabled for unprotected transactions. The default state of this register is to allow all access. Address values used for protection are only physical addresses.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25090

Offset: 0x5090

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								highaddr RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
highaddr RW 0x0				lowaddr RW 0x0											

protruleaddr Fields

Bit	Name	Description	Access	Reset
23:12	highaddr	Upper 12 bits of the address for a check. Address is compared to be greater than or equal to the address of a transaction. Note that since AXI transactions cannot cross a 4K byte boundary, the transaction start and transaction end address must also fall within the same 1MByte block pointed to by this address pointer.	RW	0x0
11:0	lowaddr	Lower 12 bits of the address for a check. Address is compared to be less than or equal to the address of a transaction. Note that since AXI transactions cannot cross a 4K byte boundary, the transaction start and transaction end address must also fall within the same 1MByte block pointed to by this address pointer.	RW	0x0

protruleid

This register configures the AxID for a given protection rule.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25094

Offset: 0x5094

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								highid RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
highid RW 0x0				lowid RW 0x0											

protruleid Fields

Bit	Name	Description	Access	Reset
23:12	highid	AxID for the protection rule. Incoming AxID needs to be less than or equal to this value. For all AxIDs from a port, AxID high should be programmed to all ones.	RW	0x0
11:0	lowid	AxID for the protection rule. Incoming AxID needs to be greater than or equal to this value. For all AxIDs from a port, AxID high should be programmed to all ones.	RW	0x0

protruledata

This register configures the protection memory characteristics of each protection rule.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25098

Offset: 0x5098

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ruleresult RW 0x0	portmask RW 0x0										validrule RW 0x0	security RW 0x0	

protruledata Fields

Bit	Name	Description	Access	Reset
13	ruleresult	Set this bit to a one to force a protection failure, zero to allow the access the succeed	RW	0x0
12:3	portmask	Set a bit X in this field to to have the rule apply to port X. Clear a bit X in this field to have the rule not apply to the corresponding port X. Ports 0 through 5 are the FPGA fabric ports. Port 6 is the L3 read port. Port 7 is the CPU read port. Port 8 is the L3 write port. Port 9 is the CPU write port. &	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>2</td><td>validrule</td><td>Set to bit to a one to make a rule valid, set to a zero to invalidate a rule.</td><td>RW</td><td>0x0</td></tr>
<tr><td>1:0</td><td>security</td><td>Valid security field encodings are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Rule applies to secure transactions</td></tr>
<tr><td>0x1</td><td>Rule applies to non-secure transactions</td></tr>
<tr><td>0x2 or 0x3</td><td>Rule applies to secure and non-secure transactions</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

protrulerdwr

This register is used to perform read and write operations to the internal protection table.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2509C

Offset: 0x509C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									readrule RW 0x0	writerule RW 0x0	ruleoffset RW 0x0				

protrulerdwr Fields

Bit	Name	Description	Access	Reset
6	readrule	Write to this bit to have the memory_prot_data register loaded with the value from the internal protection table at offset. Table value will be loaded before a rdy is returned so read data from the register will be correct for any follow-on reads to the memory_prot_data register.	RW	0x0
5	writerule	Write to this bit to have the memory_prot_data register to the table at the offset specified by port_offset. Bit automatically clears after a single cycle and the write operation is complete.	RW	0x0

Bit	Name	Description	Access	Reset
4:0	`ruleoffset`	This field defines which of the 20 rules in the protection table you want to read or write.	`RW`	`0x0`

mppriority

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
`sdr`	`0xFFC20000`	`0xFFC250AC`

Offset: `0x50AC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		userpriority RW 0x0													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
userpriority RW 0x0															

mppriority Fields

Bit	Name	Description	Access	Reset
29:0	`userpriority`	User Priority: This field sets the absolute user priority of each port, which is represented as a 3-bit value. 0x0 is the lowest priority and 0x7 is the highest priority. Port 0 is configured by programming `userpriority[2:0]`, port 1 is configured by programming `userpriority[5:3]`, port 2 is configured by programming `userpriority[8:6]`, and so on.	`RW`	`0x0`

remappriority

This register applies another level of port priority after a transaction is placed in the single port queue.

Module Instance	Base Address	Register Address
`sdr`	`0xFFC20000`	`0xFFC250E0`

Offset: `0x50E0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								priorityremap RW 0x0							

remappriority Fields

Bit	Name	Description	Access	Reset
7:0	`priorityremap`	Each bit of this field represents a priority level. If bit N in the `priorityremap` field is set, then any port transaction with absolute user priority of N jumps to the front of the single port queue and is serviced ahead of any tranactions in the queue. For example, if bit 5 is set in the `priorityremap` field of the `remappriority` register, then any port transaction with a `userpriority` value of 0x5 in the `mppriority` register is serviced ahead of any other transaction already in the single port queue.	RW	0x0

Port Sum of Weight Register Register Descriptions

This register is used to configure the DRAM burst operation scheduling.

Offset: `0xb0`

mpweight_0_4 on page 11-64
This register is used to configure the DRAM burst operation scheduling.

mpweight_1_4 on page 11-65
This register is used to configure the DRAM burst operation scheduling.

mpweight_2_4 on page 11-65
This register is used to configure the DRAM burst operation scheduling.

mpweight_3_4 on page 11-66
This register is used to configure the DRAM burst operation scheduling.

mpweight_0_4

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B0

Offset: `0x50B0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
staticweight_31_0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
staticweight_31_0 RW 0x0															

mpweight_0_4 Fields

Bit	Name	Description	Access	Reset
31:0	staticweight_31_0	Set static weight of the port. Each port is programmed with a 5 bit value. Port 0 is bits 4:0, port 1 is bits 9:5, up to port 9 being bits 49:45	RW	0x0

mpweight_1_4

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B4

Offset: 0x50B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
sumofweights_13_0 RW 0x0														staticweight_49_32 RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
staticweight_49_32 RW 0x0															

mpweight_1_4 Fields

Bit	Name	Description	Access	Reset
31:18	sumofweights_13_0	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0
17:0	staticweight_49_32	Set static weight of the port. Each port is programmed with a 5 bit value. Port 0 is bits 4:0, port 1 is bits 9:5, up to port 9 being bits 49:45	RW	0x0

mpweight_2_4

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B8

Offset: 0x50B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
sumofweights_45_14 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sumofweights_45_14 RW 0x0															

mpweight_2_4 Fields

Bit	Name	Description	Access	Reset
31:0	sumofweights_45_14	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0

mpweight_3_4

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250BC

Offset: 0x50BC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														sumofweights_63_46 RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sumofweights_63_46 RW 0x0															

mpweight_3_4 Fields

Bit	Name	Description	Access	Reset
17:0	sumofweights_63_46	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0

Document Revision History

Date	Version	Changes
June 2014	2014.6.30	• Added *Port Mappings* section. • Added *SDRAM Controller Memory Options* section. • Enhanced *Example of Configuration for TrustZone* section. • Added SDRAM Controller address map and registers.
December 2013	2013.12.30	• Added *Generating a Preloader Image for HPS with EMIF* section. • Added *Debugging HPS SDRAM in the Preloader* section. • Enhanced *Simulation* section.
November 2012	1.1	Added address map and register definitions section.
January 2012	1.0	Initial release.

On-Chip Memory 12

2014.06.30

cv_54009 Subscribe Send Feedback

The hard processor system (HPS) contains two types of on-chip memory. The on-chip memory types are the following:

- On-Chip RAM—The on-chip RAM provides 64 KB of general-purpose RAM.
- Boot ROM—The boot ROM contains the code required to boot the HPS from cold or warm reset.

Both memories connect to the level 3 (L3) interconnect.

Related Information

- **On-Chip RAM** on page 12-1
- **Boot ROM** on page 12-3

On-Chip RAM

Features of the On-Chip RAM

The on-chip RAM offers the following features:

- 64-bit slave interface
- 64 KB size
- Single-ported RAM
- Read acceptance of two, write acceptance of two, and a total acceptance of four
- Error correction code (ECC) support
- High throughput - read or write every clock cycle.

Related Information

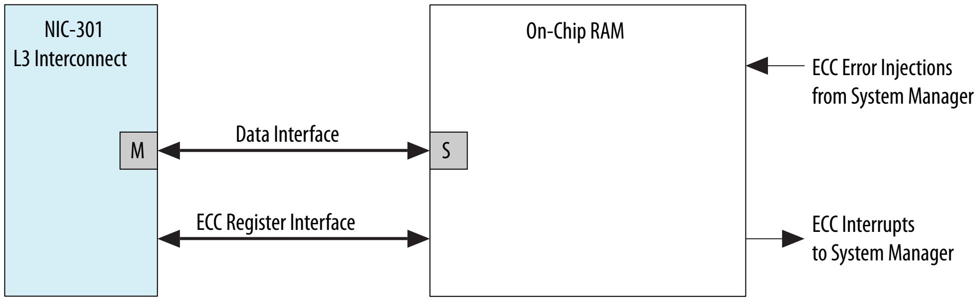
Clock Manager on page 2-1
For more information about the operating frequency and maximum throughput, refer to the *Clock Manager* chapter.

On-Chip RAM Block Diagram and System Integration

Transfers between memory and the NIC-301 L3 interconnect happen through a 64-bit interface, gated by the `l3_main_clk` interconnect clock. ECC logic detects single-bit, corrected and double-bit, uncorrected errors. The memory has a read acceptance of two, a write acceptance of two, and a total acceptance of two with round-robin arbitration.

The entire RAM is either secure or non-secure. Security is enforced by the NIC-301 L3 interconnect.

Figure 12-1: On-Chip RAM Block Diagram



Note: You must initialize the on-chip RAM before you enable the ECC support to prevent false ECC interrupts triggered by uninitialized bits.

Related Information

System Interconnect on page 7-1
For more information about security, refer to the *Interconnect* chapter.

Functional Description of the On-Chip RAM

The on-chip RAM serves as a general-purpose memory accessible from the FPGA.

The on-chip RAM uses an 64-bit slave interface. The slave interface supports transfers between memory and the NIC-301 L3 interconnect. All reads and writes are serviced in order.

Related Information

- Clock Manager on page 2-1
  For more information about the operating frequency and maximum throughput, refer to the *Clock Manager* chapter.
- Reset Manager on page 3-1

Clocks

The on-chip RAM is driven by the `l3_main_clk` interconnect clock.

The on-chip RAM uses an 64-bit slave interface. The slave interface supports transfers between memory and the NIC-301 L3 interconnect. All reads and writes are serviced in order.

Resets

The contents of the RAM remain unchanged on a cold or warm reset. Reset only clears the state associated with the slave interface.

The on-chip RAM reset is driven by the `onchip_ram_rst_n` interconnect reset.

On-Chip RAM Initialization

You must initialize the on-chip RAM before you enable the ECC. Failure to do so triggers spurious interrupts.

Initialize the on-chip RAM using the following steps:

1. Disable ECC interrupts
2. Enable ECC generation
3. Initialize memory by clearing the contents by writing 0x0 in the address space
4. Enable ECC interrupts

Boot ROM

This section describes hardware aspects of the HPS boot ROM.

Related Information

Booting and Configuration Introduction on page 30-1
For more information about the about the boot ROM software, refer to the *Booting and Configuration* appendix.

Features of the Boot ROM

The boot ROM offers the following features:

- 32-bit interface
- 64 KB size
- Single-ported ROM
- Read acceptance of two
- High throughput - read every clock cycle.

Related Information

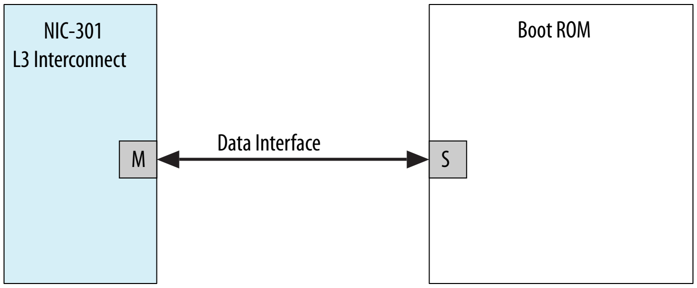
Clock Manager on page 2-1
For more information about the operating frequency and maximum throughput, refer to the *Clock Manager* chapter.

Boot ROM Block Diagram and System Integration

Transfers between memory and the NIC-301 L3 interconnect happen through a 32-bit data interface, gated by the `l3_main_clk` interconnect clock.

The entire RAM is either secure or nonsecure. Security is enforced by the NIC-301 L3 interconnect.

Figure 12-2: Boot ROM Block Diagram



Related Information

System Interconnect on page 7-1
For more information about security, refer to the *Interconnect* chapter.

Functional Description of the Boot ROM

The boot ROM is used only for booting the system. On a cold or warm reset of the microprocessor unit (MPU) subsystem, MPU0 executes the pre-bootloader code stored in the boot ROM.

The boot ROM uses an 32-bit slave interface. The slave interface supports transfers between memory and the NIC-301 L3 interconnect. All writes return an error response.

Clocks

The boot ROM is driven by the `l3_main_clk` interconnect clock.

Resets

The contents of the ROM remain unchanged on a cold or warm reset. Reset only clears the state associated with the slave interface.

The boot ROM reset is driven by the `boot_rom_rst_n` interconnect clock.

Related Information

- Clock Manager on page 2-1
  For more information about the operating frequency and maximum throughput, refer to the *Clock Manager* chapter.
- Reset Manager on page 3-1
- Booting and Configuration Introduction on page 30-1
  For more information about the about the boot ROM software, refer to the *Booting and Configuration* appendix.

On-Chip Memory Address Map and Register Definitions

The address map and register definitions for the On-Chip Memory consist of the following regions:

- Boot ROM Module
- On-chip RAM Module

Related Information

Introduction to Cyclone V Hard Processor System on page 1-1
The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter in the *Cyclone V Device Handbook, Volume 3.*

On-chip RAM Address Map

This is the address space allocated to the on-chip RAM. The on-chip RAM can be used by the HPS for storing data or user code.

Table 12-1: On-chip RAM Address Range

Memory Instance	Start Address	End Address
`OCRAM`	`0xFFFF0000`	`0xFFFFFFFF`

Boot ROM Address Map

This address range is allocated for the boot ROM.

Table 12-2: Boot ROM Address Range

Module Instance	Start Address	End Address
`bootROM`	`0xFFFD0000`	`0xFFFDFFFF`

Document Revision History

Table 12-3: Document Revision History

Date	Version	Changes
June 2014	2014.06-30	Added address maps and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.1	Added address map section
January 2012	1.0	Initial release

NAND Flash Controller 13

2014.07.31

cv_54010 Subscribe Send Feedback

The hard processor system (HPS) provides a NAND flash controller to interface with external NAND flash memory in Altera® system-on-a-chip (SoC) FPGA systems. You can use external flash memory to store a processor boot image, software, or as extra storage capacity for large applications or user data. The HPS NAND flash controller is based on the Cadence® Design IP® NAND Flash Memory Controller.

NAND Flash Controller Features

The NAND flash controller provides the following functionality and features:

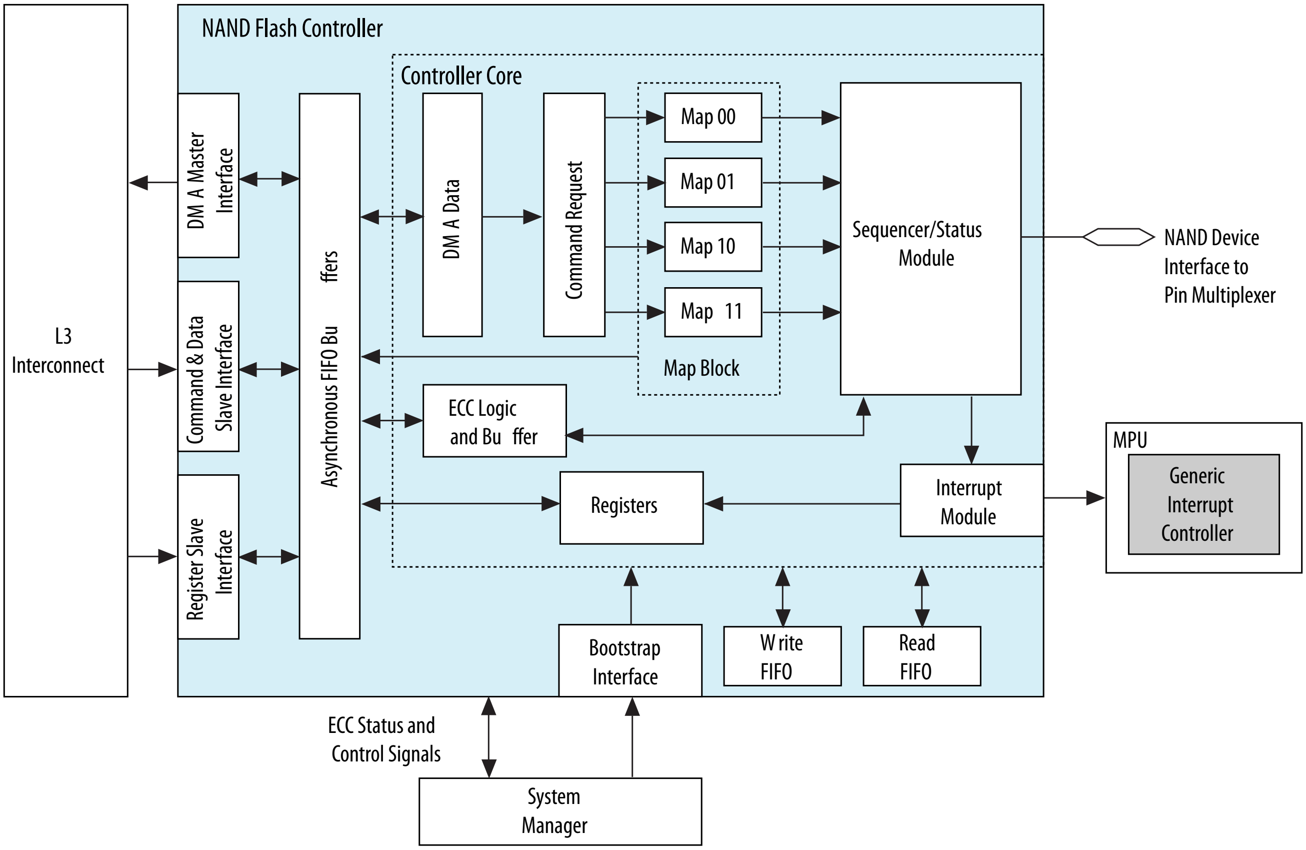
- Supports one x8 NAND flash device
- Supports Open NAND Flash Interface (ONFI) 1.0
- Supports NAND flash memories from Hynix, Samsung, Toshiba, Micron, and ST Micro
- Supports programmable 512 byte (4-, 8-, or 16-bit correction) or 1024 byte (24-bit correction) error correction code (ECC) sector size
- Supports pipeline read-ahead and write commands for enhanced read/write throughput
- Supports devices with 32, 64, 128, 256, 384, or 512 pages per block
- Supports multiplane devices
- Supports page sizes of 512 bytes, 2 kilobytes (KB), 4 KB, or 8 KB
- Supports single-level cell (SLC) and multi-level cell (MLC) devices with programmable correction capabilities
- Provides internal direct memory access (DMA)
- Provides programmable access timing

NAND Flash Controller Block Diagram and System Integration

The flash controller receives commands and data from the host through the command and data slave interface. The host accesses the flash controller’s control and status registers (CSRs) through the register slave interface. The flash controller handles all command sequencing and flash device interactions. The bootstrap interface supports configuration of the NAND flash controller when booting the HPS from NAND flash memory. The flash controller generates interrupts to the HPS Cortex-A9 MPCore™ processor generic interrupt controller. The DMA master interface provides accesses to and from the flash controller through the controller's built-in DMA.

Figure 13-1: NAND Flash Controller Block Diagram

The following figure shows integration of the NAND flash controller in the HPS.



Functional Description of the NAND Flash Controller

This section describes the functionality of the NAND flash controller.

Discovery and Initialization

The NAND flash controller requires a specific initialization sequence after the HPS receives power and the flash device is stable. During initialization, the flash controller queries the flash device and configures itself according to one of the following flash device types:

- ONFI 1.0 compliant devices
- Legacy (non-ONFI) NAND devices

The NAND flash controller identifies ONFI-compliant connected devices using ONFI discovery protocol, by sending the `Read Electronic Signature` command. For devices that do not recognize this command (especially for 512-byte page size devices), software must write to the system manager to assert the `bootstrap_512B_device` signal to identify the device type before reset is de-asserted.

To support booting and initialization, the `rdy_busy_in` pin must be connected. The NAND flash controller sends the `reset` command to the connected device.

The NAND flash controller performs the following initialization steps:

1. If the system manager is asserting `bootstrap_inhibit_init`, the flash controller goes directly to step 7.
2. When the device is ready, the flash controller sends the `ONFI Read ID` command to read the ONFI signature from the memory device, to determine whether an ONFI or a legacy device is connected.
3. If the data returned by the memory device has an ONFI signature, the flash controller then reads the device parameter page. The flash controller stores the relevant device feature information in internal memory control registers, enabling it to correctly program other registers in the flash device, and goes to step 5.
4. If the data does not have a valid ONFI signature, the flash controller assumes that it is a legacy (non-ONFI) device. The flash controller then performs the following steps:

   a. Sends the `reset` command to the device
   b. Reads the device signature information
   c. Stores the relevant values into internal memory controller registers
5. The flash controller resets the memory device. At the same time, it verifies the width of the memory interface. The HPS supports one 8-bit NAND flash device. As a result, the flash controller always detects an 8-bit memory interface.
6. The flash controller sends the `Page Load` command to block 0, page 0 of the device, configuring direct read access, so the processor can boot from that page. The processor can start reading from the first page of the flash memory, which is the expected location of the pre-loader software.

   Note: The system manager can bypass this step by asserting `bootstrap_inhibit_b0p0_load` before reset is de-asserted.
7. The flash controller sends the `reset` command to the flash.
8. The flash controller sets the value of the `rst_comp` bit in the `intr_status0` register in the `status` group.

Bootstrap Interface

The NAND flash controller provides a bootstrap interface that allows software to override the default behavior of the flash controller. The bootstrap interface contains four bits, which when set appropriately, allows the flash controller to skip the initialization phase and begin loading from flash memory immediately after reset. These bits are driven by software through the system manager. They are sampled by the NAND flash controller when the controller is released from reset.

Related Information

System Manager on page 5-1
For more information about the bootstrap interface control bits.

Bootstrap Setting Bits

The following table lists the relevant bootstrap setting bits, found in the system manager's `bootstrap` register, in the `nandgrp` group. This table also lists recommended bootstrap settings for a 512-byte page device.

Table 13-1: Bootstrap Setting Bits

Register	Value
`noinit`	1[22]
`page512`	1
`noloadb0p0`	1
`tworowaddr`	• 1—flash device supports two-cycle addressing • 0—flash device support three-cycle addressing

Related Information

Configuration by Host on page 13-4

Configuration by Host

If the system manager sets `bootstrap_inhibit_init` to 1, the NAND flash controller does not perform the discovery and initialization process. In this case, the host processor must configure the flash controller.

When performance is not a concern in the design, the timing registers can be left unprogrammed.

Related Information

Bootstrap Setting Bits on page 13-3
For recommended configuration-by-host settings to enable the basic read, write, and erase operations for a single-plane, 512 bytes/page device.

Recommended Bootstrap Settings for 512-Byte Page Device

Table 13-2: Recommended Bootstrap Settings for 512-Byte Page Device

Register[23]	Value
`devices_connected`	1
`device_width`	0 indicating an 8-bit NAND flash device
`number_of_planes`	1 indicating a single-plane device
`device_main_area_size`	The value of this register must reflect the flash device's page main area size.
`device_spare_area_size`	The value of this register must reflect the flash device's page spare area size.

[22] When this register is set, the NAND flash controller expects the host to program the related device parameter registers. For more information, refer to *Configuration by Host*.

[23] All registers are in the `config` group.

Register[23]	Value
`pages_per_block`	The value of this register must reflect number of pages per block in the flash device.

NAND Page Main and Spare Areas

Each NAND page has a main area and a spare area. The main area is intended for data storage. The spare area is intended for ECC and maintenance data, such as wear leveling information. Each block consists of a group of pages.

The sizes of the main and spare areas, and the number of blocks in a page, depend on the specific NAND device connected to the NAND flash controller. Therefore, the device-dependent registers, `device_main_area_size`, `device_spare_area_size`, and `pages_p er_block`, must be programmed to match the characteristics of the device.

If your software does not perform the discovery and initialization sequence, the software must include an alternative method to determine the correct value of the device-dependent registers. The HPS boot ROM code enables discovery and initialization by default (that is, `bootstrap_inhibit_init = 0`).

Clocks

Table 13-3: Clock Inputs to NAND Flash Controller

Clock Signal	Description
`nand_x_clk`	Clock for master and slave interfaces and the ECC sector buffer
`nand_clk`	Clock for the NAND flash controller

The frequency of `nand_x_clk` is four times the frequency of `nand_clk`.

For more information about the clock inputs, refer to the *Clock Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

Related Information

Clock Manager on page 2-1

Resets

The NAND flash controller has one reset signal, `nand_flash_rst_n`. The reset manager drives this signal to the NAND flash controller on a cold or warm reset.

Before the NAND flash controller comes out of the reset state, the pin multiplexers for the flash external interface must be configured.

For more information about the reset manager, refer to the *Reset Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

Related Information

Reset Manager on page 3-1

[23] All registers are in the `config` group.

Indexed Addressing

The NAND flash controller uses indexed addressing to reduce the address span consumed by the flash controller.

Indirect addressing is controlled by two registers, accessed through the command and data slave interface in the `nanddata` map, as described in *Register Map for Indexed Addressing*.

Related Information

Register Map for Indexed Addressing on page 13-6

Register Map for Indexed Addressing

Table 13-4: Register Map for Indexed Addressing

Register Name	Offset Address	Usage
`Control`	0x0	Software writes the 32-bit control information consisting of MAP command type, block, and page address. The upper four bits must be set to 0. For specific usage of the `Control` register, refer to *MAP00 Address Mapping*, *MAP01 Address Mapping*, *MAP10 Address Mapping*, and *MAP10 Operations*.
`Data`	0x10	The `Data` register is a page-size window into the NAND flash. By reading from or writing to locations starting at this offset, the software reads directly from or writes directly to the page and block of NAND flash memory specified by the `Control` register.

Related Information

- MAP00 Address Mapping on page 13-7
- MAP01 Address Mapping on page 13-8
- MAP10 Address Mapping on page 13-10
- MAP10 Operations on page 13-10

Indexed Addressing Host Usage

The host uses indexed addressing as follows:

1. Program the 32-bit index-address field into the `Control` register at offset 0x0 of the data/command slave port. This action provides the flash address parameters to the NAND flash controller.
2. Perform a 32-bit read or write in the `Data` register at offset 0x10 of the data/command slave port.
3. Perform additional 32-bit reads and writes if they are in the same page and block in flash memory.

It is unnecessary to write to the control register for every data transfer if a group of data transfers targets the same page and block address. For example, you can write the control register at the beginning of a page with the block and page address, and then read or write the entire page by directing consecutive transactions to the `Data` register.

Command Mapping

The NAND flash controller supports several flash controller-specific MAP commands, providing an abstraction level for programming a NAND flash device. By using the MAP commands, you can avoid directly programming device-specific commands. Using this abstraction layer provides enhanced performance. Commands take multiple cycles to send off-chip. The MAP commands let you initiate commands and let the flash controller sequence them off-chip to the NAND device.

The NAND flash controller supports the following flash controller-specific MAP commands:

- MAP00 Commands—boot-read or buffer read/write during read-modify-write operations
- MAP01 Commands—memory arrays read/write
- MAP10 Commands—NAND flash controller commands
- MAP11 Commands—low-level direct access

Related Information

- **MAP00 Commands** on page 13-7
- **MAP01 Commands** on page 13-8
- **MAP10 Commands** on page 13-9
- **MAP11 Commands** on page 13-11

MAP00 Commands

MAP00 commands access a page buffer in the NAND flash device. Addressing always begins at 0x0 and extends to the page size specified by the `device_main_area_size` and `device_spare_area_size` registers in the `config` group. You can use this command to perform a boot read. Use MAP00 commands in read-modify-write (RMW) operations to read or write any word in the buffer. MAP00 commands allow a direct data path to the page buffer in the device.

The host can access the page buffer directly using the MAP00 commands only if there are no other MAP01 or MAP10 commands active on the NAND flash controller.

Related Information

MAP00 Address Mapping on page 13-7

MAP00 Address Mapping

Table 13-5: MAP00 Address Mapping

Address Bits	Name	Description
31:28	(reserved)	Set to 0
27:26	`CMD_MAP`	Set to 0
25:13	(reserved)	Set to 0
12:2	`BUFF_ADDR`	Data width-aligned buffer address on the memory device. Maximum page access is 8 KB.
1:0	(reserved)	Set to 0

MAP00 Usage Limitations

The usage of this command under normal operations is limited to the following situations:

- It can be used to perform an Execute-in-Place (XIP) boot from the device; reading directly from the page buffer while booting directly from the device.
- MAP00 commands can be used to perform RMW operations where MAP00 writes are used to modify a read page in the device page buffer. Because the NAND flash controller does not perform ECC correction during such an operation, Altera does not recommend this method in an MLC device.
- In association with MAP11 commands, MAP00 commands provide a way for the host to directly access the device bypassing the hardware abstractions provided by NAND flash controller with MAP01 and MAP10 commands. This method is also used for debugging, or for issuing an operation that the flash controller might not support with MAP01 or MAP10 commands.

Restrictions:

- MAP00 commands cannot be used with MAP01 commands to read part of a page. Accesses using MAP01 commands must perform a complete page transfer.
- No ECC is performed during a MAP00 data access.
- DMA must be disabled (the `flag` bit of the `dma_enable` register in the `dma` group must be set to 0) while performing MAP00 operations.

MAP01 Commands

MAP01 commands transfer complete pages between the host memory and a specific page of the NAND flash device. Because the NAND flash controller supports only page addresses, the entire page must be read or written at once. The actual number of commands required depends on the size of the data transfer. You must use the same address until the entire page is transferred, even if multiple commands are required.

When the NAND flash controller receives a read command, it issues a load operation on the device, waits for the load to complete, and then returns read data. Read data must be read from the start of the page to the end of the page. Write data must be written from the start of the page to the end of the page.

When the NAND flash controller receives confirmation of the transfer, it issues commands to program the data into the device. The flash controller ignores the byte enables for read and write commands and transfers the entire data width.

Related Information

MAP01 Address Mapping on page 13-8

MAP01 Address Mapping

Table 13-6: MAP01 Address Mapping

Address Bits	Name	Description
31:28	(reserved)	Set to 0
27:26	`CMD_MAP`	Set to 1
25:24	(reserved)	Set to 0
23:<*M*> [24]	`BLK_ADDR`	Block address in the device

Address Bits	Name	Description
(<*M*>-1):0 [24]	`PAGE_ADDR`	Page address in the device

ECC

The NAND flash controller incorporates ECC on-the-fly correction that corrects data read from the device internally before transferring the data out from the flash controller. The ECC sector buffers store data, while the ECC engine computes the error location.

MAP01 Command Usage

Use the MAP01 command as follows:

- A complete page must be read or written using a MAP01 command. During such transfers, every transaction from the host must have the same block and page address. The NAND flash controller internally keeps track of how much of data it reads or writes.
- MAP00 commands cannot be used in between using MAP01 commands for reading or writing a page.
- DMA must be disabled (the `flag` bit of the `dma_enable` register in the `dma` group must be set to 0) while the host is performing MAP01 operations directly. If the host issues MAP01 commands to the NAND flash controller while DMA is enabled, the flash controller discards the request and generates an `unsup_cmd` interrupt.

MAP10 Commands

MAP10 commands provide an interface to the control plane of the NAND flash controller. MAP10 commands control special functions of the flash device, such as erase, lock, unlock, copy back, and page spare area access. Data passed in this command pathway targets the NAND flash controller rather than the flash device. Unlike other command types, the data (input or output) related to these transactions does not affect the contents of the flash device. Rather, this data specifies and performs the exact commands of the flash controller. Only the lower 16 bits of the `Data` register contain the relevant information.

Related Information

MAP10 Address Mapping on page 13-10

[24] <*M*> depends on the number of pages per block in the device. <*M*> = ceil(log2(<*device pages per block*>)) . Therefore, use the following values:

32 pages per block: <*M*>=5

64 pages per block: <*M*>=6

128 pages per block: <*M*>=7

256 pages per block: <*M*>=8

384 pages per block: <*M*>=9

512 pages per block: <*M*>=9

MAP10 Address Mapping

Table 13-7: MAP10 Address Mapping

Address Bits	Name	Description
31:28	(reserved)	Set to 0
27:26	`CMD_MAP`	Set to 2
25:24	(reserved)	Set to 0
23:<M>[25]	`BLK_ADDR`	Block address in the device
(<M>-1):0[25]	`PAGE_ADDR`	Page address in the device

MAP10 Operations

Table 13-8: MAP10 Operations

Command	Function
0x01	Sets block address for erase and initiates operation
0x10	Sets unlock start address
0x11	Sets unlock end address and initiates unlock
0x21	Initiates a lock of all blocks
0x31	Initiates a lock-tight of all blocks
0x41	Sets up for spare area access
0x42	Sets up for default area access

[25] <*M*> depends on the number of pages per block in the device. <*M*> = ceil(log2(<*device pages per block*>)) . Therefore, use the following values:

32 pages per block: <*M*>=5

64 pages per block: <*M*>=6

128 pages per block: <*M*>=7

256 pages per block: <*M*>=8

384 pages per block: <*M*>=9

512 pages per block: <*M*>=9

Command	Function
0x43	Sets up for main+spare area access
0x60	Loads page to the buffer for a RMW operation
0x61	Sets the destination address for the page buffer in RMW operation
0x62	Writes the page buffer for a RMW operation
0x1000	Sets copy source address
0x11<*PP*>	Sets copy destination address and initiates a copy of <*PP*> pages
0x20<*PP*>	Sets up a pipeline read-ahead of <*PP*> pages
0x21<*PP*>	Sets up a pipeline write of <*PP*> pages

MAP10 Command Usage

Use the MAP10 command as follows:

- MAP10 commands should be used to issue commands to the device, such as erase, copy-back, lock, or unlock.
- MAP10 pipeline commands should also be used to read or write consecutive multiple pages from the flash device within a device block boundary. The host must first issue a MAP10 pipeline read or write command and then issue MAP01 commands to do the actual data transfers. The MAP10 pipeline read or write command instructs the NAND flash controller to use high-performance commands such as cache or multiplane because the flash controller has knowledge of multiple consecutive pages to be read. The pages must not cross a block boundary. If a block boundary is crossed, the flash controller generates an unsupported command (`unsup_cmd`) interrupt and drops the command.
- Up to four pipeline read or write commands can be issued to the NAND flash controller.
- While the NAND flash controller is performing MAP10 pipeline read or write commands, DMA must be disabled (the `flag` bit of the `dma_enable` register in the `dma` group must be set to 0). DMA must be disabled because the host is directly transferring data from and to the flash device through the flash controller.

MAP11 Commands

MAP11 commands provide direct access to the NAND flash controller's address and control cycles, allowing software to issue the commands directly to the flash device using the `Command` and `Data` registers. The MAP11 command is useful if the flash device supports a device-specific command not supported by standard flash commands. It can also be useful for low-level debugging.

MAP11 commands provide a direct control path to the flash device. These commands execute command, address, and data read/write cycles directly on the NAND device interface. Command, address, and write data values are placed in the `Data` register. On a read, the returned data also appears in the `Data` register. The indirect address register encodes the control operation type. Command and address cycles to the device must be a write transaction on the host bus. For data cycles, the type of transaction on the host bus (read/write) determines the data cycle type on the device interface. The host can issue only single-beat accesses to the data slave port while using MAP11 commands.

Related Information

MAP11 Addressing Mapping on page 13-12

MAP11 Addressing Mapping

Table 13-9: MAP11 Addressing Mapping

Address Bits	Name	Description
31:28	(reserved)	Set to 0
27:26	`CMD_MAP`	Set to 3
25:2	(reserved)	Set to 0
1:0	`TYPE`	Sets the control type as follows: • 0 = Command cycle • 1 = Address cycle • 2 = Data Read/Write Cycle

MAP11 Command Usage

Use the MAP11 command as follows:

- Use MAP11 commands only in special cases, for debugging or sending device-specific commands that are not supported by the NAND flash controller.
- DMA must be disabled before you use MAP11 operations.
- The host can use only single beat access transfers when using MAP11 commands.

Note: MAP11 commands provide direct, unstructured access to the NAND flash device. Incorrect use can lead to unpredictable behavior.

Data DMA

The DMA transfers data with minimal host involvement. Software initiates data DMA with the MAP10 command.

The `flag` bit of the `dma_enable` register in the `dma` group enables data DMA functionality. Only enable or disable this functionality when there are no active transactions pending in the NAND flash controller. When the DMA is enabled, the flash controller initiates one DMA transfer per MAP10 command over the DMA master interface. When the DMA is disabled, all operations with the flash controller occur through the data/command slave interface.

The NAND flash controller supports up to four outstanding DMA commands, and ignores additional DMA commands. If software issues more than four outstanding DMA commands, the flash controller issues the `unsup_cmd` interrupt. On receipt of a DMA command, the flash controller performs command sequencing to transfer the number of pages requested in the DMA command. The DMA master reads or writes page data from the system memory in programmed burst-length chunks. After the DMA command completes, the flash controller issues an interrupt, and starts working on the next queued DMA command.

Pipelining allows the NAND flash controller to optimize its performance while executing back-to-back commands of the same type.

With certain restrictions, non-DMA MAP10 commands can be issued to the NAND flash controller while the flash controller is servicing DMA transactions. MAP00, MAP01, and MAP11 commands cannot be issued while DMA mode is enabled because the flash controller is operating in an extremely tightly-coupled, high-performance data transfer mode. On receipt of erroneous commands (MAP00, MAP01 or MAP11), the flash controller issues an `unsup_cmd` interrupt to inform the host about the violating command.

When the host issues a data DMA command, the NAND flash controller transfers data between the flash device and host memory if data DMA is enabled (the `flag` bit of the `dma_enable` register in the `dma` group is set to 1). On the completion of the transfer the flash controller informs the host by asserting an interrupt.

- A data DMA command is a type of MAP10 command. This command is interpreted by the data DMA engine and not by the flash controller core.
- No MAP01, MAP00, or MAP11 commands are allowed when DMA is enabled.
- Before the flash controller can accept data DMA commands, DMA must be enabled by setting the `flag` bit of the `dma_enable` register in the `dma` group.
- When DMA is enabled and the DMA engine initiates data transfers, ECC can be enabled for as-needed data correction concurrent with the data transfer.
- MAP10 commands are used along with data movements similar to MAP01 commands.
- With the exception of data DMA commands and MAP10 pipeline read and write commands, all other MAP10 commands such as erase, lock, unlock, and copy-back are forwarded to the flash controller.
- At any time, up to four outstanding data DMA commands can be handled by flash controller. During multi-page operations, the DMA transfer must not cross a flash block boundary. If it does, the flash controller generates an unsupported command (`unsup_cmd`) interrupt and drops the command.
- Data DMA commands are typically multi-page read and write commands with an associated pointer in host memory. The multi-page data is transferred to or from the host memory starting from the host memory pointer.
- Data DMA uses the `flash_burst_length` register in the `dma` group to determine the burst length value to drive on the interconnect. The data DMA hardware does not account for the interconnect's boundary crossing restrictions. The host must initialize the starting host address so that the DMA master burst does not cross a 4 KB boundary.

There are two methods for initiating a DMA transaction: the multitransaction DMA command, and the burst DMA command.

Multitransaction DMA Command

The NAND flash controller processes multitransaction DMA commands only if it receives all four command-data pairs in order. The flash controller responds to out-of-order commands with an `unsup_cmd` interrupt. The flash controller also responds with an `unsup_cmd` interrupt if sequenced commands are interleaved with other flash controller MAP commands.

To initiate DMA with a multitransaction DMA command, you send four command-data pairs to the NAND flash controller's data and control slave port, as shown in *Command-Data Pair Formats*.

Related Information

Command-Data Pair Formats on page 13-14

Command-Data Pair Formats

Table 13-10: Command-Data Pair 1

<table>
<tr><th></th><th>31:28</th><th>27:26</th><th>25:24</th><th colspan="4">23:<M></th><th>(<M> – 1):0</th></tr>
<tr><td>Command</td><td>0x0</td><td>0x2</td><td>0x0</td><td colspan="4">Block address</td><td>Page address</td></tr>
<tr><td></td><td colspan="4">31:16</td><td>15:12</td><td>11:8</td><td colspan="2">7:0</td></tr>
<tr><td>Data</td><td colspan="4">0x0</td><td>0x2</td><td>0x0 = Read
0x1 = Write</td><td colspan="2"><PP>= Number of pages</td></tr>
</table>

Table 13-11: Command-Data Pair 2

<table>
<tr><th></th><th>31:28</th><th>27:26</th><th>25:24</th><th colspan="3">23:8</th><th>7:0</th></tr>
<tr><td>Command</td><td>0x0</td><td>0x2</td><td>0x0</td><td colspan="3">Memory address high[27]</td><td>0x0</td></tr>
<tr><td></td><td colspan="4">31:16</td><td>15:12</td><td>11:8</td><td>7:0</td></tr>
<tr><td>Data</td><td colspan="4">0x0</td><td>0x2</td><td>0x2</td><td>0x0</td></tr>
</table>

Table 13-12: Command-Data Pair 3

<table>
<tr><th></th><th>31:28</th><th>27:26</th><th>25:24</th><th colspan="3">23:8</th><th>7:0</th></tr>
<tr><td>Command</td><td>0x0</td><td>0x2</td><td>0x0</td><td colspan="3">Memory address low[28]</td><td>0x0</td></tr>
<tr><td></td><td colspan="4">31:16</td><td>15:12</td><td>11:8</td><td>7:0</td></tr>
<tr><td>Data</td><td colspan="4">0x0</td><td>0x2</td><td>0x3</td><td>0x0</td></tr>
</table>

[26] <*M*> depends on the number of pages per block in the device. <*M*> = ceil(log2(<*device pages per block*>)). Therefore, use the following values:

32 pages per block: <*M*>=5

64 pages per block: <*M*>=6

128 pages per block: <*M*>=7

256 pages per block: <*M*>=8

384 pages per block: <*M*>=9

512 pages per block: <*M*>=9

[27] The buffer address in host memory, which must be aligned to 32 bits.

[28] The buffer address in host memory, which must be aligned to 32 bits.

Table 13-13: Command-Data Pair 4

<table>
<tr><th></th><th>31:28</th><th>27:26</th><th>25:24</th><th>23:17</th><th>16</th><th colspan="2">15:8</th><th>7:0</th></tr>
<tr><td><b>Command</b></td><td>0x0</td><td>0x2</td><td>0x0</td><td>0x0</td><td>INT</td><td colspan="2">Burst length</td><td>0x0</td></tr>
<tr><th></th><th colspan="5">31:16</th><th>15:12</th><th>11:8</th><th>7:0</th></tr>
<tr><td><b>Data</b></td><td colspan="5">0x0</td><td>0x2</td><td>0x4</td><td>0x0</td></tr>
</table>

Related Information

- **Indexed Addressing** on page 13-6
- **Burst DMA Command** on page 13-15

Using Multitransaction DMA Commands

If you want the NAND flash controller DMA to perform cacheable accesses then you must configure the cache bits by writing the `l3master` register in the `nandgrp` group in the system manager. The NAND flash controller DMA must be idle before you use the system manager to change its cache capabilities.

You can issue non-DMA MAP10 commands while the NAND flash controller is in DMA mode. For example, you might trigger a host-initiated page move between DMA commands, to achieve wear leveling. However, do not interleave non-DMA MAP10 commands between the command-data pairs in a set of multitransaction DMA commands. You must issue all four command-data pairs shown in the above tables before sending a different command.

Note: Do not issue MAP00, MAP01 or MAP11 commands while DMA is enabled.

MAP10 commands in multitransaction format are written to the `Data` register at offset 0x10 in `nanddata`, the same as MAP10 commands in increment four (INCR4) format (described in *Burst DMA Command*).

Related Information

- **Indexed Addressing** on page 13-6
- **Burst DMA Command** on page 13-15
- **System Manager** on page 5-1

Burst DMA Command

You can initiate a DMA transfer by sending a command to the NAND flash controller as a burst transaction of four 16-bit accesses. This form of DMA command might be useful for initiating DMA transfers from custom IP in the FPGA fabric. Most processor cores cannot use this form of DMA command, because they cannot control the width of the burst.

When DMA is enabled, the NAND flash controller recognizes the MAP10 pipeline DMA command as an INCR4 command, in the format shown in the following table. The address decoding for MAP10 pipeline DMA command remains the same, as shown in *MAP10 Address Mapping*.

[29] INT specifies the host interrupt to be generated at the end of the complete DMA transfer. INT controls the value of the `dma_cmd_comp` bit of the `intr_status0` register in the `status` group at the end of the DMA transfer. INT can take on one of the following values:

0—Do not interrupt host. The `dma_cmd_comp` bit is set to 0.

1—Interrupt host. The `dma_cmd_comp` bit is set to 1.

MAP10 commands in INCR4 format are written to the `Data` register at offset 0x10 in `nanddata`, the same as MAP10 commands in multitransaction format (described in *Multitransaction DMA Command*).

Table 13-14: MAP10 Burst DMA (INCR4) Command Structure

The following table lists the MAP10 burst DMA command structure. The burst DMA command carries the same information as the multi-transaction DMA command-data pairs, but in a very different format.

Data Beat	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Beat 0	0x2				0x0: read. 0x1: write.				<*PP*>=number of pages							
Beat 1 [30]	Memory address high															
Beat 2 [30]	Memory address low															
Beat 3	0x0							INT[31]	Burst length							

You can optionally send the 16-bit fields in the above table to the NAND flash controller as four separate bursts of length 1 in sequential order. Altera recommends this method.

If you want the NAND flash controller DMA to perform cacheable accesses, you must configure the cache bits by writing the `l3master` register in the `nandgrp` group in the system manager. The NAND flash controller DMA must be idle before you use the system manager to modify its cache capabilities.

Related Information

- **Multitransaction DMA Command** on page 13-13
- **MAP10 Address Mapping** on page 13-10
- **System Manager** on page 5-1

ECC

The NAND flash controller incorporates ECC logic to calculate and correct bit errors. The flash controller uses a Bose-Chaudhuri-Hocquenghem (BCH) algorithm for detection of multiple errors in a page.

The NAND flash controller supports 512- and 1024-byte ECC sectors. The flash controller inserts ECC check bits for every 512 or 1024 bytes of data, depending on the selected sector size. After 512 or 1024 bytes, the flash controller writes the ECC check bit information to the device page.

ECC information is striped in between 512 or 1024 bytes of data across the page. The NAND flash controller reads ECC information in the same pattern and the presence of errors is calculated according to 512 or 1024 bytes of data read.

[30] The buffer address in host memory, which must be aligned to 32 bits

[31] INT specifies the host interrupt to be generated at the end of the complete DMA transfer. INT controls the value of the `dma_cmd_comp` bit of the `intr_status0` register in the `status` group at the end of the DMA transfer. INT can take on one of the following values:

0—Do not interrupt host. The `dma_cmd_comp` bit is set to 0.

1—Interrupt host. The `dma_cmd_com p` bit is set to 1.

Correction Capability, Sector Size, and Check Bit Size

Table 13-15: Correction Capability, Sector Size, and Check Bit Size

Correction	Sector Size in Bytes	Check Bit Size in Bytes
4	512	8
8	512	14
16	512	26
24	1024	46

ECC Programming Modes

The NAND flash controller provides the following ECC programming modes that software uses to format a page:

- Main Area Transfer Mode
- Spare Area Transfer Mode
- Main+Spare Area Transfer Mode

Related Information

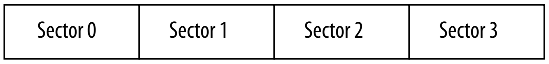
- **Main Area Transfer Mode** on page 13-17
- **Spare Area Transfer Mode** on page 13-17
- **Main+Spare Area Transfer Mode** on page 13-18

Main Area Transfer Mode

In main area transfer mode, when ECC is enabled, the NAND flash controller inserts ECC check bits in the data stream on writes and strips ECC check bits on reads. Software does not need to manage the ECC sectors when writing a page. ECC checking is performed by the flash controller, so software simply transfers the data.

If ECC is turned off, the NAND flash controller does not read or write ECC check bits.

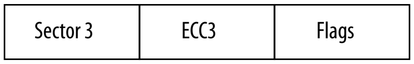
Figure 13-2: Main Area Transfer Mode Programming Model for ECC



Spare Area Transfer Mode

The NAND flash controller does not introduce or interpret ECC check bits in spare area transfer mode, and acts as a pass-through for data transfer.

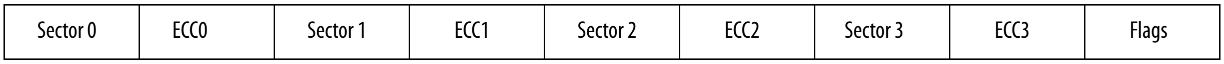
Figure 13-3: Spare Area Transfer Mode Programming Model for ECC



Main+Spare Area Transfer Mode

In main+spare area transfer mode, the NAND flash controller expects software to format a page as shown in following figure. When ECC is enabled during a write operation, the flash controller-generated ECC check bits replace the ECC check bit data provided by software. During read operations, the flash controller forwards the ECC check bits from the device to the host. If ECC is disabled, page data received from the software is written to the device, and read data received from the device is forwarded to the host.

Figure 13-4: Main+Spare Area Transfer Mode Programming Model for ECC



Sector 0	ECC0	Sector 1	ECC1	Sector 2	ECC2	Sector 3	ECC3	Flags

Preserving Bad Block Markers

When flash device manufacturers test their devices at the time of manufacture, they mark any bad device blocks that are found. Each bad block is marked at specific, known offsets, typically at the base of the spare area. A bad block marker is any byte value other than 0xFF (the normal state of erased flash).

Bad block markers can be overwritten by the last sector data in a page when ECC is enabled. This happens because the NAND flash controller also uses the main area of a page to store ECC information, which causes the last sector to spill over into the spare area. It is necessary for the system to preserve the bad block information prior to writing data, to ensure the correct identification of bad blocks in the flash device.

You can configure the NAND flash controller to skip over a specified number of bytes when it writes the last sector in a page to the spare area. This option allows the flash controller to preserve bad block markers. To use this option, write the desired offset to the `spare_area_skip_bytes` register in the `config` group. For example, if the device page size is 2 KB, and the device manufacturer stores the bad block markers in the first two bytes in the spare area, set the `spare_area_skip_bytes` register to 2. When the flash controller writes the last sector of the page that overlaps with the spare area, it starts at offset 2 in the spare area, skipping the bad block marker at offset 0. A value of 0 (default) specifies that no bytes are skipped. The value of `spare_area_skip_bytes` must be an even number. For example, if the bad block marker is a single byte, set `spare_area_skip_bytes` to 2.

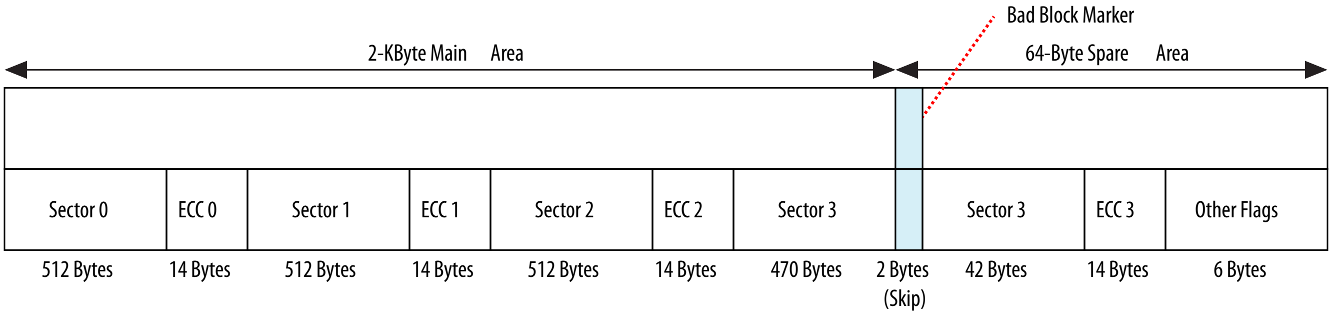
In main area transfer mode, the NAND flash controller does not skip the bad block marker. Instead, it overrides the bad block marker with the value programmed in the `spare_area_marker` register in the `config` group. This 8-bit register is used in conjunction with the `spare_area_skip_bytes` register in the `config` group to determine which bytes in the spare area of a page should be written with a the new marker value. For example, to mark a block as good set the `spare_area_marker` register to 0xFF and set the `spare_area_skip_bytes` register to the number of bytes that the marker should be written to, starting from the base of the spare area.

In the spare area transfer mode, the NAND flash controller ignores the `spare_area_skip_bytes` and `spare_area_marker` registers. The flash controller transfers the data exactly as received from the host or device.

In the main+spare area transfer mode, the NAND flash controller starts writing the last sector in a page into the spare area, starting at the offset specified in the `spare_area_skip_bytes` register. However, the area containing the bad block identifier information is overwritten by the data the host writes into the page. The host writes both the data sectors and the bad block markers. The flash controller depends on the host software to set up the bad block markers properly before writing the data.

Figure 13-5: Bad Block Marker

The following figure shows an example of how the NAND flash controller can skip over a bad block marker. In this example, the flash device has a 2-KB page with a 64-byte spare area. A 14-byte sector ECC is shown, with 8 byte per sector correction.



Related Information

- **Main+Spare Area Transfer Mode** on page 13-18
  For more information about the formatting of the data.
- **Transfer Mode Operations** on page 13-25
  For detailed information about configuring the NAND flash controller for default, spare, or main +spare area transfer mode.

Error Correction Status

The ECC error correction information (ECCCorInfo_b01) register, in the ecc group, contains error correction information for each read or write that the NAND flash controller performs. The ECCCorInfo_b01 register contains ECC error correction information in the max_errors_b0 and uncor_err_b0 fields.

At the end of data correction for the transaction in progress, ECCCorInfo_b01 holds the maximum number of corrections applied to any ECC sector in the transaction. In addition, this register indicates whether the transaction as a whole has correctable errors, uncorrectable errors, or no errors at all. A transaction has no errors when none of the ECC sectors in the transaction has any errors. The transaction is marked as uncorrectable if any one of the sectors is uncorrectable. The transaction is marked as correctable if any one sector has correctable errors and none is uncorrectable.

At the end of each transaction, the host must read this register. The value of this register provides data to the host about the block. The host can take corrective action after the number of correctable errors encountered reaches a particular threshold value.

Interface Signals

Table 13-16: NAND Flash Interface Signals

As listed in the following table, the HPS I/O pins support a single x8 device.

Signal	Width	I/O	Description
ad	8	in/out	Command, address and data for the flash device

Signal	Width	I/O	Description
`ale`	1	out	Address latch enable
`ce_n`	1	out	Output Active-low chip enable
`cle`	1	out	Command latch enable
`re_n`	1	out	Active-low read enable signal
`rb`	1	in	Ready/busy signal
`we_n`	1	out	Active-low write enable signal
`wp_n`	1	out	Active-low write protect signal

NAND Flash Controller Programming Model

This section describes how the NAND flash controller is to be programmed by software running on the microprocessor unit (MPU).

Note: If you write a configuration register and follow it up with a data operation that is dependent on the value of this configuration register, Altera recommends that you read the value of the register before performing the data operation. This read operation ensures that the posted write of the register is completed and takes effect before the data operation is issued to the NAND flash controller.

Basic Flash Programming

This section describes the steps that must be taken by the software to access and control the NAND flash controller.

NAND Flash Controller Optimization Sequence

The software must configure the flash device for interrupt or polling mode, using the `bank0` bit of the `rb_pin_enabled` register in the `config` group. If the device is in polling mode, the software must also program the additional registers, to select the times and frequencies of the polling. Program the following registers in the `config` group:

- Set the `rb_pin_enabled` register to the desired mode of operation for each flash device.
- For polling mode, set the `load_wait_cnt` register to the appropriate value depending on the speed of operation of the NAND flash controller, and the desired wait value.
- For polling mode, set the `program_wait_cnt` register to the appropriate value by software depending on the speed of operation of the NAND flash controller, and the desired wait value.
- For polling mode, set the `erase_wait_cnt` register to the appropriate value by software depending on the speed of operation of the NAND flash controller, and the desired wait value.
- For polling mode, set the `int_mon_cyccnt` register to the appropriate value by software depending on the speed of operation of the NAND flash controller, and the desired wait value.

At any time, the software can change any flash device from interrupt mode to polling mode or vice-versa, using the `bank0` bit of the `rb_pin_enabled` register.

The software must ensure that the particular flash device does not have any outstanding transactions before changing the mode of operation for that particular flash device.

Device Initialization Sequence

At initialization, the host software must program the following registers in the `config` group:

- Set the `devices_connected` register to 1.
- Set the `device_width` register to 8.
- Set the `device_main_area_size` register to the appropriate value.
- Set the `device_spare_area_size` register to the appropriate value.
- Set the `pages_per_block` register according to the parameters of the flash device.
- Set the `number_of_planes` register according to the parameters of the flash device.
- If the device allows two ROW address cycles, the `flag` bit of the `two_row_addr_cycles` register must be set to 1. The host program can ensure this condition either of the following ways:
  - Set the `flag` bit of the `bootstrap_two_row_addr_cycles` register to 1 prior to the NAND flash controller's reset initialization sequence, causing the flash controller to initialize the bit automatically.
  - Set the `flag` bit of the `two_row_addr_cycles` register directly to 1.
- Clear the `chip_enable_dont_care` register in the `config` group to 0.

The NAND flash controller can identify the flash device features, allowing you to initialize the flash controller registers to interface correctly with the device, as described in *Discovery and Initialization*.

However, a few NAND devices do not follow any universally accepted identification protocol. If connected to such a device, the NAND flash controller cannot identify it correctly. If you are using such a device, your software must use other means to ensure that the initialization registers are set up correctly.

Related Information

Discovery and Initialization on page 13-2

Device Operation Control

This section provides a list of registers that you need to program while choosing to use multi-plane or cache operations on the device. If the device does not support multi-plane operations or cache operations, then these registers can be left at their power-on reset values with no impact on the functionality of the NAND flash controller. Even if the device supports these sequences, the software may choose not to use these sequences and can leave these registers at their power-on reset values.

Program the following registers in the `config` group to achieve the best performance from a given device:

- Set `flag` bit in the `multiplane_operation` register in the `config` group to 1 if the device supports multi-plane operations to access the data on the flash device connected to the NAND flash controller. If the flash controller is set up for multi-plane operations, the number of pages to be accessed is always a multiple of the number of planes in the device.
- If the NAND flash controller is configured for multi-plane operation, and if the device has support for multi-plane read command sequence, set the `multiplane_read_enable` register in the `config` group.
- If the device implements multiplane address restrictions, set the `flag` bit in the `multiplane_addr_restrict` register to 1.
- Initialize the `die_mask` and `first_block_of_next_plane` registers as per device requirements.

- If the device supports cache command sequences, enable the cache_write_enable and cache_read_enable registers in the config group.
- Clear the flag bit of the copyback_disable register in the config group to 0 if the device does not support the copyback command sequences. The register defaults to enabled state.
- The read_mode, write_mode and copyback_mode registers, in the config group, currently need not be written by software, because the NAND flash controller is capable of using the correct sequences based on a combination of some multi-plane or cache-related settings of the NAND flash controller and the manufacturer ID. If at some future time these settings change, program the registers to accommodate the change.

ECC Enabling

Before you start any data operation on the flash device, you must decide whether you want the ECC enabled or disabled.

If the ECC needs enabling, set up the appropriate correction level depending on the page size and the spare area available on the device.

Set the flag bit in the ecc_enable register in the config group to 1 to enable ECC. If enabled, the following registers in the config group must be programmed accordingly, else they can be ignored:

- Initialize the ecc_correction register to the appropriate correction level.
- Program the spare_area_skip_bytes and spare_area_marker registers in the config group if the software needs to preserve the bad block marker.

Related Information

ECC on page 13-16
For detailed information about ECCs.

NAND Flash Controller Performance Registers

These registers specify the size of the bursts on the device interface, which maximizes the overall performance on the NAND flash controller.

Initialize the flash_burst_length register in the dma group to a value which maximizes the performance of the device interface by minimizing the number of bursts required to transfer a page.

Interrupt and DMA Enabling

Prior to initiating any data operation on the NAND flash controller, the software must set appropriate interrupt status register bits. If the software chooses to use the DMA logic in the flash controller, then the appropriate DMA enable and interrupts bits in the register space must be set.

1. Set the flag bit in the global_int_enable register in the config group to 1, to enable global interrupt.
2. Set the relevant bits of the intr_en0 register in the status group to 1 before sending any operations if the flash controller is in interrupt mode.
3. Enable DMA if your application needs DMA mode. Enable DMA by setting the flag bit of the dma_enable register in the dma group. Altera recommends that the software reads back this register to ensure that the mode change is accepted before sending a DMA command to the flash controller.
4. If the DMA is enabled, then set up the appropriate bits of the dma_intr_en register in the dma group.

Order of Interrupt Status Bits Assertion

The following interrupt status bits, in the intr_status0 register in the status group, are listed in the order of interrupt bit setting:

1. time_out—All other interrupt bits are set to 0 when the watchdog time_out bit is asserted.
2. dma_cmd_comp—This interrupt status bit is the last to be asserted during a DMA operation to transfer data. This bit signifies the completion of data transfer sequence.
3. pipe_cpybck_cmd_comp—This bit is asserted when a copyback command or the last page of a pipeline command completes.
4. locked_blk—This bit is asserted when a program (or erase) is performed on a locked block.
5. INT_act—No relationship with other interrupt status bits. Indicates a transition from 0 to 1 on the ready_busy pin value for that flash device.
6. rst_comp—No relationship with other interrupt status bits. Occurs after a reset command has completed.
7. For an erase command:
   a. erase_fail (if failure)
   b. erase_comp
8. For a program command:
   a. locked_blk (if performed on a locked block)
   b. pipe_cmd_err (if the pipeline sequence is broken by a MAP01 command)
   c. page_xfer_inc (at the end of each page data transfer)
   d. program_fail (if failure)
   e. pipe_cpybck_cmd_comp
   f. program_comp
   g. dma_cmd_comp (If DMA enabled)
9. For a read command:
   a. pipe_cmd_err (if the pipeline sequence is broken by a MAP01 command)
   b. page_xfer_inc (at the end of each page data transfer)
   c. pipe_cpybck_cmd_comp
   d. load_comp
   e. ecc_uncor_error (if failure)
   f. dma_cmd_comp (If DMA enabled)

Timing Registers

You must optimize the following registers for your flash device's speed grade and clock frequency. The NAND flash controller operates correctly with the power-on reset values. However, functioning with power-on reset values is a non-optimal mode that provides loose timing (large margins to the signals).

Set the following registers in the config group to optimize the NAND flash controller for the speed grade of the connected device and frequency of operation of the flash controller:

- twhr2_and_we_2_re
- tcwaw_and_addr_2_data
- re_2_we
- acc_clks
- rdwr_en_lo_cnt
- rdwr_en_hi_cnt

- max_rd_delay
- cs_setup_cnt
- re_2_re

Registers to Ignore

You do not need to initialize the following registers in the config group:

- The transfer_spare_reg register—Data transfer mode can be initialized using MAP10 commands.
- The write_protect register—Does not need initializing unless you are testing the write protection feature.

Flash-Related Special Function Operations

This section describes all the special functions that can be performed on the flash memory.

The functions are defined by MAP10 commands as described in *Command Mapping*.

Related Information

Command Mapping on page 13-7

Erase Operations

Before data can be written to flash, an erase cycle must occur. The NAND flash memory controller supports single block and multi-plane erases.

The controller decodes the block address from the indirect addressing shown in *MAP10 Address Mapping*.

Related Information

MAP10 Address Mapping on page 13-10

Single Block Erase

A single command is needed to complete a single-block erase, as follows:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the desired erase block.
2. Write 0x01 to the Data register.

For a single block erase, the register multiplane_operation in the config group must be reset.

After device completes erase operation, the controller generates an erase_comp interrupt. If the erase operation fails, the erase_fail interrupt is issued. The failing block's address is updated in the err_block_addr0 register in the status group.

Multi-Plane Erase

For multi-plane erases, the number_of_planes register in the config group holds the number of planes in the flash device, and the block address specified must be aligned to the number of planes in the device. The NAND flash controller consecutively erases each block of the memory, up to the number of planes available. Issue this command as follows:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the desired erase block.
2. Write 0x01 to the Data register.

For multi-plane erase, the register multiplane_operation in the config group must be set.

After the device completes erase operation on all planes, the NAND flash controller generates an erase_comp interrupt. If the erase operation fails on any of the blocks in a multi-plane erase command, an erase_fail interrupt is issued. The failing block's address is updated in the err_block_addr0 register in the status group.

Lock Operations

The NAND flash controller supports the following features:

- Flash locking—The NAND flash controller supports all flash locking operations.

  The flash device itself might have limited support for these functions. If the device does not support locking functions, the flash controller ignores these commands.
- Lock-tight—With the lock-tight feature, the NAND flash controller can prevent lock status from being changed. After the memory is locked tight, the flash controller must be reset before any flash area can be locked or unlocked.

Unlocking a Span of Memory Blocks

To unlock several blocks of memory, perform the following steps:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the starting address of the area to unlock.
2. Write 0x10 to the Data register.
3. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the ending address of the area to unlock.
4. Write 0x11 to the Data register.

When unlocking a range of blocks, the start block address must be less than the end block address. Otherwise, the NAND flash controller exhibits undetermined behavior.

Locking All Memory Blocks

To lock the entire memory:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to any memory address.
2. Write 0x21 to the Data register.

Setting Lock-Tight on All Memory Blocks

After the lock-tight is applied, unlocked areas cannot be locked, and locked areas cannot be unlocked. To lock-tight the entire memory:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to any memory address.
2. Write 0x31 to the Data register.

To disable the lock-tight, reset the memory controller.

Transfer Mode Operations

You can configure the NAND flash controller in one of the following modes of data transfer:

- Default area transfer mode
- Spare area transfer mode
- Main+spare area transfer mode

The NAND flash controller determines the default transfer mode from the setting of transfer_spare_reg register in the config group. Use MAP10 commands to dynamically change the transfer mode from the existing mode to the new mode. All subsequent commands are in the new mode of transfer. You must consider that transfer modes can be changed at logical data transfer boundaries. For example:

- At the beginning or end of a page in case of single page read or write.
- At the beginning or end of a complete multi-page pipeline read or write command.

transfer_spare_reg and MAP10 Transfer Mode Commands

The following table lists the functionality of the MAP10 transfer mode commands, and their mappings to the transfer_spare_reg register in the config group.

Table 13-17: transfer_spare_reg and MAP10 Transfer Mode Commands

transfer_spare_reg	MAP10 Transfer Mode Commands	Resulting NAND Flash Controller Mode
0	0x42	Main[32]
0	0x41	Spare
0	0x43	Main+spare
1	0x42	Main+spare[32]
1	0x41	Spare
1	0x43	Main+spare

Related Information

MAP10 Commands on page 13-9

For detailed information about the MAP10 commands.

Configure for Default Area Access

You only need to configure for default area access if the transfer mode was previously changed to spare area or main+spare area. To configure default area access:

1. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to any block.
2. Write 0x42 to the Data register.

The NAND flash controller determines the default area transfer mode from the setting of the transfer_spare_reg register in the config group. If it is set to 1, then the transfer mode becomes main +spare area, otherwise it is main area.

Configure for Spare Area Access

To access only the spare area of the flash device, use the MAP10 command to set up the NAND flash controller to read or write only the spare area on the device. After the flash controller is set up, use

[32] Default access mode (0x42) maps to either main (only) or main+spare mode, depending on the value of transfer_spare_reg.

MAP01 read and write commands to access the spare area of the appropriate block and page addresses. To configure the NAND flash controller to access the spare area only, perform the following steps:

1. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the target block.
2. Write 0x41 to the `Data` register.

Configure for Main+Spare Area Access

To configure the NAND flash controller to access the main+spare area:

1. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ ADDR` field to the target block.
2. Write 0x43 to the `Data` register.

Read-Modify-Write Operations

To read a specific page or modify a few words, bytes, or bits in a page, use the RMW operations. A read command copies the desired data from flash memory to a page buffer. You can then modify the information in the buffer using MAP00 buffer read and write commands and issue another command to write that information back to the memory.

The read-modify-write command operates on an entire page. This command is also useful for a copy type operation, where most of a page is saved to a new location. In this type of operation, the NAND flash controller reads the data, modifies a specified number of words in the page, and then writes the modified page to a new location.

Note: Because the data is modified within the page buffer of the flash device, the NAND flash controller ECC hardware is not used in RMW operations. Software must update the ECC during RMW operations.

Note: For a read-modify-write command to work with hardware ECC, the entire page must be read into system memory, modified, then written back to flash without relying on the RMW feature.

Read-Modify-Write Operation Flow

1. Start the flow by reading a page from the memory:
   - Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the starting address of the desired block.
   - Write 0x60 to the `Data` register.

   This step makes the page available to you in the page buffer in the flash device.
2. Provide the destination page address:
   - Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the destination address of the desired block.
   - Write 0x61 to the `Data` register.

   This step initiates the page program and provides the destination address to the device.
3. Use the MAP00 page buffer read and write commands to modify the data in the page buffer.
4. Write the page buffer data back to memory:
   - Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the same destination address.
   - Write 0x62 to the `Data` register.

This step performs the write.

After the device completes the load operation, the NAND flash controller issues a `load_comp` interrupt. A `program_comp` interrupt is issued when the host issues the write command and the device completes the program operation.

If the page program operation (as a part of an RMW operation) results in a program failure in the device, `program_fail` interrupt is issued. The failing page's block and page address is updated in the `err_block_addr0` and `err_page_addr0` registers in the `status` group.

Copy-back Operations

The NAND flash controller supports copy back operations. However, the flash device might have limited support for this function. If you attempt to perform a copy-back operation on a device that does not support copy-back, the NAND flash controller triggers an interrupt. An interrupt is also triggered if the source block is not specified before the destination block is specified, or if the destination block is not specified in the next command following a source block specification.

The NAND flash controller cannot do ECC validation in case of copy-back commands. The flash controller copies the ECC data, but does not check it during the copy operation.

Note: Altera recommends that you use copy-back only if the ECC implemented in the flash controller is strong enough so that the next access can correct accumulated errors.

The 8-bit value <*PP*> specifies the number of pages for copy-back. With this feature, the NAND flash controller can copy multiple consecutive pages with a single command. When you issue a copy-back command, the flash controller performs the operation in the background. The flash controller puts other commands on hold until the current copy-back completes.

For a multi-plane device, if the `flag` bit in the `multiplane_operation` register in the `config` group is set to 1, multi-plane copy-back is available as an option. In this case, the block address specified must be plane-aligned and the value <*PP*> must specify the total number of pages to copy as a multiple of the number of planes. The block address continues incrementing, keeping the page address fixed, for the total number of planes in the device before incrementing the page address.

A `pipe_cpyba ck_cmd_comp` interrupt is generated when the flash controller has completed copy-back operation of all <*PP*> pages. If any page program operation (as a part of copy back operation) results in a program failure in the device, the `program_fail` interrupt is issued. The failing page's block and page address is updated in the `err_block_addr0` and `err_page_addr0` registers in the `status` group.

Copying a Memory Area (Single Plane)

To copy <*PP*> pages from one memory location to another:

1. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the starting address of the area to be copied.
2. Write 0x1000 to the `Data` register.
3. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the starting address of the new area to be written.
4. Write 0x11<*PP*> to the `Data` register, where <*PP*> is the number of pages to copy.

Copying a Memory Area (Multi-Plane)

To copy <*PP*> pages from one memory location to another:

1. Set the flag bit of the multiplane_operation register in the config group to 1.
2. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the starting address of the area to be copied. The address must be plane-aligned.
3. Write 0x1000 to the Data register.
4. Write to the command register, setting the CMD_MAP field to 2 and the BLK_ADDR field to the starting address of the new area to be written. This address must also be plane-aligned.
5. Write 0x11<*PP*> to the Data register, where <*PP*> is the number of pages to copy.

The parameter <*PP*> must be a multiple of the number of planes in the device.

Pipeline Read-Ahead and Write-Ahead Operations

The NAND flash controller supports pipeline read-ahead and write-ahead operations. However, the flash device might have limited support for this function. If the device does not support pipeline read-ahead or write-ahead, the flash controller processes these commands as standard reads or writes.

The NAND flash controller can handle at the most four outstanding pipeline commands, queued up in the order in which the flash controller received the commands. The flash controller operates on the pipeline command at the head of the queue until all the pages corresponding to the pipeline command are executed. The flash controller then pops the pipeline command at the head of the queue and proceeds to work on the next pipeline command in the queue.

The pipeline read-ahead function allows for a continuous reading of the flash memory. On receiving a pipeline read command, the flash controller immediately issues a load command to the device. While data is read out with MAP01 commands in a consecutive or multi-plane address pattern, the flash controller maintains additional cache or multi-plane read command sequencing for continuous streaming of data from the flash device.

The pipeline write-ahead function allows for a continuous writing of the flash memory. While data is written with MAP01 commands in a consecutive or multi-plane address pattern, the NAND flash controller maintains cache or multi-plane command sequences for continuous streaming of data into the flash device.

MAP01 commands must read or write pages in the same sequence that the pipelined commands were issued to the NAND flash controller. If the host issues multiple pipeline commands, pages must be read or written in the order the pipeline commands were issued. It is not possible to read or write pages for a second pipeline command before completing the first pipeline command. If the pipeline sequence is broken by a MAP01 command, the pipe_cmd_err interrupt is issued, and the flash controller clears the pipeline command queue. The flash controller services the violating incoming MAP01 read or write request with a normal page read or write sequence.

For a multi-plane device that supports multi-plane programming, you must set the flag bit of the multiplane_operation register in the config group to 1. In this case, the data is interleaved into page-size chunks to consecutive blocks.

Pipeline read-ahead commands can read data from the queue in this interleaved fashion. The parameter <PP> denotes the total number of pages in multiples of the number of planes available, and the block address must be plane-aligned, which keeps the page address constant while incrementing the block address for each page-size chunk of data. After reading from every plane, the NAND flash controller increments the page address and resets the block address to the initial address. You can also use pipeline write-ahead commands in multi-plane mode. The write operation works similarly to the read operation, holding the page address constant while incrementing the block address until all planes are written.

Note: The same four-entry queue is used to queue the address and page count for pipeline read-ahead and write-ahead commands. This commonality requires that you use MAP01 commands to read out all pages for a pipeline read-ahead command before the next pipeline command can be processed. Similarly, you must write to all pages pertaining to pipeline write-ahead command before the next pipeline command can be processed.

Because the value of the flag bit of the multiplane_operation register in the config group determines pipeline read-ahead or write-ahead behavior, it can only be changed when the pipeline registers are empty.

When the host issues a pipeline read-ahead command, and the flash controller is idle, the load operation occurs immediately.

Note: The read-ahead command does not return the data to the host, and the write-ahead command does not write data to the flash address. The NAND flash controller loads the read data. The read data is returned to the host only when the host issues MAP01 commands to read the data. Similarly, the flash controller loads the write data, and writes it to the flash only when the host issues MAP01 commands to write the data.

A pipe_cpyback_cmd_comp interrupt is generated when the NAND flash controller has finished processing a pipeline command and has discarded that command from its queue. At this point of time, the host can send another pipeline command. A pipeline command is popped from the queue, and an interrupt is issued when the flash controller has started processing the last page of pipeline command. Hence, the pipe_cpyback_cmd_comp interrupt is issued prior to the last page load in the case of a pipeline read command and start of data transfer of the last page to be programmed, in the case of a pipeline write command.

An additional program_comp interrupt is generated when the last page program operation completes in the case of a pipeline write command.

If the device command set requires the NAND flash controller to issue a load command for the last page in the pipeline read command, a load_comp interrupt is generated after the last page load operation completes.

For pipeline write commands, if any page program results in a failure in the device, a program_fail interrupt is issued. The failing page's block and page address is updated in the err_block_addr0 and err_page_addr0 registers in the status group.

The pipeline commands sequence advanced commands in the device, such as cache and multi-plane. When the NAND flash controller receives a multi-page read or write pipeline command, it sequences commands sent to the device depending on settings in the following registers in the config group:

- cache_read_enable
- cache_write_enable
- multiplane_operation

For a device that supports cache read sequences, the flag bit of the cache_read_enable register must be set to 1. The NAND flash controller sequences each multi-page pipeline read command as a cache read sequence. For a device that supports cache program command sequences, cache_write_enable must be set. The flash controller sequences each multi-page write pipeline command as a cache write sequence.

For a device that has multi-planes and supports multi-plane program commands, the NAND flash controller register multiplane_operation, in the config group, must be set. On receiving the multi-page pipeline write command, the flash controller sequences the device with multi-plane program commands and expects that the host transfers data to the flash controller in an even-odd block increment addressing mode.

Set Up a Single Area for Pipeline Read-Ahead

To set up an area for pipeline read-ahead, perform the following steps:

1. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the starting address of the block to pre-read.
2. Write 0x20<*PP*> to the `Data` register, where the 0 sets this command as a read-ahead and <*PP*> is the number of pages to pre-read. The pages must not cross a block boundary. If a block boundary is crossed, the NAND flash controller generates an unsupported command (`unsup_cmd`) interrupt and drops the command.

The read-ahead command is a hint to the flash device to start loading the next page in the page buffer as soon as the previous page buffer operation has completed. After you set up the read-ahead, use a MAP01 command to actually read the data. In the MAP01 command, specify the same starting address as in the read-ahead.

If the read command received following a pipeline read-ahead request is not to a pre-read page, then an interrupt bit is set to 1 and the pipeline read-ahead or write-ahead registers are cleared. You must issue a new pipeline read-ahead request to re-load the same data. You must use MAP01 commands to read all of the data that is pre-read before the NAND flash controller returns to the idle state.

Set Up a Single Area for Pipeline Write-Ahead

To set up an area for pipeline write-ahead:

1. Write to the command register, setting the `CMD_MAP` field to 2 and the `BLK_ADDR` field to the starting address of the block to pre-write.
2. Write 0x21<PP> to the `Data` register, where the value 1 sets this command as a write-ahead and <PP> is the number of pages to pre-write. The pages must not cross a block boundary. If a block boundary is crossed, the NAND flash controller generates an unsupported command (`unsup_cmd`) interrupt and drops the command.

After you set up the write-ahead, use a MAP01 command to actually write the data. In the MAP01 command, specify the same starting address as in the write-ahead.

If the write command received following a pipeline write-ahead request is not to a pre-written page, then an interrupt bit is set to 1 and the pipeline read-ahead or write-ahead registers are cleared. You must issue a new pipeline write-ahead request to configure the write logic.

You must use MAP01 commands to write all of the data that is pre-written before the NAND flash controller returns to the idle state.

NAND Flash Controller Address Map and Register Definitions

The address map and register definitions for the NAND Flash Controller consist of the following regions:

- NAND Flash Controller Module Data
- NAND Flash Controller Module Registers

NAND Controller Module Data (AXI Slave) Address Map on page 13-32
This address space is allocated for indexed addressing by the NAND flash controller. For more information, please refer to the *NAND Flash Controller* chapter of the *Hard Processor System Technical Reference Manual*.

NAND Flash Controller Module Registers (AXI Slave) Address Map on page 13-32
Registers in the NAND Flash Controller module accessible via its register AXI slave

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**
  Web-based address map and register definitions.

NAND Controller Module Data (AXI Slave) Address Map

This address space is allocated for indexed addressing by the NAND flash controller. For more information, please refer to the *NAND Flash Controller* chapter of the *Hard Processor System Technical Reference Manual*.

Table 13-18: NAND Controller Module Data Space Address Range

Module Instance	Start Address	End Address
NAND_DATA	0xFF900000	0xFF9FFFFF

NAND Flash Controller Module Registers (AXI Slave) Address Map

Registers in the NAND Flash Controller module accessible via its register AXI slave

Base Address: 0xFFB80000

Configuration registers

Register	Offset	Width	Acces s	Reset Value	Description
device_reset on page 13-39	0x0	32	RW	0x0	
transfer_spare_reg on page 13-40	0x10	32	RW	0x0	
load_wait_cnt on page 13-41	0x20	32	RW	0x1F4	
program_wait_cnt on page 13-41	0x30	32	RW	0x1F4	
erase_wait_cnt on page 13-42	0x40	32	RW	0x1F4	
int_mon_cyccnt on page 13-43	0x50	32	RW	0x1F4	
rb_pin_enabled on page 13-43	0x60	32	RW	0x1	
multiplane_operation on page 13-44	0x70	32	RW	0x0	
multiplane_read_enable on page 13-45	0x80	32	RW	0x0	

Register	Offset	Width	Access	Reset Value	Description
copyback_disable on page 13-45	0x90	32	RW	0x0	
cache_write_enable on page 13-46	0xA0	32	RW	0x0	
cache_read_enable on page 13-46	0xB0	32	RW	0x0	
prefetch_mode on page 13-47	0xC0	32	RW	0x1	
chip_enable_dont_care on page 13-48	0xD0	32	RW	0x0	
ecc_enable on page 13-48	0xE0	32	RW	0x1	
global_int_enable on page 13-49	0xF0	32	RW	0x0	
twhr2_and_we_2_re on page 13-50	0x100	32	RW	0x1432	
tcwaw_and_addr_2_data on page 13-50	0x110	32	RW	0x1432	
re_2_we on page 13-51	0x120	32	RW	0x32	
acc_clks on page 13-52	0x130	32	RW	0x0	
number_of_planes on page 13-52	0x140	32	RW	0x0	
pages_per_block on page 13-53	0x150	32	RW	0x0	
device_width on page 13-53	0x160	32	RW	0x3	
device_main_area_size on page 13-54	0x170	32	RW	0x0	
device_spare_area_size on page 13-54	0x180	32	RW	0x0	
two_row_addr_cycles on page 13-55	0x190	32	RW	0x0	
multiplane_addr_restrict on page 13-56	0x1A0	32	RW	0x0	
ecc_correction on page 13-56	0x1B0	32	RW	0x8	
read_mode on page 13-57	0x1C0	32	RW	0x0	

Register	Offset	Width	Acces s	Reset Value	Description
write_mode on page 13-59	0x1D0	32	RW	0x0	
copyback_mode on page 13-59	0x1E0	32	RW	0x0	
rdwr_en_lo_cnt on page 13-60	0x1F0	32	RW	0x12	
rdwr_en_hi_cnt on page 13-61	0x200	32	RW	0xC	
max_rd_delay on page 13-62	0x210	32	RW	0x0	
cs_setup_cnt on page 13-62	0x220	32	RW	0x3	
spare_area_skip_bytes on page 13-63	0x230	32	RW	0x0	
spare_area_marker on page 13-64	0x240	32	RW	0xFFFF	
devices_connected on page 13-64	0x250	32	RW	0x0	
die_mask on page 13-65	0x260	32	RW	0x0	
first_block_of_next_plane on page 13-65	0x270	32	RW	0x1	
write_protect on page 13-66	0x280	32	RW	0x1	
re_2_re on page 13-67	0x290	32	RW	0x32	
por_reset_count on page 13-67	0x2A0	32	RW	0x13B	
watchdog_reset_count on page 13-68	0x2B0	32	RW	0x5B9A	

Device parameters

Register	Offset	Width	Acces s	Reset Value	Description
manufacturer_id on page 13-69	0x300	32	RW	0x0	
device_id on page 13-70	0x310	32	RO	0x0	
device_param_0 on page 13-70	0x320	32	RO	0x0	

Register	Offset	Width	Access	Reset Value	Description
device_param_1 on page 13-71	0x330	32	RO	0x0	
device_param_2 on page 13-71	0x340	32	RO	0x0	
logical_page_data_size on page 13-72	0x350	32	RO	0x0	
logical_page_spare_size on page 13-72	0x360	32	RO	0x0	
revision on page 13-73	0x370	32	RO	0x5	
onfi_device_features on page 13-73	0x380	32	RO	0x0	
onfi_optional_commands on page 13-74	0x390	32	RO	0x0	
onfi_timing_mode on page 13-74	0x3A0	32	RO	0x0	
onfi_pgm_cache_timing_mode on page 13-75	0x3B0	32	RO	0x0	
onfi_device_no_of_luns on page 13-76	0x3C0	32	RW	0x0	
onfi_device_no_of_blocks_per_lun_l on page 13-76	0x3D0	32	RO	0x0	
onfi_device_no_of_blocks_per_lun_u on page 13-77	0x3E0	32	RO	0x0	
features on page 13-77	0x3F0	32	RO	0x841	

Interrupt and Status Registers

Register	Offset	Width	Access	Reset Value	Description
transfer_mode on page 13-79	0x400	32	RO	0x0	
intr_status0 on page 13-80	0x410	32	RW	0x0	
intr_en0 on page 13-82	0x420	32	RW	0x2000	
page_cnt0 on page 13-84	0x430	32	RO	0x0	
err_page_addr0 on page 13-84	0x440	32	RO	0x0	

Register	Offset	Width	Access	Reset Value	Description
err_block_addr0 on page 13-85	0x450	32	RO	0x0	
intr_status1 on page 13-85	0x460	32	RW	0x0	
intr_en1 on page 13-87	0x470	32	RW	0x2000	
page_cnt1 on page 13-88	0x480	32	RO	0x0	
err_page_addr1 on page 13-89	0x490	32	RO	0x0	
err_block_addr1 on page 13-89	0x4A0	32	RO	0x0	
intr_status2 on page 13-90	0x4B0	32	RW	0x0	
intr_en2 on page 13-91	0x4C0	32	RW	0x2000	
page_cnt2 on page 13-93	0x4D0	32	RO	0x0	
err_page_addr2 on page 13-93	0x4E0	32	RO	0x0	
err_block_addr2 on page 13-94	0x4F0	32	RO	0x0	
intr_status3 on page 13-94	0x500	32	RW	0x0	
intr_en3 on page 13-96	0x510	32	RW	0x2000	
page_cnt3 on page 13-98	0x520	32	RO	0x0	
err_page_addr3 on page 13-98	0x530	32	RO	0x0	
err_block_addr3 on page 13-99	0x540	32	RO	0x0	

ECC registers

Register	Offset	Width	Access	Reset Value	Description
ECCCorInfo_b01 on page 13-99	0x650	32	RO	0x0	
ECCCorInfo_b23 on page 13-100	0x660	32	RO	0x0	

DMA registers

Register	Offset	Width	Access	Reset Value	Description
dma_enable on page 13-102	0x700	32	RW	0x0	
dma_intr on page 13-102	0x720	32	RW	0x0	
dma_intr_en on page 13-103	0x730	32	RW	0x0	
target_err_addr_lo on page 13-103	0x740	32	RO	0x0	
target_err_addr_hi on page 13-104	0x750	32	RO	0x0	
flash_burst_length on page 13-104	0x770	32	RW	0x1	
chip_interleave_enable_and_allow_int_reads on page 13-105	0x780	32	RW	0x10	
no_of_blocks_per_lun on page 13-106	0x790	32	RW	0xF	
lun_status_cmd on page 13-107	0x7A0	32	RW	0x7878	

Configuration registers Register Descriptions

Common across all types of flash devices, configuration registers setup the basic operating modes of the controller

Offset: 0x0

device_reset on page 13-39
Device reset. Controller sends a RESET command to device. Controller resets bit after sending command to device

transfer_spare_reg on page 13-40
Default data transfer mode. (Ignored during Spare only mode)

load_wait_cnt on page 13-41
Wait count value for Load operation

program_wait_cnt on page 13-41
Wait count value for Program operation

erase_wait_cnt on page 13-42
Wait count value for Erase operation

int_mon_cyccnt on page 13-43
Interrupt monitor cycle count value

rb_pin_enabled on page 13-43
Interrupt or polling mode. Ready/Busy pin is enabled from device.

multiplane_operation on page 13-44
Multiplane transfer mode. Pipelined read, copyback, erase and program commands are transfered in multiplane mode

multiplane_read_enable on page 13-45
Device supports multiplane read command sequence

copyback_disable on page 13-45
Device does not support copyback command sequence

cache_write_enable on page 13-46
Device supports cache write command sequence

cache_read_enable on page 13-46
Device supports cache read command sequence

prefetch_mode on page 13-47
Enables read data prefetching to faster performance

chip_enable_dont_care on page 13-48
Device can work in the chip enable dont care mode

ecc_enable on page 13-48
Enable controller ECC check bit generation and correction

global_int_enable on page 13-49
Global Interrupt enable and Error/Timeout disable.

twhr2_and_we_2_re on page 13-50

tcwaw_and_addr_2_data on page 13-50

re_2_we on page 13-51
Timing parameter between re high to we low (Trhw)

acc_clks on page 13-52
Timing parameter from read enable going low to capture read data

number_of_planes on page 13-52
Number of planes in the device

pages_per_block on page 13-53
Number of pages in a block

device_width on page 13-53
I/O width of attached devices

device_main_area_size on page 13-54
Page main area size of device in bytes

device_spare_area_size on page 13-54
Page spare area size of device in bytes

two_row_addr_cycles on page 13-55
Attached device has only 2 ROW address cycles

multiplane_addr_restrict on page 13-56
Address restriction for multiplane commands

ecc_correction on page 13-56
Correction capability required

read_mode on page 13-57
The type of read sequence that the controller will follow for pipe read commands.

write_mode on page 13-59
The type of write sequence that the controller will follow for pipe write commands.

copyback_mode on page 13-59
The type of copyback sequence that the controller will follow.

rdwr_en_lo_cnt on page 13-60
Read/Write Enable low pulse width

rdwr_en_hi_cnt on page 13-61
Read/Write Enable high pulse width

max_rd_delay on page 13-62
Max round trip read data delay for data capture

cs_setup_cnt on page 13-62
Chip select setup time

spare_area_skip_bytes on page 13-63
Spare area skip bytes

spare_area_marker on page 13-64
Spare area marker value

devices_connected on page 13-64
Number of Devices connected on one bank

die_mask on page 13-65
Indicates the die differentiator in case of NAND devices with stacked dies.

first_block_of_next_plane on page 13-65
The starting block address of the next plane in a multi plane device.

write_protect on page 13-66
This register is used to control the assertion/de-assertion of the WP# pin to the device.

re_2_re on page 13-67
Timing parameter between re high to re low (Trhz) for the next bank

por_reset_count on page 13-67
The number of cycles the controller waits after reset to issue the first RESET command to the device.

watchdog_reset_count on page 13-68
The number of cycles the controller waits before flagging a watchdog timeout interrupt.

device_reset

Device reset. Controller sends a RESET command to device. Controller resets bit after sending command to device

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80000`

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												bank3 RW 0x0	bank2 RW 0x0	bank1 RW 0x0	bank0 RW 0x0

device_reset Fields

Bit	Name	Description	Access	Reset
3	bank3	Issues reset to bank 3. Controller resets the bit after reset command is issued to device.	RW	0x0
2	bank2	Issues reset to bank 2. Controller resets the bit after reset command is issued to device.	RW	0x0
1	bank1	Issues reset to bank 1. Controller resets the bit after reset command is issued to device.	RW	0x0
0	bank0	Issues reset to bank 0. Controller resets the bit after reset command is issued to device.	RW	0x0

transfer_spare_reg

Default data transfer mode. (Ignored during Spare only mode)

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

transfer_spare_reg Fields

Bit	Name	Description	Access	Reset
0	flag	On all read or write commands through Map 01, if this bit is set, data in spare area of memory will be transfered to host along with main area of data. The main area will be transfered followed by spare area. [list][*]1 - MAIN+SPARE [*]0 - MAIN[/list]	RW	0x0

load_wait_cnt

Wait count value for Load operation

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x1F4															

load_wait_cnt Fields

Bit	Name	Description	Access	Reset
15:0	value	Number of clock cycles after issue of load operation before NAND Flash Controller polls for status. This values is of relevance for status polling mode of operation and has been provided to minimize redundant polling after issuing a command. After a load command, the first polling will happen after this many number of cycles have elapsed and then on polling will happen every int_mon_cyccnt cycles. The default values is equal to the default value of int_mon_cyccnt	RW	0x1F4

program_wait_cnt

Wait count value for Program operation

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80030

Offset: 0x30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x1F4															

program_wait_cnt Fields

Bit	Name	Description	Access	Reset
15:0	value	Number of clock cycles after issue of program operation before NAND Flash Controller polls for status. This values is of relevance for status polling mode of operation and has been provided to minimize redundant polling after issuing a command. After a program command, the first polling will happen after this many number of cycles have elapsed and then on polling will happen every int_mon_cyccnt cycles. The default values is equal to the default value of int_mon_cyccnt. The controller internally multiplies the value programmed into this register by 16 to provide a wider range for polling.	RW	0x1F4

erase_wait_cnt

Wait count value for Erase operation

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80040

Offset: 0x40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x1F4															

erase_wait_cnt Fields

Bit	Name	Description	Access	Reset
15:0	value	Number of clock cycles after issue of erase operation before NAND Flash Controller polls for status. This values is of relevance for status polling mode of operation and has been provided to minimize redundant polling after issuing a command. After a erase command, the first polling will happen after this many number of cycles have elapsed and then on polling will happen every int_mon_cyccnt cycles. The default values is equal to the default value of int_mon_cyccnt. The controller internally multiplies the value programmed into this register by 16 to provide a wider range for polling.	RW	0x1F4

int_mon_cyccnt

Interrupt monitor cycle count value

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x1F4															

int_mon_cyccnt Fields

Bit	Name	Description	Access	Reset
15:0	value	In polling mode, sets the number of cycles Denali Flash Controller must wait before checking the status register. This register is only used when R/B pins are not available to NAND Flash Controller.	RW	0x1F4

rb_pin_enabled

Interrupt or polling mode. Ready/Busy pin is enabled from device.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80060

Offset: 0x60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												bank3 RW 0x0	bank2 RW 0x0	bank1 RW 0x0	bank0 RW 0x1

rb_pin_enabled Fields

Bit	Name	Description	Access	Reset
3	bank3	Sets Denali Flash Controller in interrupt pin or polling mode [list][*]1 - R/B pin enabled for bank 3. Interrupt pin mode. [*]0 - R/B pin disabled for bank 3. Polling mode.[/list]	RW	0x0
2	bank2	Sets Denali Flash Controller in interrupt pin or polling mode [list][*]1 - R/B pin enabled for bank 2. Interrupt pin mode. [*]0 - R/B pin disabled for bank 2. Polling mode.[/list]	RW	0x0
1	bank1	Sets Denali Flash Controller in interrupt pin or polling mode [list][*]1 - R/B pin enabled for bank 1. Interrupt pin mode. [*]0 - R/B pin disabled for bank 1. Polling mode.[/list]	RW	0x0
0	bank0	Sets Denali Flash Controller in interrupt pin or polling mode [list][*]1 - R/B pin enabled for bank 0. Interrupt pin mode. [*]0 - R/B pin disabled for bank 0. Polling mode.[/list]	RW	0x1

multiplane_operation

Multiplane transfer mode. Pipelined read, copyback, erase and program commands are transfered in multiplane mode

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80070

Offset: 0x70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

multiplane_operation Fields

Bit	Name	Description	Access	Reset
0	`flag`	[list][*]1 - Multiplane operation enabled [*]0 - Multiplane operation disabled[/list]	RW	0x0

multiplane_read_enable

Device supports multiplane read command sequence

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80080

Offset: 0x80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

multiplane_read_enable Fields

Bit	Name	Description	Access	Reset
0	`flag`	Certain devices support dedicated multiplane read command sequences to read data in the same fashion as is written with multiplane program commands. This bit set should be set for the above devices. When not set, pipeline reads in multiplane mode will still happen in the order of multiplane writes, though normal read command sequences will be issued to the device. [list][*]1 - Device supports multiplane read sequence [*]0 - Device does not support multiplane read sequence[/list]	RW	0x0

copyback_disable

Device does not support copyback command sequence

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

copyback_disable Fields

Bit	Name	Description	Access	Reset
0	flag	[list][*]1 - Copyback disabled [*]0 - Copyback enabled[/list]	RW	0x0

cache_write_enable

Device supports cache write command sequence

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

cache_write_enable Fields

Bit	Name	Description	Access	Reset
0	flag	[list][*]1 - Cache write supported [*]0 - Cache write not supported[/list]	RW	0x0

cache_read_enable

Device supports cache read command sequence

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800B0

Offset: 0xB0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

cache_read_enable Fields

Bit	Name	Description	Access	Reset
0	flag	[list][*]1 - Cache read supported [*]0 - Cache read not supported[/list]	RW	0x0

prefetch_mode

Enables read data prefetching to faster performance

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800C0

Offset: 0xC0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
prefetch_burst_length RW 0x0												Reserved			prefetch_en RW 0x1

prefetch_mode Fields

Bit	Name	Description	Access	Reset
15:4	prefetch_burst_length	If prefetch_en is set and prefetch_burst_length is set to ZERO, the controller will start prefetching data only after the receiving the first Map01 read command for the page. If prefetch_en is set and prefetch_burst_length is set to a non-ZERO, valid value, the controller will start prefetching data corresponding to this value even before the first Map01 for the current page has been received. The value written here should be in bytes.	RW	0x0
0	prefetch_en	Enable prefetch of Data	RW	0x1

chip_enable_dont_care

Device can work in the chip enable dont care mode

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800D0

Offset: 0xD0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

chip_enable_dont_care Fields

Bit	Name	Description	Access	Reset
0	flag	Controller can interleave commands between banks when this feature is enabled. [list][*]1 - Device in dont care mode [*]0 - Device cares for chip enable[/list]	RW	0x0

ecc_enable

Enable controller ECC check bit generation and correction

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800E0

Offset: 0xE0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x1

ecc_enable Fields

Bit	Name	Description	Access	Reset
0	flag	Enables or disables controller ECC capabilities. When enabled, controller calculates ECC check-bits and writes them onto device on program operation. On page reads, check-bits are recomputed and errors reported, if any, after comparing with stored check-bits. When disabled, controller does not compute check-bits. [list][*]1 - ECC Enabled [*]0 - ECC disabled[/list]	RW	0x1

global_int_enable

Global Interrupt enable and Error/Timeout disable.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB800F0

Offset: 0xF0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							error_rpt_disable RW 0x0	Reserved			timeout_disable RW 0x0	Reserved			flag RW 0x0

global_int_enable Fields

Bit	Name	Description	Access	Reset
8	error_rpt_disable	Command and ECC uncorrectable failures will not be reported when this bit is set	RW	0x0

Bit	Name	Description	Access	Reset
4	timeout_disable	Watchdog timer logic will be de-activated when this bit is set.	RW	0x0
0	flag	Host will receive an interrupt only when this bit is set.	RW	0x0

twhr2_and_we_2_re

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80100

Offset: 0x100

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="6">twhr2
RW 0x14</td><td colspan="2">Reserved</td><td colspan="6">we_2_re
RW 0x32</td></tr>
</table>

twhr2_and_we_2_re Fields

Bit	Name	Description	Access	Reset
13:8	twhr2	Signifies the number of controller clocks that should be introduced between the last command of a random data output command to the start of the data transfer.	RW	0x14
5:0	we_2_re	Signifies the number of bus interface nand_mp_clk clocks that should be introduced between write enable going high to read enable going low. The number of clocks is the function of device parameter Twhr and controller clock frequency.	RW	0x32

tcwaw_and_addr_2_data

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80110

Offset: 0x110

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		tcwaw RW 0x14						Reserved		addr_2_data RW 0x32					

tcwaw_and_addr_2_data Fields

Bit	Name	Description	Access	Reset
13:8	tcwaw	Signifies the number of controller clocks that should be introduced between the command cycle of a random data input command to the address cycle of the random data input command.	RW	0x14
5:0	addr_2_data	Signifies the number of bus interface nand_mp_clk clocks that should be introduced between address latch enable going low to write enable going low. The number of clocks is the function of device parameter Tadl and controller clock frequency.	RW	0x32

re_2_we

Timing parameter between re high to we low (Trhw)

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80120

Offset: 0x120

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										value RW 0x32					

re_2_we Fields

Bit	Name	Description	Access	Reset
5:0	value	Signifies the number of bus interface nand_mp_clk clocks that should be introduced between read enable going high to write enable going low. The number of clocks is the function of device parameter Trhw and controller clock frequency.	RW	0x32

acc_clks

Timing parameter from read enable going low to capture read data

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80130

Offset: 0x130

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0x0			

acc_clks Fields

Bit	Name	Description	Access	Reset
3:0	value	Signifies the number of bus interface nand_mp_clk clock cycles, controller should wait from read enable going low to sending out a strobe of nand_mp_clk for capturing of incoming data.	RW	0x0

number_of_planes

Number of planes in the device

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80140

Offset: 0x140

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													value RW 0x0		

number_of_planes Fields

Bit	Name	Description	Access	Reset
2:0	value	Controller will read Electronic Signature of devices and populate this field as the number of planes information is present in the signature. For 512B device, this information needs to be programmed by software. Software could also choose to override the populated value. The values in the fields should be as follows[list] [*]3'h0 - Monoplane device [*]3'h1 - Two plane device [*]3'h3 - 4 plane device [*]3'h7 - 8 plane device [*]All other values - Reserved[/list]	RW	0x0

pages_per_block

Number of pages in a block

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80150

Offset: 0x150

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

pages_per_block Fields

Bit	Name	Description	Access	Reset
15:0	value	Controller will read Electronic Signature of devices and populate this field. The PAGE512 field of the System Manager NANDGRP_BOOTSTRAP register will determine the value of this field to be of 32. Software could also choose to override the populated value.	RW	0x0

device_width

I/O width of attached devices

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80160

Offset: 0x160

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														value RW 0x3	

device_width Fields

Bit	Name	Description	Access	Reset
1:0	value	Controller will read Electronic Signature of devices and populate this field. Software could also choose to override the populated value although only one value is supported. The values in this field should be as follows[list][*]2'h00 - 8bit device[*]All other values - Reserved[/list]	RW	0x3

device_main_area_size

Page main area size of device in bytes

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80170

Offset: 0x170

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

device_main_area_size Fields

Bit	Name	Description	Access	Reset
15:0	value	Controller will read Electronic Signature of devices and populate this field. The PAGE512 field of the System Manager NANDGRP_BOOTSTRAP register will determine the value of this field to be 512. Software could also choose to override the populated value.	RW	0x0

device_spare_area_size

Page spare area size of device in bytes

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80180

Offset: 0x180

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

device_spare_area_size Fields

Bit	Name	Description	Access	Reset
15:0	value	Controller will read Electronic Signature of devices and populate this field. The PAGE512 field of the System Manager NANDGRP_BOOTSTRAP register will determine the value of this field to be 16. Software could also choose to override the populated value.	RW	0x0

two_row_addr_cycles

Attached device has only 2 ROW address cycles

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80190

Offset: 0x190

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

two_row_addr_cycles Fields

Bit	Name	Description	Access	Reset
0	flag	This flag must be set for devices which allow for 2 ROW address cycles instead of the usual 3. Alternatively, the TWOROWADDR field of the System Manager NANDGRP_BOOTSTRAP register when asserted will set this flag.	RW	0x0

multiplane_addr_restrict

Address restriction for multiplane commands

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801A0

Offset: 0x1A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

multiplane_addr_restrict Fields

Bit	Name	Description	Access	Reset
0	flag	This flag must be set for devices which require that during multiplane operations all but the address for the last plane should have their address cycles tied low. The last plane address cycles has proper values. This ensures multiplane address restrictions in the device.	RW	0x0

ecc_correction

Correction capability required

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801B0

Offset: 0x1B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RW 0x8							

ecc_correction Fields

Bit	Name	Description	Access	Reset
7:0	value	The required correction capability can be a number less than the configured error correction capability. A smaller correction capability will lead to lesser number of ECC check-bits being written per ECC sector.	RW	0x8

read_mode

The type of read sequence that the controller will follow for pipe read commands.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801C0

Offset: 0x1C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0x0			

read_mode Fields

Bit	Name	Description	Access	Reset
3:0	value	The values in the field should be as follows[list] [*]4'h0 - This value informs the controller that the pipe read sequence to follow is of a normal read. For 512 byte page devices, Normal read sequence is, C00, Address, Data, ... For devices with page size greater that 512 bytes, the sequence is, C00, Address, C30, Data..... [*]4'h1 - This value informs the controller that the pipe read sequence to follow is of a Cache Read with the following sequence, C00, Address, C30, C31, Data, C31, Data, ..., C3F, Data. [*]4'h2 - This value informs the controller that the pipe read sequence to follow is of a Cache Read with the following sequence, C00, Address, C31, Data, Data, ..., C34. [*]4'h3 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Read with the following sequence, C00, Address, C00, Address, C30, Data, C06, Address, CE0, Data..... [*]4'h4 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Read with the following sequence, C60, Address, C60, Address, C30, C00, Address, C05, Address, CE0, Data, C00, Address, C05, Address, CE0, Data..... [*]4'h5 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Cache Read with the following sequence, C60, Address, C60, Address, C30, C31, C00, Address, C05, Address, CE0, Data, C00, Address, C05, Address, CE0, Data, ..., C3F, C00, Address, C05, Address, CE0, Data, C00, Address, C05, Address, CE0, Data [*]4'h6 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Read with the following sequence, C00, Address, C32, .., C00, Address, C30, C06, Address, CE0, Data, C06, Address, CE0, Data,.... [*]4'h7 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Cache Read with the following sequence, C00, Address, C32,..., C00, Address, C30, C31,C06, Address, CE0, Data, C31, C06, Address, CE0, Data, C3F, C06, Address, CE0, Data.... [*]4'h8 - This value informs the controller that the pipe read sequence to follow is of a 'N' Plane Cache Read with the following sequence, C60, Address, C60, Address, C33, C31, C00, Address, C05, Address, CE0, Data, C00, Address, C05, Address, CE0, Data, ..., C3F, C00, Address, C05, Address, CE0, Data, C00, Address, C05, Address, CE0, Data [*]4'h9 - 4'h15 - Reserved. [/list] ..... indicates that the previous sequence is repeated till the last page.	RW	0x0

write_mode

The type of write sequence that the controller will follow for pipe write commands.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801D0

Offset: 0x1D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0x0			

write_mode Fields

Bit	Name	Description	Access	Reset
3:0	value	The values in the field should be as follows[list] [*]4'h0 - This value informs the controller that the pipe write sequence to follow is of a normal write with the following sequence, C80, Address, Data, C10..... [*]4'h1 - This value informs the controller that the pipe write sequence to follow is of a Cache Program with the following sequence, C80, Address, Data, C15, ..., C80, Address, Data, C10. [*]4'h2 - This value informs the controller that the pipe write sequence to follow is of a Two/Four Plane Program with the following sequence, C80, Address, Data, C11, C81, Address, Data, C10..... [*]4'h3 - This value informs the controller that the pipe write sequence to follow is of a 'N' Plane Program with the following sequence, C80, Address, Data, C11, C80, Address, Data, C10..... [*]4'h4 - This value informs the controller that the pipe write sequence to follow is of a 'N' Plane Cache Program with the following sequence, C80, Address, Data, C11, C80, Address, Data, C15.....C80, Address, Data, C11, C80, Address, Data, C10. [*]4'h5 - This value informs the controller that the pipe write sequence to follow is of a 'N' Plane Cache Program with the following sequence, C80, Address, Data, C11, C81, Address, Data, C15.....C80, Address, Data, C11, C81, Address, Data, C10. [*]4'h6 - 4'h15 - Reserved. [/list] ..... indicates that the previous sequence is repeated till the last page.	RW	0x0

copyback_mode

The type of copyback sequence that the controller will follow.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801E0

Offset: 0x1E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0x0			

copyback_mode Fields

Bit	Name	Description	Access	Reset
3:0	value	The values in the field should be as follows[list] [*]4'h0 - This value informs the controller that the copyback sequence to follow is, C00, Address, C35, C85, Address, C10 [*]4'h1 - This value informs the controller that the copyback sequence to follow is, C00, Address, C30, C8C, Address, C10 [*]4'h2 - This value informs the controller that the copyback sequence to follow is, C00, Address, C8A, Address, C10 [*]4'h3 - This value informs the controller that the copyback sequence to follow is of a four plane copyback sequence, C00, Address, C03, Address, C03, Address, C03, Address, C8A, Address, C11, C8A, Address, C11, C8A, Address, C11, C8A, Address, C10. [*]4'h4 - This value informs the controller that the copyback sequence to follow is of a two plane copyback sequence, C00, Address, C35, C00, Address, C35, C85, Address, C11, C81, Address, C10. [*]4'h5 - This value informs the controller that the copyback sequence to follow is of a two plane copyback sequence, C60, Address, C60, Address, C35, C85, Address, C11, C81, Address, C10. [*]4'h6 - This value informs the controller that the copyback sequence to follow is of a two plane copyback sequence, C00, Address, C00, Address, C35, C85, Address, C11, C80, Address, C10. [*]4'h7 - This value informs the controller that the copyback sequence to follow is of a two plane copyback sequence, C60, Address, C60, Address, C30, C8C, Address, C11, C8C, Address, C10. [*]4'h8 - 4'h15 - Reserved.[/list]	RW	0x0

rdwr_en_lo_cnt

Read/Write Enable low pulse width

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB801F0

Offset: 0x1F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											value RW 0x12				

rdwr_en_lo_cnt Fields

Bit	Name	Description	Access	Reset
4:0	value	Number of nand_mp_clk cycles that read or write enable will kept low to meet the min Trp/Twp parameter of the device. The value in this register plus rdwr_en_hi_cnt register value should meet the min cycle time of the device connected. The default value is calculated assuming the max nand_mp_clk time period of 4ns to work with ONFI Mode 0 mode of 100ns device cycle time. This assumes a 1x/4x clocking scheme.	RW	0x12

rdwr_en_hi_cnt

Read/Write Enable high pulse width

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80200

Offset: 0x200

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											value RW 0xC				

rdwr_en_hi_cnt Fields

Bit	Name	Description	Access	Reset
4:0	value	Number of nand_mp_clk cycles that read or write enable will kept high to meet the min Treh/Tweh parameter of the device. The value in this register plus rdwr_en_lo_cnt register value should meet the min cycle time of the device connected. The default value is calculated assuming the max nand_mp_clk time period of 4ns to work with ONFI Mode 0 mode of 100ns device cycle time. This assumes a 1x/4x clocking scheme.	RW	0xC

max_rd_delay

Max round trip read data delay for data capture

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80210

Offset: 0x210

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0x0			

max_rd_delay Fields

Bit	Name	Description	Access	Reset
3:0	value	Number of nand_mp_clk cycles after generation of feedback nand_mp_clk pulse when it is safe to synchronize received data to nand_mp_clk domain. Data should have been registered with nand_mp_clk and stable by the time max_rd_delay cycles has elapsed. A default value of zero will mean a value of nand_mp_clk multiple minus one.	RW	0x0

cs_setup_cnt

Chip select setup time

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80220

Offset: 0x220

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											value RW 0x3				

cs_setup_cnt Fields

Bit	Name	Description	Access	Reset
4:0	value	Number of nand_mp_clk cycles required for meeting chip select setup time. This register refers to device timing parameter Tcs. The value in this registers reflects the extra setup cycles for chip select before read/write enable signal is set low. The default value is calculated for ONFI Timing mode 0 Tcs = 70ns and maximum nand_mp_clk period of 4ns for 1x/4x clock multiple for 16ns cycle time device. Please refer to Figure 3.3 for the relationship between the cs_setup_cnt and rdwr_en_lo_cnt values.	RW	0x3

spare_area_skip_bytes

Spare area skip bytes

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80230

Offset: 0x230

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										value RW 0x0					

spare_area_skip_bytes Fields

Bit	Name	Description	Access	Reset
5:0	value	Number of bytes to skip from start of spare area before last ECC sector data starts. The bytes will be written with the value programmed in the spare_area_marker register. This register could be potentially used to preserve the bad block marker in the spare area by marking it good. The default value is zero which means no bytes will be skipped and last ECC sector will start from the beginning of spare area.	RW	0x0

spare_area_marker

Spare area marker value

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80240

Offset: 0x240

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0xFFFF															

spare_area_marker Fields

Bit	Name	Description	Access	Reset
15:0	value	The value that will be written in the spare area skip bytes. This value will be used by controller while in the MAIN mode of data transfer. Only the least-significant 8 bits of the field value are used.	RW	0xFFFF

devices_connected

Number of Devices connected on one bank

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80250

Offset: 0x250

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													value RW 0x0		

devices_connected Fields

Bit	Name	Description	Access	Reset
2:0	value	Indicates the number of devices connected to a bank. At reset, the value loaded is the maximum possible devices that could be connected in this configuration.	RW	0x0

die_mask

Indicates the die differentiator in case of NAND devices with stacked dies.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80260

Offset: 0x260

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RW 0x0							

die_mask Fields

Bit	Name	Description	Access	Reset
7:0	value	The die_mask register information will be used for devices having address restrictions. For example, in certain Samsung devices, when the first address in a two-plane command is being sent, it is expected that the address is all zeros. But if the NAND device internally has multiple dies stacked, the die information (MSB of final row address) has to be sent. The value programmed in this register will be used to mask the address while sending out the last row address.	RW	0x0

first_block_of_next_plane

The starting block address of the next plane in a multi plane device.

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80270`

Offset: `0x270`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`value` `RW 0x1`															

first_block_of_next_plane Fields

Bit	Name	Description	Access	Reset
15:0	`value`	This values informs the controller of the plane structure of the device. In case the device is a multi plane device and the value here is 1, the controller understands that the next plane starts from Block number 1 and in conjunction with the number of planes parameter can decide upon the distribution of blocks in a plane in the device.	`RW`	`0x1`

write_protect

This register is used to control the assertion/de-assertion of the WP# pin to the device.

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80280`

Offset: `0x280`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`flag` `RW 0x1`

write_protect Fields

Bit	Name	Description	Access	Reset
0	flag	When the controller is in reset, the WP# pin is always asserted to the device. Once the reset is removed, the WP# is de-asserted. The software will then have to come and program this bit to assert/de-assert the same. [list][*]1 - Write protect de-assert [*]0 - Write protect assert[/list]	RW	0x1

re_2_re

Timing parameter between re high to re low (Trhz) for the next bank

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80290

Offset: 0x290

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										value RW 0x32					

re_2_re Fields

Bit	Name	Description	Access	Reset
5:0	value	Signifies the number of bus interface nand_mp_clk clocks that should be introduced between read enable going high to a bank to the read enable going low to the next bank. The number of clocks is the function of device parameter Trhz and controller clock frequency.	RW	0x32

por_reset_count

The number of cycles the controller waits after reset to issue the first RESET command to the device.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB802A0

Offset: 0x2A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x13B															

por_reset_count Fields

Bit	Name	Description	Access	Reset
15:0	value	The controller waits for this number of cycles before issuing the first RESET command to the device. The number in this register is multiplied internally by 16 in the controller to form the final reset wait count.	RW	0x13B

watchdog_reset_count

The number of cycles the controller waits before flagging a watchdog timeout interrupt.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB802B0

Offset: 0x2B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x5B9A															

watchdog_reset_count Fields

Bit	Name	Description	Access	Reset
15:0	value	The controller waits for this number of cycles before issuing a watchdog timeout interrupt. The value in this register is multiplied internally by 32 in the controller to form the final watchdog counter.	RW	0x5B9A

Device parameters Register Descriptions

Controller reads device parameters after initialization and stores in the following registers for software

Offset: 0x300

manufacturer_id on page 13-69

device_id on page 13-70

device_param_0 on page 13-70

device_param_1 on page 13-71

device_param_2 on page 13-71

logical_page_data_size on page 13-72
Logical page data area size in bytes

logical_page_spare_size on page 13-72
Logical page data area size in bytes

revision on page 13-73
Controller revision number

onfi_device_features on page 13-73
Features supported by the connected ONFI device

onfi_optional_commands on page 13-74
Optional commands supported by the connected ONFI device

onfi_timing_mode on page 13-74
Asynchronous Timing modes supported by the connected ONFI device

onfi_pgm_cache_timing_mode on page 13-75
Asynchronous Program Cache Timing modes supported by the connected ONFI device

onfi_device_no_of_luns on page 13-76
Indicates if the device is an ONFI compliant device and the number of LUNS present in the device

onfi_device_no_of_blocks_per_lun_l on page 13-76
Lower bits of number of blocks per LUN present in the ONFI complaint device.

onfi_device_no_of_blocks_per_lun_u on page 13-77
Upper bits of number of blocks per LUN present in the ONFI complaint device.

features on page 13-77
Shows Available hardware features or attributes

manufacturer_id

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80300

Offset: 0x300

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RW 0x0							

manufacturer_id Fields

Bit	Name	Description	Access	Reset
7:0	`value`	Manufacturer ID	RW	0x0

device_id

Module Instance	Base Address	Register Address
`nandregs`	0xFFB80000	0xFFB80310

Offset: `0x310`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

device_id Fields

Bit	Name	Description	Access	Reset
7:0	`value`	Device ID. This register is updated only for Legacy NAND devices.	RO	0x0

device_param_0

Module Instance	Base Address	Register Address
`nandregs`	0xFFB80000	0xFFB80320

Offset: `0x320`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

device_param_0 Fields

Bit	Name	Description	Access	Reset
7:0	value	3rd byte relating to Device Signature. This register is updated only for Legacy NAND devices.	RO	0x0

device_param_1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80330

Offset: 0x330

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

device_param_1 Fields

Bit	Name	Description	Access	Reset
7:0	value	4th byte relating to Device Signature. This register is updated only for Legacy NAND devices.	RO	0x0

device_param_2

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80340

Offset: 0x340

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

device_param_2 Fields

Bit	Name	Description	Access	Reset
7:0	`value`	Reserved.	RO	0x0

logical_page_data_size

Logical page data area size in bytes

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80350`

Offset: `0x350`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

logical_page_data_size Fields

Bit	Name	Description	Access	Reset
15:0	`value`	Logical page spare area size in bytes. If multiple devices are connected on a single chip select, physical page data size will be multiplied by the number of devices to arrive at logical page size.	RO	0x0

logical_page_spare_size

Logical page data area size in bytes

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80360`

Offset: `0x360`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

logical_page_spare_size Fields

Bit	Name	Description	Access	Reset
15:0	`value`	Logical page spare area size in bytes. If multiple devices are connected on a single chip select, physical page spare size will be multiplied by the number of devices to arrive at logical page size.	`RO`	`0x0`

revision

Controller revision number

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80370`

Offset: `0x370`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">value
RO 0x5</td></tr>
</table>

revision Fields

Bit	Name	Description	Access	Reset
15:0	`value`	Controller revision number	`RO`	`0x5`

onfi_device_features

Features supported by the connected ONFI device

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80380`

Offset: `0x380`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">value
RO 0x0</td></tr>
</table>

onfi_device_features Fields

Bit	Name	Description	Access	Reset
15:0	value	The values in the field should be interpreted as follows[list] [*]Bit 0 - Supports 16 bit data bus width. [*]Bit 1 - Supports multiple LUN operations. [*]Bit 2 - Supports non-sequential page programming. [*]Bit 3 - Supports interleaved program and erase operations. [*]Bit 4 - Supports odd to even page copyback. [*]Bit 5 - Supports source synchronous. [*]Bit 6 - Supports interleaved read operations. [*]Bit 7 - Supports extended parameter page. [*]Bit 8 - Supports program page register clear enhancement. [*]Bit 9-15 - Reserved.[/list]	RO	0x0

onfi_optional_commands

Optional commands supported by the connected ONFI device

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80390

Offset: 0x390

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

onfi_optional_commands Fields

Bit	Name	Description	Access	Reset
15:0	value	The values in the field should be interpreted as follows[list] [*]Bit 0 - Supports page cache program command. [*]Bit 1 - Supports read cache commands. [*]Bit 2 - Supports get and set features. [*]Bit 3 - Supports read status enhanced commands. [*]Bit 4 - Supports copyback. [*]Bit 5 - Supports Read Unique Id. [*]Bit 6 - Supports Change Read Column Enhanced. [*]Bit 7 - Supports change row address. [*]Bit 8 - Supports Change small data move. [*]Bit 9 - Supports RESET Lun. [*]Bit 10-15 - Reserved.[/list]	RO	0x0

onfi_timing_mode

Asynchronous Timing modes supported by the connected ONFI device

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803A0

Offset: 0x3A0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										value RO 0x0					

onfi_timing_mode Fields

Bit	Name	Description	Access	Reset
5:0	value	The values in the field should be interpreted as follows[list] [*]Bit 0 - Supports Timing mode 0. [*]Bit 1 - Supports Timing mode 1. [*]Bit 2 - Supports Timing mode 2. [*]Bit 3 - Supports Timing mode 3. [*]Bit 4 - Supports Timing mode 4. [*]Bit 5 - Supports Timing mode 5.[/list]	RO	0x0

onfi_pgm_cache_timing_mode

Asynchronous Program Cache Timing modes supported by the connected ONFI device

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803B0

Offset: 0x3B0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										value RO 0x0					

onfi_pgm_cache_timing_mode Fields

Bit	Name	Description	Access	Reset
5:0	value	The values in the field should be interpreted as follows[list] [*]Bit 0 - Supports Timing mode 0. [*]Bit 1 - Supports Timing mode 1. [*]Bit 2 - Supports Timing mode 2. [*]Bit 3 - Supports Timing mode 3. [*]Bit 4 - Supports Timing mode 4. [*]Bit 5 - Supports Timing mode 5.[/list]	RO	0x0

onfi_device_no_of_luns

Indicates if the device is an ONFI compliant device and the number of LUNS present in the device

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803C0

Offset: 0x3C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							onfi_device RW 0x0	no_of_luns RO 0x0							

onfi_device_no_of_luns Fields

Bit	Name	Description	Access	Reset
8	onfi_device	Indicates if the device is an ONFI compliant device. [list] [*]0 - Non-ONFI compliant device [*]1 - ONFI compliant device[/list]	RW	0x0
7:0	no_of_luns	Indicates the number of LUNS present in the device	RO	0x0

onfi_device_no_of_blocks_per_lun_l

Lower bits of number of blocks per LUN present in the ONFI complaint device.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803D0

Offset: 0x3D0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

onfi_device_no_of_blocks_per_lun_l Fields

Bit	Name	Description	Access	Reset
15:0	value	Indicates the lower bits of number of blocks per LUN present in the ONFI complaint device.	RO	0x0

onfi_device_no_of_blocks_per_lun_u

Upper bits of number of blocks per LUN present in the ONFI complaint device.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803E0

Offset: 0x3E0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

onfi_device_no_of_blocks_per_lun_u Fields

Bit	Name	Description	Access	Reset
15:0	value	Indicates the upper bits of number of blocks per LUN present in the ONFI complaint device.	RO	0x0

features

Shows Available hardware features or attributes

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB803F0

Offset: 0x3F0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		lba RO 0x0	dfi_intf RO 0x0	index_addr RO 0x1	gpreg RO 0x0	xdma_sideband RO 0x0	partition RO 0x0	cmd_dma RO 0x0	dma RO 0x1	Reserved				n_banks RO 0x1	

features Fields

Bit	Name	Description	Access	Reset
13	lba	if set, hardware supports Toshiba LBA devices.	RO	0x0
12	dfi_intf	if set, hardware supports ONFI2.x synchronous interface.	RO	0x0
11	index_addr	if set, hardware support only Indexed addressing.	RO	0x1
10	gpreg	if set, General purpose registers are is present in hardware.	RO	0x0
9	xdma_sideband	if set, Side band DMA signals are present in hardware.	RO	0x0
8	partition	if set, Partition logic is present in hardware.	RO	0x0
7	cmd_dma	Not implemented.	RO	0x0
6	dma	if set, DATA-DMA is present in hardware.	RO	0x1
1:0	n_banks	Maximum number of banks supported by hardware. This is an encoded value. [list][*]0 - Two banks [*]1 - Four banks [*]2 - Eight banks [*]3 - Sixteen banks[/list]	RO	0x1

Interrupt and Status Registers Register Descriptions

Contains interrupt and status registers of controller accessible by software.

Offset: 0x400

transfer_mode on page 13-79
Current data transfer mode is Main only, Spare only or Main+Spare. This information is per bank.

intr_status0 on page 13-80
Interrupt status register for bank 0

intr_en0 on page 13-82
Enables corresponding interrupt bit in interrupt register for bank 0

page_cnt0 on page 13-84
Decrementing page count bank 0

err_page_addr0 on page 13-84
Erred page address bank 0

err_block_addr0 on page 13-85
Erred block address bank 0

intr_status1 on page 13-85
Interrupt status register for bank 1

intr_en1 on page 13-87
Enables corresponding interrupt bit in interrupt register for bank 1

page_cnt1 on page 13-88
Decrementing page count bank 1

err_page_addr1 on page 13-89
Erred page address bank 1

err_block_addr1 on page 13-89
Erred block address bank 1

intr_status2 on page 13-90
Interrupt status register for bank 2

intr_en2 on page 13-91
Enables corresponding interrupt bit in interrupt register for bank 2

page_cnt2 on page 13-93
Decrementing page count bank 2

err_page_addr2 on page 13-93
Erred page address bank 2

err_block_addr2 on page 13-94
Erred block address bank 2

intr_status3 on page 13-94
Interrupt status register for bank 3

intr_en3 on page 13-96
Enables corresponding interrupt bit in interrupt register for bank 3

page_cnt3 on page 13-98
Decrementing page count bank 3

err_page_addr3 on page 13-98
Erred page address bank 3

err_block_addr3 on page 13-99
Erred block address bank 3

transfer_mode

Current data transfer mode is Main only, Spare only or Main+Spare. This information is per bank.

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80400`

Offset: `0x400`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="2">value3
RO 0x0</td><td colspan="2">value2
RO 0x0</td><td colspan="2">value1
RO 0x0</td><td colspan="2">value0
RO 0x0</td></tr>
</table>

transfer_mode Fields

Bit	Name	Description	Access	Reset
7:6	`value3`	[list][*]00 - Bank 3 is in Main mode [*]01 - Bank 3 is in Spare mode [*]10 - Bank 3 is in Main+Spare mode[/list]	RO	0x0
5:4	`value2`	[list][*]00 - Bank 2 is in Main mode [*]01 - Bank 2 is in Spare mode [*]10 - Bank 2 is in Main+Spare mode[/list]	RO	0x0
3:2	`value1`	[list][*]00 - Bank 1 is in Main mode [*]01 - Bank 1 is in Spare mode [*]10 - Bank 1 is in Main+Spare mode[/list]	RO	0x0
1:0	`value0`	[list][*]00 - Bank 0 is in Main mode [*]01 - Bank 0 is in Spare mode [*]10 - Bank 0 is in Main+Spare mode[/list]	RO	0x0

intr_status0

Interrupt status register for bank 0

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80410`

Offset: `0x410`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x0	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_status0 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	Controller has finished reset and initialization process	RW	0x0
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0

Bit	Name	Description	Access	Reset
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	Ecc logic detected uncorrectable error while reading data from flash device.	RW	0x0

intr_en0

Enables corresponding interrupt bit in interrupt register for bank 0

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80420

Offset: 0x420

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x1	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_en0 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	A reset command has completed on this bank	RW	0x1
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	If set, Controller will interrupt processor when Ecc logic detects uncorrectable error.	RW	0x0

page_cnt0

Decrementing page count bank 0

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80430

Offset: 0x430

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

page_cnt0 Fields

Bit	Name	Description	Access	Reset
7:0	value	Maintains a decrementing count of the number of pages in the multi-page (pipeline and copyback) command being executed.	RO	0x0

err_page_addr0

Erred page address bank 0

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80440

Offset: 0x440

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_page_addr0 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the page address that resulted in a failure on program or erase operation.	RO	0x0

err_block_addr0

Erred block address bank 0

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80450

Offset: 0x450

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_block_addr0 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the block address that resulted in a failure on program or erase operation.	RO	0x0

intr_status1

Interrupt status register for bank 1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80460

Offset: 0x460

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x0	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_status1 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	The NAND Flash Memory Controller has completed its reset and initialization process	RW	0x0
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0

Bit	Name	Description	Access	Reset
0	ecc_uncor_err	Ecc logic detected uncorrectable error while reading data from flash device.	RW	0x0

intr_en1

Enables corresponding interrupt bit in interrupt register for bank 1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80470

Offset: 0x470

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x1	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_en1 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	A reset command has completed on this bank	RW	0x1
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0

Bit	Name	Description	Access	Reset
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	If set, Controller will interrupt processor when Ecc logic detects uncorrectable error.	RW	0x0

page_cnt1

Decrementing page count bank 1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80480

Offset: 0x480

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

page_cnt1 Fields

Bit	Name	Description	Access	Reset
7:0	value	Maintains a decrementing count of the number of pages in the multi-page (pipeline and copyback) command being executed.	RO	0x0

err_page_addr1

Erred page address bank 1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80490

Offset: 0x490

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_page_addr1 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the page address that resulted in a failure on program or erase operation.	RO	0x0

err_block_addr1

Erred block address bank 1

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB804A0

Offset: 0x4A0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_block_addr1 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the block address that resulted in a failure on program or erase operation.	RO	0x0

intr_status2

Interrupt status register for bank 2

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB804B0

Offset: 0x4B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x0	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_status2 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	The NAND Flash Memory Controller has completed its reset and initialization process	RW	0x0
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0

Bit	Name	Description	Access	Reset
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	Ecc logic detected uncorrectable error while reading data from flash device.	RW	0x0

intr_en2

Enables corresponding interrupt bit in interrupt register for bank 2

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB804C0

Offset: 0x4C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x1	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_en2 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	A reset command has completed on this bank	RW	0x1
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0

Bit	Name	Description	Access	Reset
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	If set, Controller will interrupt processor when Ecc logic detects uncorrectable error.	RW	0x0

page_cnt2

Decrementing page count bank 2

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB804D0

Offset: 0x4D0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

page_cnt2 Fields

Bit	Name	Description	Access	Reset
7:0	value	Maintains a decrementing count of the number of pages in the multi-page (pipeline and copyback) command being executed.	RO	0x0

err_page_addr2

Erred page address bank 2

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB804E0`

Offset: `0x4E0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_page_addr2 Fields

Bit	Name	Description	Access	Reset
15:0	`value`	Holds the page address that resulted in a failure on program or erase operation.	`RO`	`0x0`

err_block_addr2

Erred block address bank 2

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB804F0`

Offset: `0x4F0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_block_addr2 Fields

Bit	Name	Description	Access	Reset
15:0	`value`	Holds the block address that resulted in a failure on program or erase operation.	`RO`	`0x0`

intr_status3

Interrupt status register for bank 3

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80500

Offset: 0x500

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x0	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_status3 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	The NAND Flash Memory Controller has completed its reset and initialization process	RW	0x0
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0

Bit	Name	Description	Access	Reset
7	`program_comp`	Device finished the last issued program command.	RW	0x0
6	`load_comp`	Device finished the last issued load command.	RW	0x0
5	`erase_fail`	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	`program_fail`	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	`time_out`	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	`dma_cmd_comp`	Not implemented.	RW	0x0
0	`ecc_uncor_err`	Ecc logic detected uncorrectable error while reading data from flash device.	RW	0x0

intr_en3

Enables corresponding interrupt bit in interrupt register for bank 3

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80510

Offset: 0x510

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
page_xfer_inc RW 0x0	pipe_cmd_err RW 0x0	rst_comp RW 0x1	INT_act RW 0x0	unsup_cmd RW 0x0	locked_blk RW 0x0	pipe_cpybck_cmd_comp RW 0x0	erase_comp RW 0x0	program_comp RW 0x0	load_comp RW 0x0	erase_fail RW 0x0	program_fail RW 0x0	time_out RW 0x0	dma_cmd_comp RW 0x0	Reserved	ecc_uncor_err RW 0x0

intr_en3 Fields

Bit	Name	Description	Access	Reset
15	page_xfer_inc	For every page of data transfer to or from the device, this bit will be set.	RW	0x0
14	pipe_cmd_err	A pipeline command sequence has been violated. This occurs when Map 01 page read/write address does not match the corresponding expected address from the pipeline commands issued earlier.	RW	0x0
13	rst_comp	A reset command has completed on this bank	RW	0x1
12	INT_act	R/B pin of device transitioned from low to high	RW	0x0
11	unsup_cmd	An unsupported command was received. This interrupt is set when an invalid command is received, or when a command sequence is broken.	RW	0x0
10	locked_blk	The address to program or erase operation is to a locked block and the operation failed due to this reason	RW	0x0
9	pipe_cpybck_cmd_comp	A pipeline command or a copyback bank command has completed on this particular bank	RW	0x0
8	erase_comp	Device erase operation complete	RW	0x0
7	program_comp	Device finished the last issued program command.	RW	0x0
6	load_comp	Device finished the last issued load command.	RW	0x0
5	erase_fail	Erase failure occurred in the device on issuance of a erase command. err_block_addr and err_page_addr contain the block address and page address that failed erase operation.	RW	0x0
4	program_fail	Program failure occurred in the device on issuance of a program command. err_block_addr and err_page_addr contain the block address and page address that failed program operation.	RW	0x0
3	time_out	Watchdog timer has triggered in the controller due to one of the reasons like device not responding or controller state machine did not get back to idle	RW	0x0
2	dma_cmd_comp	Not implemented.	RW	0x0
0	ecc_uncor_err	If set, Controller will interrupt processor when Ecc logic detects uncorrectable error.	RW	0x0

page_cnt3

Decrementing page count bank 3

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80520

Offset: 0x520

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RO 0x0							

page_cnt3 Fields

Bit	Name	Description	Access	Reset
7:0	value	Maintains a decrementing count of the number of pages in the multi-page (pipeline and copyback) command being executed.	RO	0x0

err_page_addr3

Erred page address bank 3

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80530

Offset: 0x530

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_page_addr3 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the page address that resulted in a failure on program or erase operation.	RO	0x0

err_block_addr3

Erred block address bank 3

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80540

Offset: 0x540

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

err_block_addr3 Fields

Bit	Name	Description	Access	Reset
15:0	value	Holds the block address that resulted in a failure on program or erase operation.	RO	0x0

ECC registers Register Descriptions

Offset: 0x650

ECCCorInfo_b01 on page 13-99
ECC Error correction Information register. Controller updates this register when it completes a transaction. The values are held in this register till a new transaction completes.

ECCCorInfo_b23 on page 13-100
ECC Error correction Information register. Controller updates this register when it completes a transaction. The values are held in this register till a new transaction completes.

ECCCorInfo_b01

ECC Error correction Information register. Controller updates this register when it completes a transaction. The values are held in this register till a new transaction completes.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80650

Offset: 0x650

Access: RO

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>uncor_err_b1
RO 0x0</td><td colspan="7">max_errors_b1
RO 0x0</td><td>uncor_err_b0
RO
0x0</td><td colspan="7">max_errors_b0
RO 0x0</td></tr>
</table>

ECCCorInfo_b01 Fields

Bit	Name	Description	Access	Reset
15	uncor_err_b1	Uncorrectable error occurred while reading pages for last transaction in Bank1. Uncorrectable errors also generate interrupts in intr_statusx register.	RO	0x0
14:8	max_errors_b1	Maximum of number of errors corrected per sector in Bank1. This field is not valid for uncorrectable errors. A value of zero indicates that no ECC error occurred in last completed transaction.	RO	0x0
7	uncor_err_b0	Uncorrectable error occurred while reading pages for last transaction in Bank0. Uncorrectable errors also generate interrupts in intr_statusx register.	RO	0x0
6:0	max_errors_b0	Maximum of number of errors corrected per sector in Bank0. This field is not valid for uncorrectable errors. A value of zero indicates that no ECC error occurred in last completed transaction.	RO	0x0

ECCCorInfo_b23

ECC Error correction Information register. Controller updates this register when it completes a transaction. The values are held in this register till a new transaction completes.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80660

Offset: 0x660

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uncor_err_b3 RO 0x0	max_errors_b3 RO 0x0							uncor_err_b2 RO 0x0	max_errors_b2 RO 0x0						

ECCCorInfo_b23 Fields

Bit	Name	Description	Access	Reset
15	uncor_err_b3	Uncorrectable error occurred while reading pages for last transaction in Bank3. Uncorrectable errors also generate interrupts in intr_statusx register.	RO	0x0
14:8	max_errors_b3	Maximum of number of errors corrected per sector in Bank3. This field is not valid for uncorrectable errors. A value of zero indicates that no ECC error occurred in last completed transaction.	RO	0x0
7	uncor_err_b2	Uncorrectable error occurred while reading pages for last transaction in Bank2. Uncorrectable errors also generate interrupts in intr_statusx register.	RO	0x0
6:0	max_errors_b2	Maximum of number of errors corrected per sector in Bank2. This field is not valid for uncorrectable errors. A value of zero indicates that no ECC error occurred in last completed transaction.	RO	0x0

DMA registers Register Descriptions

Offset: 0x700

dma_enable on page 13-102

dma_intr on page 13-102
DMA interrupt register

dma_intr_en on page 13-103
Enables corresponding interrupt bit in dma interrupt register

target_err_addr_lo on page 13-103
Transaction address for which controller initiator interface received an ERROR target response.

target_err_addr_hi on page 13-104
Transaction address for which controller initiator interface received an ERROR target response.

flash_burst_length on page 13-104

chip_interleave_enable_and_allow_int_reads on page 13-105

no_of_blocks_per_lun on page 13-106

lun_status_cmd on page 13-107
Indicates the command to be sent while checking status of the next LUN.

dma_enable

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80700`

Offset: `0x700`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															flag RW 0x0

dma_enable Fields

Bit	Name	Description	Access	Reset
0	`flag`	Enables data DMA operation in the controller 1 - Enable DMA 0 - Disable DMA	RW	`0x0`

dma_intr

DMA interrupt register

Module Instance	Base Address	Register Address
`nandregs`	`0xFFB80000`	`0xFFB80720`

Offset: `0x720`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															target_error RW 0x0

dma_intr Fields

Bit	Name	Description	Access	Reset
0	target_error	Controller initiator interface received an ERROR target response for a transaction.	RW	0x0

dma_intr_en

Enables corresponding interrupt bit in dma interrupt register

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80730

Offset: 0x730

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															target_error RW 0x0

dma_intr_en Fields

Bit	Name	Description	Access	Reset
0	target_error	Controller initiator interface received an ERROR target response for a transaction.	RW	0x0

target_err_addr_lo

Transaction address for which controller initiator interface received an ERROR target response.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80740

Offset: 0x740

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

target_err_addr_lo Fields

Bit	Name	Description	Access	Reset
15:0	value	Least significant 16 bits	RO	0x0

target_err_addr_hi

Transaction address for which controller initiator interface received an ERROR target response.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80750

Offset: 0x750

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x0															

target_err_addr_hi Fields

Bit	Name	Description	Access	Reset
15:0	value	Most significant 16 bits	RO	0x0

flash_burst_length

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80770

Offset: 0x770

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">reserved
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">reserved
RW 0x0</td><td colspan="3">Reserved</td><td>continous_burst
RW
0x0</td><td colspan="2">Reserved</td><td colspan="2">value
RW 0x1</td></tr>
</table>

flash_burst_length Fields

Bit	Name	Description	Access	Reset
31:8	reserved	Reserved	RW	0x0
4	continous_burst	When this bit is set, the Data DMA will burst the entire page from/to the flash device. Please make sure that the host system can provide/sink data at a fast pace to avoid unnecessary pausing of data on the device interface.	RW	0x0
1:0	value	Sets the burst used by data dma for transferring data to/from flash device. This burst length is different and is larger than the burst length on the host bus so that larger amount of data can be transferred to/from device, descreasing controller data transfer overhead in the process. 00 - 64 bytes, 01 - 128 bytes, 10 - 256 bytes, 11 - 512 bytes. The host burst size multiplied by the number of outstanding requests on the host side should be greater than equal to this value. If not, the device side burst length will be equal to host side burst length.	RW	0x1

chip_interleave_enable_and_allow_int_reads

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80780

Offset: 0x780

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											allow_int_reads_within_luns RW 0x1	Reserved			chip_interleave_enable RW 0x0

chip_interleave_enable_and_allow_int_reads Fields

Bit	Name	Description	Access	Reset
4	allow_int_reads_ within_luns	This bit informs the controller to enable or disable simultaneous read accesses to different LUNS in the same bank. This bit is of importance only if the controller supports interleaved operations among LUNs and if the device has multiple LUNS. If the bit is disabled, the controller will send read commands to different LUNS of of the same bank only sequentially and if enabled, the controller will issue simultaneous read accesses to LUNS of same bank if required. [list][*]1 - Enable [*]0 - Disable[/list]	RW	0x1
0	chip_interleave_ enable	This bit informs the controller to enable or disable interleaving among banks/LUNS to increase the net performance of the controller. [list][*]1 - Enable interleaving [*]0 - Disable Interleaving[/list]	RW	0x0

no_of_blocks_per_lun

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB80790

Offset: 0x790

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												value RW 0xF			

no_of_blocks_per_lun Fields

Bit	Name	Description	Access	Reset
3:0	value	Indicates the first block of next LUN. This information is used for extracting the target LUN during LUN interleaving. After Initialization, if the controller detects an ONFi device, this field is automatically updated by the controller. For other devices, software will need to write to this register for proper interleaving. The value in this register is interpreted as follows- [list][*]0 - Next LUN starts from 1024. [*]1 - Next LUN starts from 2048. [*]2 - Next LUN starts from 4096 and so on... [/list]	RW	0xF

lun_status_cmd

Indicates the command to be sent while checking status of the next LUN.

Module Instance	Base Address	Register Address
nandregs	0xFFB80000	0xFFB807A0

Offset: 0x7A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x7878															

lun_status_cmd Fields

Bit	Name	Description	Access	Reset
15:0	value	[list][*]7:0 - Indicates the command to check the status of the first LUN/Die. [*]15:8 - Indicates the command to check the status of the other LUN/Die.[/list]	RW	0x7878

Document Revision History

Table 13-19: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Added address map and register definitions. • Removed Command DMA section.
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.2	• Supports one 8-bit device • Show additional supported block sizes • Bad block marker handling
May 2012	1.1	Added programming model section.
January 2012	1.0	Initial release.

SD/MMC Controller 14

2014.06.30

cv_54011 Subscribe Send Feedback

The hard processor system (HPS) provides a Secure Digital/Multimedia Card (SD/MMC) controller for interfacing to external SD and MMC flash cards, secure digital I/O (SDIO) devices, and Consumer Electronics Advanced Transport Architecture (CE-ATA) hard drives. The SD/MMC controller enables you to store boot images and boot the processor system from the removable flash card. You can also use the flash card to expand the on-board storage capacity for larger applications or user data. Other applications include interfacing to embedded SD (eSD) and embedded MMC (eMMC) nonremovable flash devices.

The SD/MMC controller is based on the Synopsys DesignWare Mobile Storage Host (DWC_mobile_storage) controller.

This document refers to SD/SDIO commands, which are documented in detail in the *Physical Layer Simplified Specification, Version 3.01* and the *SDIO Simplified Specification Version 2.00* described on the SD Association website.

Related Information

- **www.sdcard.org**
  To learn more about how SD technology works, visit the SD Association website.
- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the Introduction to the Hard Processor System chapter.

Features of the SD/MMC Controller

The HPS SD/MMC controller offers the following features:

- Supports HPS boot from mobile storage
- Supports the following standards or card types:
  - SD, including eSD—version 3.0
  - SDIO, including embedded SDIO (eSDIO)—version 3.0
  - CE-ATA—version 1.1
  - MMC, including eMMC — version 4.41, 1-bit, 4-bit, and 8-bit (in some packages, as described in *MMC Support Matrix*)[(†)]
- Integrated descriptor-based direct memory access (DMA)
- Internal 4 KB receive and transmit FIFO buffer

The SD/MMC controller does not directly support voltage switching, card interrupts, or back-end power control of eSDIO card devices. However, you can connect these signals to general-purpose I/Os (GPIOs).

The SD/MMC controller does not contain a reset output as part of the external card interface. To reset the flash card device, consider using a general purpose output pin.

Related Information

MMC Support Matrix on page 14-3
For more information on what is supported, go to the MMC Support Matrix table.

SD Card Support Matrix

Table 14-1: SD Card Support Matrix

Device Card Type	Voltages Supported		Bus Modes Supported				Bus Speed Modes Supported			
							Default Speed	High Speed	SDR12	SDR25[33]
	3.3 V	1.8 V[34]	SPI	1 bit	4 bit	8 bit	12.5 MBps 25 MHz	25 MBps 50 MHz	12.5 MBps 25 MHz	25 MBps 50 MHz
SDSC (SD)	√		√	√			√	√		
SDHC	√	√	√	√	√		√	√	√	√
SDXC	√	√	√	√	√		√	√	√	√
eSD	√	√	√	√	√		√	√	√	√
SDIO	√	√	√	√	√		√	√	√	√
eSDIO	√	√	√	√	√	√	√	√	√	√

Note: Card form factors (such as mini and micro) are not enumerated in the above table because they do not impact the card interface functionality.

[†] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys and DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non infringement, and any warranties arising out of a course of dealing or usage of trade.

† Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

[33] SDR25 speed mode requires 1.8-V signaling. Note that even if a card supports UHS-I modes (for example SDR50, SDR104, DDR50) it can still communicate at the lower speeds (for example SDR12, SDR25).

[34] Where supported, controls the voltage switch output to support 1.8-V signalling for SD.

MMC Support Matrix

Table 14-2: MMC Support Matrix

Card Device Type	Max Clock Speed (MHz)	Max Data Rate (MBps)	Voltages Supported		Bus Modes Supported				Bus Speed Modes Supported	
			3.3 V	1.8 V	SPI[35]	1 bit	4 bit	8 bit	Default Speed	High Speed
MMC	20	2.5	√		√	√			√	
RSMMC	20	10	√		√	√	√		√	√
MMCPlus	50[36]	25	√			√	√	√	√	√
MMCMobile	50	6.5	√	√		√			√	√
eMMC	50	25	√	√		√	√	√	√	√

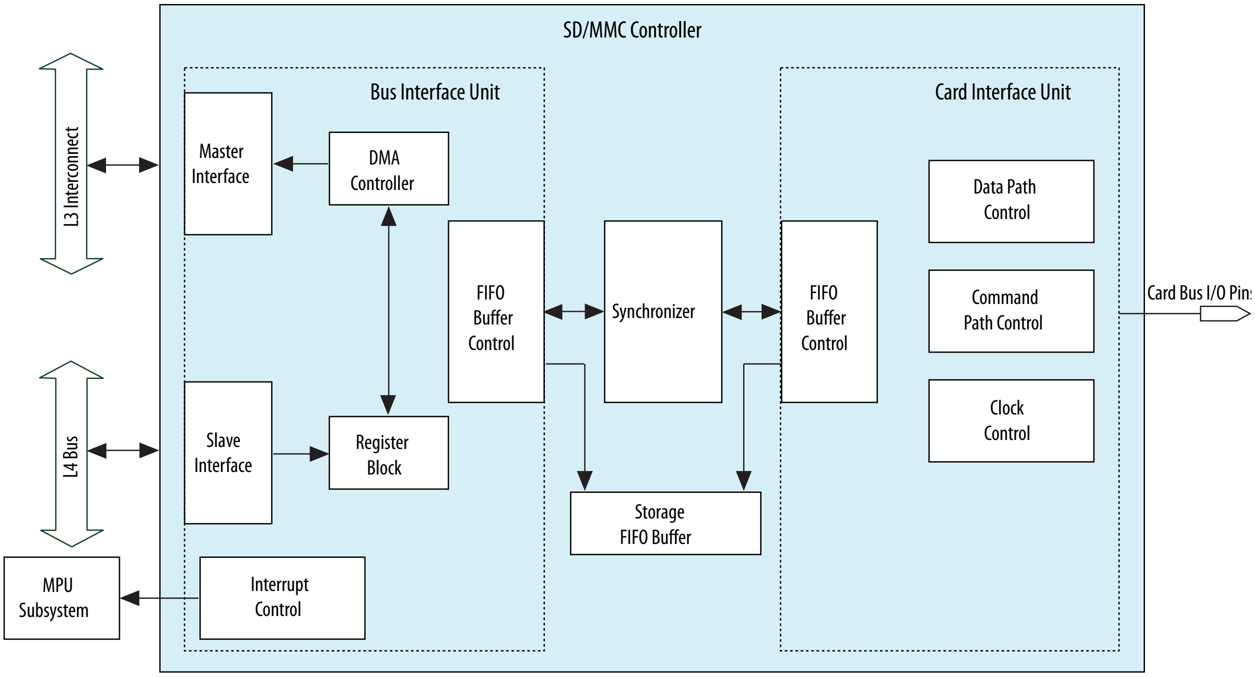
SD/MMC Controller Block Diagram and System Integration

The SD/MMC controller includes a bus interface unit (BIU) and a card interface unit (CIU). The BIU provides a slave interface for a host to access the control and status registers (CSRs). Additionally, this unit also provides independent FIFO buffer access through a DMA interface. The DMA controller is responsible for exchanging data between the system memory and FIFO buffer. The DMA registers are accessible by the host to control the DMA operation. The CIU supports the SD, MMC, and CE-ATA protocols on the controller, and provides clock management through the clock control block. The interrupt control block for generating an interrupt connects to the generic interrupt controller in the ARM Cortex-A9 microprocessor unit (MPU) subsystem.

[35] SPI mode is obsolete in the MMC 4.41 specification.

[36] Supports a maximum clock rate of 50 MHz instead of 52 MHz (specified in MMC specification).

Figure 14-1: SD/MMC Controller Connectivity



Functional Description of the SD/MMC Controller

This section describes the SD/MMC controller components and how the controller operates.

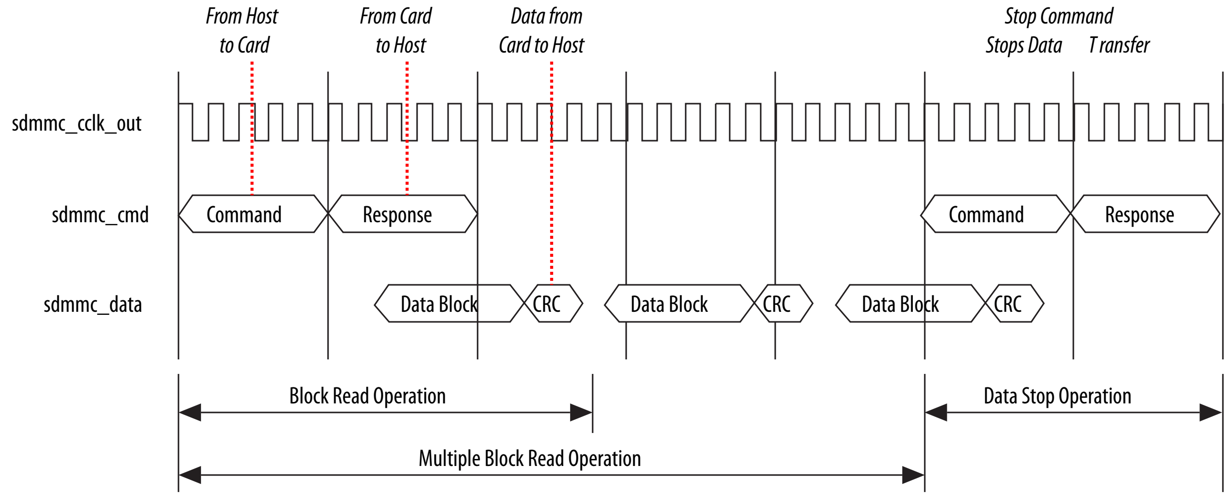
SD/MMC/CE-ATA Protocol

The SD/MMC/CE-ATA protocol is based on command and data bit streams that are initiated by a start bit and terminated by a stop bit. Additionally, the SD/MMC controller provides a reference clock and is the only master interface that can initiate a transaction. †

- Command—a token transmitted serially on the `CMD` pin that starts an operation.†
- Response—a token from the card transmitted serially on the `CMD` pin in response to certain commands.†
- Data—transferred serially using the data pins for data movement commands.†

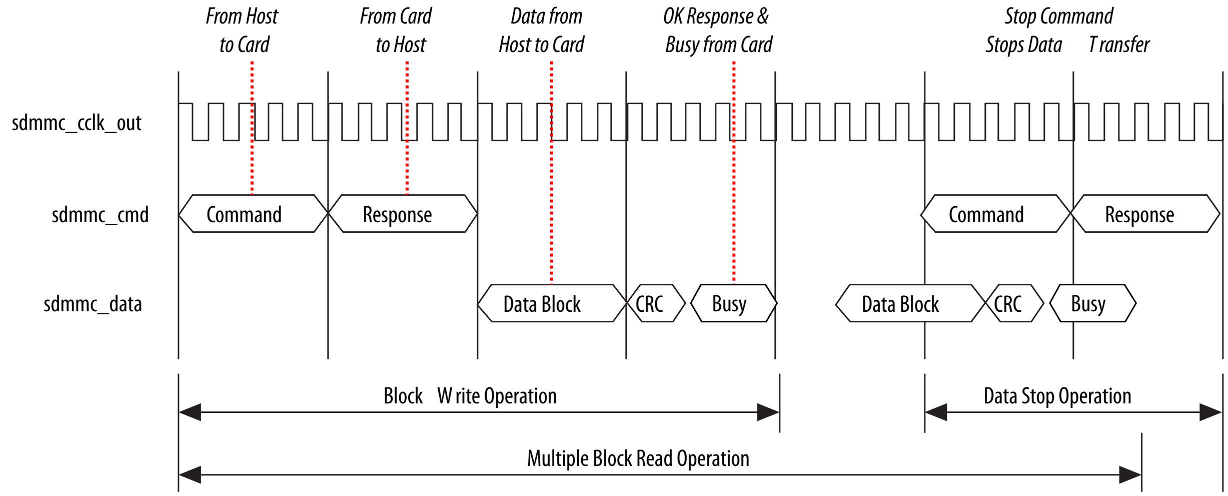
In the following figure, the clock is a representative only and does not show the exact number of clock cycles.†

Figure 14-2: Multiple-Block Read Operation†



The following figure illustrates an example of a command token sent by the host in a multiple-block write operation.

Figure 14-3: Multiple-Block Write Operation†



BIU

The BIU interfaces with the CIU, and is connected to the level 3 (L3) interconnect and level 4 (L4) peripheral buses. The BIU consists of the following primary functional blocks, which are defined in the following sections:

- Slave interface
- Register block
- FIFO buffer
- Interrupt control
- Internal DMA controller

Slave Interface

The host processor accesses the SD/MMC controller registers and data FIFO buffers through the slave interface.

Register Block

The register block is part of the BIU and provides read and write access to the CSRs.†

All registers reside in the BIU clock domain. When a command is sent to a card by setting the start command bit (start_cmd) of the command register (cmd) to 1, all relevant registers needed for the CIU operation are transferred to the CIU block. During this time, software must not write to the registers that are transferred from the BIU to the CIU. The software must wait for the hardware to reset the start_cmd bit to 0 before writing to these registers again. The register unit has a hardware locking feature to prevent illegal writes to registers.†

Registers Locked Out Pending Command Acceptance

After a command start is issued by setting the start_cmd bit of the cmd register, the following registers cannot be rewritten until the command is accepted by the CIU:†

- Command (cmd)†
- Command argument (cmdarg)†
- Byte count (bytcnt)†
- Block size (blksiz)†
- Clock divider (clkdiv)†
- Clock enable (clkena)†
- Clock source (clksrc)†
- Timeout (tmout)†
- Card type (ctype)†

The hardware resets the start_cmd bit after the CIU accepts the command. If a host write to any of these registers is attempted during this locked time, the write is ignored and the hardware lock write error bit (hle) is set to 1 in the raw interrupt status register (rintsts). Additionally, if the interrupt is enabled and not masked for a hardware lock error, an interrupt is sent to the host.†

After a command is accepted, you can send another command to the CIU—which has a one-deep command queue—under the following conditions:†

- If the previous command is not a data transfer command, the new command is sent to the SD/MMC/CE-ATA card once the previous command completes.†
- If the previous command is a data transfer command and if the wait previous data complete bit (wait_prvdata_complete) of the cmd register is set to 1 for the new command, the new command is sent to the SD/MMC/CE-ATA card only when the data transfer completes.†
- If the wait_prvdata_complete bit is 0, the new command is sent to the SD/MMC/CE-ATA card as soon as the previous command is sent. Typically, use this feature to stop or abort a previous data transfer or query the card status in the middle of a data transfer.†

Interrupt Controller Unit

The interrupt controller unit generates an interrupt that depends on the rintsts register, the interrupt mask register (intmask), and the interrupt enable bit (int_enable) of the control register (ctrl). Once an interrupt condition is detected, the controller sets the corresponding interrupt bit in the rintsts register. The bit in the rintsts register remains set to 1 until the software resets the bit to 0 by writing a 1 to the interrupt bit; writing a 0 leaves the bit untouched.

The interrupt port is an active-high, level-sensitive interrupt. The interrupt port is active only when at least one bit in the `rintsts` register is set to 1, the corresponding `intmask` register bit is 1, and the `int_enable` bit of the `ctrl` register is 1.

The following conditions can cause the interrupt to occur:

- SDIO card interrupts
- End-bit error (EBE) on read/No cyclic redundancy code (CRC) on write†
- Auto command done (ACD)†
- Start-bit error (SBE)†
- Hardware locked write error (HLE)†
- FIFO buffer underflow or overflow error (FRUN)†
- Data starvation by host timeout (HTO)†
- Data read timeout (DRTO) or boot data start (BDS)†
- Response timeout (RTO) or boot ACK received (BAR)†
- Data CRC error (DCRC)†
- Response CRC error (RCRC)†
- Receive FIFO buffer data request (RXDR)†
- Transmit FIFO buffer data request (TXDR)†
- Data transfer over (DTO)†
- Command done (CD)†
- Response error (RE)†
- Card-Detect (CDT)†

The `int_enable` bit of the `ctrl` register is set to 0 on power-on, and the `intmask` register bits are set to 0x0000000, which masks all the interrupts.†

Table 14-3: Interrupt Status Register Bits†

Bits	Interrupt	Description
31:16	SDIO Interrupts†	Interrupts from SDIO cards.† One bit for each card. Bit[31] corresponds to Card[15], and bit[16] corresponds to Card[0]. †
15	End Bit Error (read)/Write no CRC (EBE)†	Error in end-bit during read operation, or no data CRC received during write operation.† **Note:** For MMC CMD19, there may be no CRC status returned by the card. Hence, EBE is set for CMD19. The application should not treat this as an error. †

Bits	Interrupt	Description
14	Auto Command Done (ACD)†	Stop/abort commands automatically sent by card unit and not initiated by host; similar to Command Done (CD) interrupt. † **Recommendation:** Software typically need not enable this for non CE-ATA accesses; Data Transfer Over (DTO) interrupt that comes after this interrupt determines whether data transfer has correctly competed. For CE-ATA accesses, if the software sets `send_auto_stop_ccsd` bit in the control register, then software should enable this bit.†
13	Start Bit Error (SBE)†	Error in data start bit when data is read from a card. In 4-bit mode, if all data bits do not have start bit, then this error is set. †
12	Hardware Locked write Error (HLE)†	During hardware-lock period, write attempted to one of locked registers. †
11	FIFO Underrun/Overrun Error (FRUN)†	Host tried to push data when FIFO was full, or host tried to read data when FIFO was empty. Typically this should not happen, except due to error in software. † Card unit never pushes data into FIFO when FIFO is full, and pop data when FIFO is empty. † If IDMAC (Internal Direct Memory Access Controller) is enabled, FIFO underrun/overrun can occur due to a programming error on MSIZE and watermark values in FIFOTH register; for more information, refer to *Internal Direct Memory Access Controller (IDMAC)* section in the "Synopsys DesignWare Cores Mobile Storage Host Databook".†

Bits	Interrupt	Description
10	Data Starvation by Host Timeout (HTO)[†]	To avoid data loss, card clock out (`cclk_out`) is stopped if FIFO is empty when writing to card, or FIFO is full when reading from card. Whenever card clock is stopped to avoid data loss, data-starvation timeout counter is started with data-timeout value. This interrupt is set if host does not fill data into FIFO during write to card, or does not read from FIFO during read from card before timeout period. [†] Even after timeout, card clock stays in stopped state, with CIU state machines waiting. It is responsibility of host to push or pop data into FIFO upon interrupt, which automatically restarts `cclk_out` and card state machines. [†] Even if host wants to send stop/abort command, it still must ensure to push or pop FIFO so that clock starts in order for stop/abort command to send on cmd signal along with data that is sent or received on data line. [†]
9	Data Read Timeout (DRTO)/Boot Data Start (BDS)[†]	• In Normal functioning mode: Data read timeout (DRTO) Data timeout occurred. Data Transfer Over (DTO) also set if data timeout occurs. [†] • In Boot Mode: Boot Data Start (BDS) When set, indicates that DWC_mobile_storage has started to receive boot data from the card. A write to this register with a value of 1 clears this interrupt.[†]
8	Response Timeout (RTO)/ Boot Ack Received (BAR)[†]	• In Normal functioning mode: Response timeout (RTO) Response timeout occurred. Command Done (CD) also set if response timeout occurs. If command involves data transfer and when response times out, no data transfer is attempted by DWC_mobile_storage.[†] • In Boot Mode: Boot Ack Received (BAR) When expect_boot_ack is set, on reception of a boot acknowledge pattern—0-1-0—this interrupt is asserted. A write to this register with a value of 1 clears this interrupt.[†]
7	Data CRC Error (DCRC)[†]	Received Data CRC does not match with locally-generated CRC in CIU; expected when a negative CRC is received. [†]

Bits	Interrupt	Description
6	Response CRC Error (RCRC)†	Response CRC does not match with locally-generated CRC in CIU.†
5	Receive FIFO Data Request (RXDR)†	Interrupt set during read operation from card when FIFO level is greater than Receive-Threshold level.† **Recommendation:** In DMA modes, this interrupt should not be enabled.† ISR, in non-DMA mode: `pop RX_WMark + 1 data from FIFO` †

4	Transmit FIFO Data Request (TXDR)†	Interrupt set during write operation to card when FIFO level reaches less than or equal to Transmit-Threshold level.†
		Recommendation: In DMA modes, this interrupt should not be enabled.†
		ISR in non-DMA mode: †
		`if (pending_bytes > \†`
		`(FIFO_DEPTH - TX_WMark))†`
		`     push (FIFO_DEPTH - \†`
		`     TX_WMark) data into FIFO†`
		`else†`
		`     push pending_bytes data \†`
		`     into FIFO†`

3	Data Transfer (DTO)†	Data transfer completed, even if there is Start Bit Error or CRC error. This bit is also set when “read data-timeout” occurs or CCS is sampled from CE-ATA device.† **Recommendation:** In non-DMA mode, when data is read from card, on seeing interrupt, host should read any pending data from FIFO. In DMA mode, DMA controllers guarantee FIFO is flushed before interrupt.† **Note:** DTO bit is set at the end of the last data block, even if the device asserts MMC busy after the last data block.†
2	Command Done (CD)†	Command sent to card and received response from card, even if Response Error or CRC error occurs. Also set when response timeout occurs or CCSD sent to CE-ATA device.†

Bits	Interrupt	Description
1	Response Error (RE)†	Error in received response set if one of following occurs:† • Transmission bit != 0† • Command index mismatch† • End-bit != 1†
0	Card-Detect (CDT)†	When one or more cards inserted or removed, this interrupt occurs. Software should read card-detect register (CDETECT, 0x50) to determine current card status.† **Recommendation:** After power-on and before enabling interrupts, software should read card detect register and store it in memory. When interrupt occurs, it should read card detect register and compare it with value stored in memory to determine which card(s) were removed/inserted. Before exiting ISR, software should update memory with new card-detect value.†

Interrupt Setting and Clearing

The SDIO Interrupts, Receive FIFO Data Request, and Transmit FIFO Data Request interrupts are set by level-sensitive interrupt sources. Therefore, the interrupt source must be first cleared before you can reset the interrupt's corresponding bit in the `rintsts` register to 0.†

For example, on receiving the Receive FIFO Data Request interrupt, the FIFO buffer must be emptied so that the FIFO buffer count is not greater than the RX watermark, which causes the interrupt to be triggered.†

The rest of the interrupts are triggered by single clock-pulse-width sources.†

FIFO Buffer

The SD/MMC controller has a 4 KB data FIFO buffer for storing transmit and receive data. The FIFO buffer memory supports error correction codes (ECCs). Both interfaces to the FIFO buffer support single and double bit error injection. The enable and error injection pins are inputs driven by the system manager and the status pins are outputs driven to the MPU subsystem.

The SD/MMC controller provides outputs to notify the system manager when single-bit correctable errors are detected (and corrected), and when double-bit (uncorrectable) errors are detected. The system manager generates an interrupt to the GIC when an ECC error is detected.

Related Information

System Manager on page 5-1

Internal DMA Controller

The internal DMA controller has a CSR and a single transmit or receive engine, which transfers data from system memory to the card and vice versa. The controller uses a descriptor mechanism to efficiently move data from source to destination with minimal host processor intervention. You can set up the controller to interrupt the host processor in situations such as transmit and receive data transfer completion from the card, as well as other normal or error conditions. The DMA controller and the host driver communicate through a single data structure.†

The internal DMA controller transfers the data received from the card to the data buffer in the system memory, and transfers transmit data from the data buffer in the memory to the controller's FIFO buffer. Descriptors that reside in the system memory act as pointers to these buffers.†

A data buffer resides in the physical memory space of the system memory and consists of complete or partial data. The buffer status is maintained in the descriptor. Data chaining refers to data that spans multiple data buffers. However, a single descriptor cannot span multiple data buffers.†

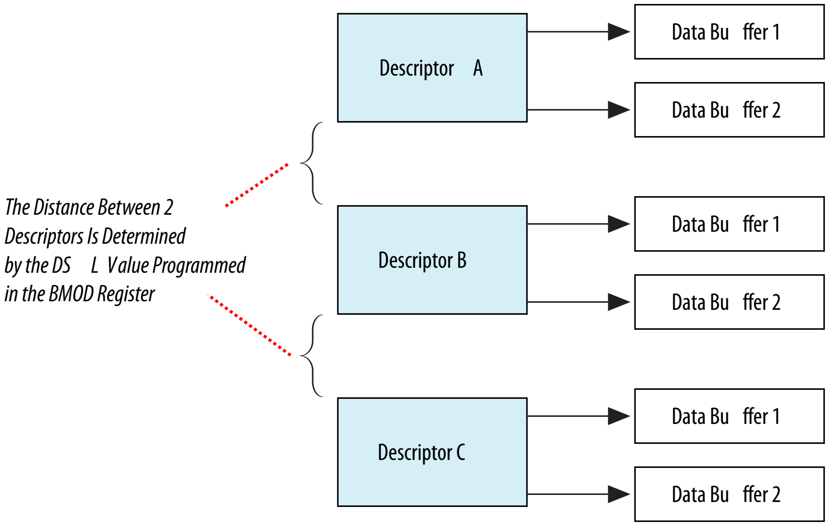
A single descriptor is used for both reception and transmission. The base address of the list is written into the descriptor list base address register (`dbaddr`). A descriptor list is forward linked. The last descriptor can point back to the first entry to create a ring structure. The descriptor list resides in the physical memory address space of the host. Each descriptor can point to a maximum of two data buffers.†

Internal DMA Controller Descriptors

The internal DMA controller uses these types of descriptor structures:†

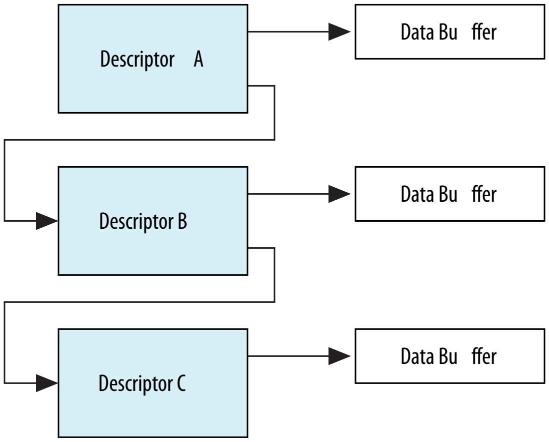
- Dual-buffer structure—The distance between two descriptors is determined by the skip length value written to the descriptor skip length field (`dsl`) of the bus mode register (`bmod`).†

Figure 14-4: Dual-Buffer Descriptor Structure†



- Chain structure—Each descriptor points to a unique buffer, and to the next descriptor in a linked list.†

Figure 14-5: Chain Descriptor Structure†



Internal DMA Controller Descriptor Address

The descriptor address must be aligned to the 32-bit bus. Each descriptor contains 16 bytes of control and status information.†

Table 14-4: Descriptor Format

Name	Offset	31	30	29 ...	26	25 ...	13	12 ...	6	5	4	3	2	1	0
DES 0	0	OWN	CES	—						ER	CH	FS	LD	DIC	—
DES 1	4	—				BS2			BS1						
DES 2	8	BAP1													
DES 3	12	BAP2 or Next Descriptor Address													

Related Information

Internal DMA Controller Descriptor Fields on page 14-13
Refer to this table for information about each of the bits of the descriptor.

Internal DMA Controller Descriptor Fields

The DES0 field in the internal DMA controller descriptor contains control and status information.

Table 14-5: Internal DMA Controller DES0 Descriptor Field†

Bits	Name	Description
31	OWN	When set to 1, this bit indicates that the descriptor is owned by the internal DMA controller. When this bit is set to 0, it indicates that the descriptor is owned by the host. The internal DMA controller resets this bit to 0 when it completes the data transfer.
30	Card Error Summary (CES)	The CES bit indicates whether a transaction error occurred. The CES bit is the logical OR of the following error bits in the `rintsts` register. • End-bit error (`ebe`) • Response timeout (`rto`) • Response CRC (`rcrc`) • Start-bit error (`sbe`) • Data read timeout (`drto`) • Data CRC for receive (`dcrc`) • Response error (`re`)
29:6	Reserved	—
5	End of Ring (ER)	When set to 1, this bit indicates that the descriptor list reached its final descriptor. The internal DMA controller returns to the base address of the list, creating a descriptor ring. ER is meaningful for only a dual-buffer descriptor structure.
4	Second Address Chained (CH)	When set to 1, this bit indicates that the second address in the descriptor is the next descriptor address rather than the second buffer address. When this bit is set to 1, BS2 (DES1[25:13]) must be all zeros.
3	First Descriptor (FS)	When set to 1, this bit indicates that this descriptor contains the first buffer of the data. If the size of the first buffer is 0, next descriptor contains the beginning of the data.
2	Last Descriptor (LD)	When set to 1, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the data.
1	Disable Interrupt on Completion (DIC)	When set to 1, this bit prevents the setting of the TI/RI bit of the internal DMA controller status register (`idsts`) for the data that ends in the buffer pointed to by this descriptor.
0	Reserved	—

The DES1 descriptor field contains the buffer size.

Table 14-6: Internal DMA Controller DES1 Descriptor Field†

Bits	Name	Description
31:26	Reserved	—
25:13	Buffer 2 Size (BS2)	These bits indicate the second data buffer byte size. The buffer size must be a multiple of four. When the buffer size is not a multiple of four, the resulting behavior is undefined. This field is not valid if DES0[4] is set to 1.
12:0	Buffer 1 Size (BS1)	Indicates the data buffer byte size, which must be a multiple of four bytes. When the buffer size is not a multiple of four, the resulting behavior is undefined. If this field is 0, the DMA ignores the buffer and proceeds to the next descriptor for a chain structure, or to the next buffer for a dual-buffer structure. If there is only one descriptor and only one buffer to be programmed, you need to use only buffer 1 and not buffer 2.

The DES2 descriptor field contains the address pointer to the data buffer.

Table 14-7: Internal DMA Controller DES2 Descriptor Field†

Bits	Name	Description
31:0	Buffer Address Pointer 1 (BAP1)	These bits indicate the physical address of the first data buffer. The internal DMA controller ignores DES2 [1:0], because it only performs 32-bit aligned accesses.

The DES3 descriptor field contains the address pointer to the next descriptor if the present descriptor is not the last descriptor in a chained descriptor structure or the second buffer address for a dual-buffer structure.†

Table 14-8: Internal DMA Controller DES3 Descriptor Field†

Bits	Name	Description
31:0	Buffer Address Pointer 2 (BAP2) or Next Descriptor Address	These bits indicate the physical address of the second buffer when the dual-buffer structure is used. If the Second Address Chained (DES0[4]) bit is set to 1, this address contains the pointer to the physical memory where the next descriptor is present. If this is not the last descriptor, the next descriptor address pointer must be aligned to 32 bits. Bits 1 and 0 are ignored.

Host Bus Burst Access

The internal DMA controller attempts to issue fixed-length burst transfers on the master interface if configured using the fixed burst bit (fb) of the bmod register. The maximum burst length is indicated and limited by the programmable burst length (pbl) field of the bmod register. When descriptors are being fetched, the master interface always presents a burst size of four to the interconnect.†

The internal DMA controller initiates a data transfer only when sufficient space to accommodate the configured burst is available in the FIFO buffer or the number of bytes to the end of transfer is less than the configured burst-length. When the DMA master interface is configured for fixed-length bursts, it transfers data using the most efficient combination of INCR4/8/16 and SINGLE transactions. If the DMA master interface is not configured for fixed length bursts, it transfers data using INCR (undefined length) and SINGLE transactions.†

Host Data Buffer Alignment

The transmit and receive data buffers in system memory must be aligned to a 32-bit boundary.

Buffer Size Calculations

The driver knows the amount of data to transmit or receive. For transmitting to the card, the internal DMA controller transfers the exact number of bytes from the FIFO buffer, indicated by the buffer size field of the DES1 descriptor field.†

If a descriptor is not marked as last (with the LD bit of the DES0 field set to 0) then the corresponding buffer(s) of the descriptor are considered full, and the amount of valid data in a buffer is accurately indicated by its buffer size field. If a descriptor is marked as last, the buffer might or might not be full, as indicated by the buffer size in the DES1 field. The driver is aware of the number of locations that are valid.† The driver is expected to ignore the remaining, invalid bytes.

Internal DMA Controller Interrupts

Interrupts can be generated as a result of various events. The idsts register contains all the bits that might cause an interrupt. The internal DMA controller interrupt enable register (idinten) contains an enable bit for each of the events that can cause an interrupt to occur.†

There are two summary interrupts—the normal interrupt summary bit (nis) and the abnormal interrupt summary bit (ais)—in the idsts register.† The nis bit results from a logical OR of the transmit interrupt (ti) and receive interrupt (ri) bits in the idsts register. The ais bit is a logical OR result of the fatal bus error interrupt (fbe), descriptor unavailable interrupt (du), and card error summary interrupt (ces) bits in the idsts register.

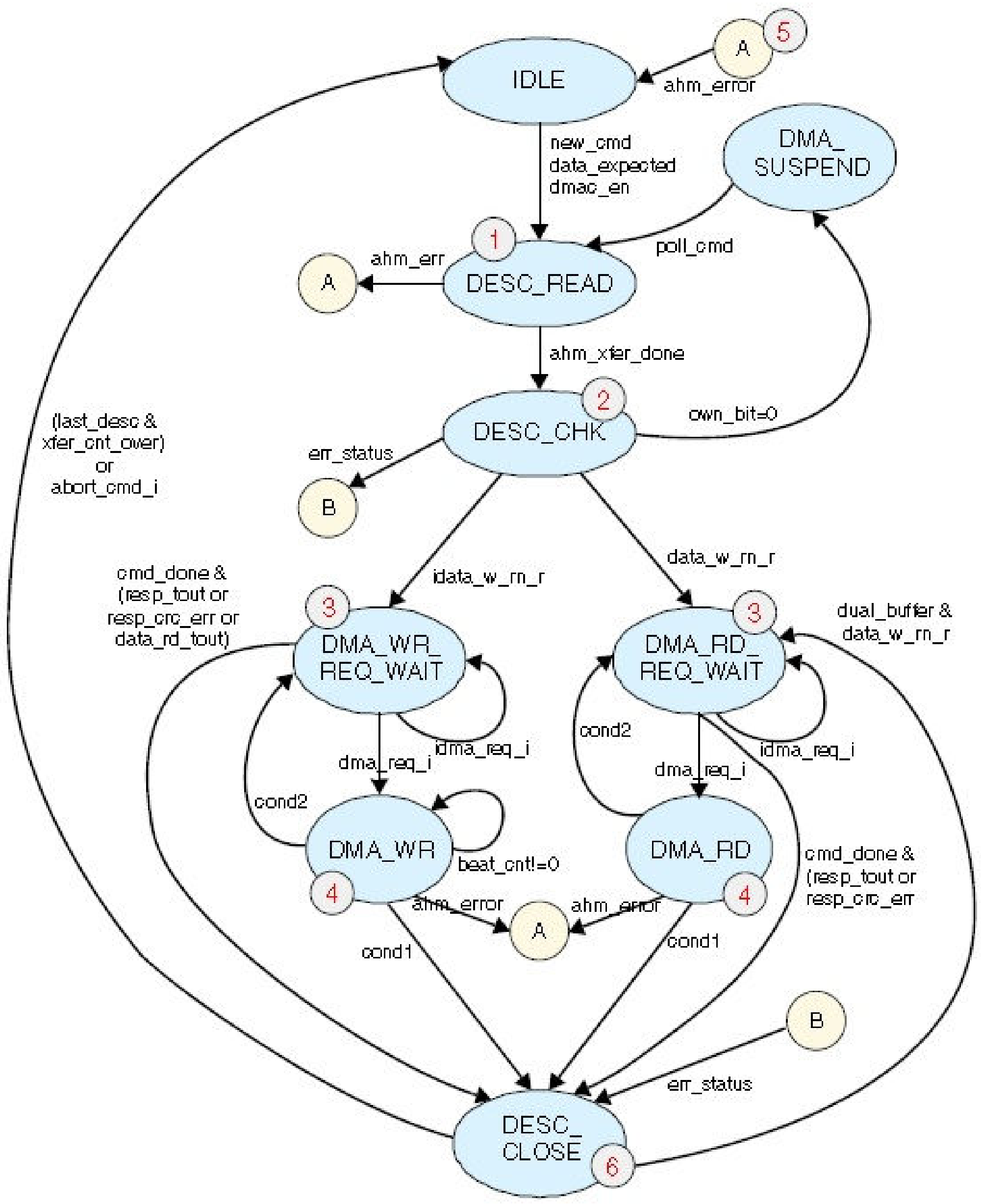
Interrupts are cleared by writing a 1 to the corresponding bit position.† If a 0 is written to an interrupt's bit position, the write is ignored, and does not clear the interrupt. When all the enabled interrupts within a group are cleared, the corresponding summary bit is set to 0. When both the summary bits are set to 0, the interrupt signal is de-asserted.†

Interrupts are not queued. If another interrupt event occurs before the driver has responded to the previous interrupt, no additional interrupts are generated. For example, the ri bit of the idsts register indicates that one or more data has been transferred to the host buffer.†

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the idsts register for the interrupt cause.† The final interrupt signal from the controller is a logical OR of the interrupts from the BIU and internal DMA controller.

Internal DMA Controller FSM

Figure 14-6: Internal DMA Controller Functional State Machine[†]



The following list explains each state of the functional state machine:[†]

1. The internal DMA controller performs four accesses to fetch a descriptor.[†]
2. The DMA controller stores the descriptor information internally. If it is the first descriptor, the controller issues a FIFO buffer reset and waits until the reset is complete.[†]
3. The internal DMA controller checks each bit of the descriptor for the correctness. If bit mismatches are found, the appropriate error bit is set to 1 and the descriptor is closed by setting the OWN bit in the DES0 field to 1.[†]

   The `rintsts` register indicates one of the following conditions:[†]

- Response timeout†
- Response CRC error†
- Data receive timeout†
- Response error†

4. The DMA waits for the RX watermark to be reached before writing data to system memory, or the TX watermark to be reached before reading data from system memory. The RX watermark represents the number of bytes to be locally stored in the FIFO buffer before the DMA writes to memory. The TX watermark represents the number of free bytes in the local FIFO buffer before the DMA reads data from memory.†
5. If the value of the programmable burst length (PBL) field is larger than the remaining amount of data in the buffer, single transfers are initiated. If dual buffers are being used, and the second buffer contains no data (buffer size = 0), the buffer is skipped and the descriptor is closed.†
6. The OWN bit in descriptor is set to 0 by the internal DMA controller after the data transfer for one descriptor is completed. If the transfer spans more than one descriptor, the DMA controller fetches the next descriptor. If the transfer ends with the current descriptor, the internal DMA controller goes to idle state after setting the `ri` bit or the `ti` bit of the `idsts` register. Depending on the descriptor structure (dual buffer or chained), the appropriate starting address of descriptor is loaded. If it is the second data buffer of dual buffer descriptor, the descriptor is not fetched again.†

Abort During Internal DMA Transfer

If the host issues an SD/SDIO STOP_TRANSMISSION command (CMD12) to the card while data transfer is in progress, the internal DMA controller closes the present descriptor after completing the data transfer until a Data Transfer Over (DTO) interrupt is asserted. Once a STOP_TRANSMISSION command is issued, the DMA controller performs single burst transfers.†

- For a card write operation, the internal DMA controller keeps writing data to the FIFO buffer after fetching it from the system memory until a DTO interrupt is asserted. This is done to keep the card clock running so that the STOP_TRANSMISSION command is reliably sent to the card.†
- For a card read operation, the internal DMA controller keeps reading data from the FIFO buffer and writes to the system memory until a DTO interrupt is generated. This is required because DTO interrupt is not generated until and unless all the FIFO buffer data is emptied.†

Note: For a card write abort, only the current descriptor during which a STOP_TRANSMISSION command is issued is closed by the internal DMA controller. The remaining unread descriptors are not closed by the internal DMA controller.†

Note: For a card read abort, the internal DMA controller reads the data out of the FIFO buffer and writes them to the corresponding descriptor data buffers. The remaining unread descriptors are not closed.†

Fatal Bus Error Scenarios

A fatal bus error occurs due to an error response through the master interface. This error is a system error, so the software driver must not perform any further setup on the controller. The only recovery mechanism from such scenarios is to perform one of the following tasks:†

- Issue a reset to the controller through the reset manager.†
- Issue a program controller reset by writing to the controller reset bit (`controller_reset`) of the `ctrl` register.†

FIFO Buffer Overflow and Underflow

During normal data transfer conditions, FIFO buffer overflow and underflow does not occur. However, if there is a programming error, a FIFO buffer overflow or underflow can result. For example, consider the following scenarios.†

For transmit:†

- PBL=4†
- TX watermark = 1†

For these programming values, if the FIFO buffer has only one location empty, the DMA attempts to read four words from memory even though there is only one word of storage available. This results in a FIFO Buffer Overflow interrupt.†

For receive:†

- PBL=4†
- RX watermark = 1†

For these programming values, if the FIFO buffer has only one location filled, the DMA attempts to write four words, even though only one word is available. This results in a FIFO Buffer Underflow interrupt.†

The driver must ensure that the number of bytes to be transferred, as indicated in the descriptor, is a multiple of four bytes. For example, if the `bytcnt` register = 13, the number of bytes indicated in the descriptor must be rounded up to 16 because the length field must always be a multiple of four bytes.†

PBL and Watermark Levels

This table shows legal PBL and FIFO buffer watermark values for internal DMA controller data transfer operations.†

Table 14-9: PBL and Watermark Levels†

PBL (Number of transfers)	TX/RX FIFO Buffer Watermark Value
1	greater than or equal to 1
4	greater than or equal to 4
8	greater than or equal to 8
16	greater than or equal to 16
32	greater than or equal to 32
64	greater than or equal to 64
128	greater than or equal to 128
256	greater than or equal to 256

CIU

The CIU interfaces with the BIU and SD/MMC cards or devices. The host processor writes command parameters to the SD/MMC controller's BIU control registers and these parameters are then passed to the CIU. Depending on control register values, the CIU generates SD/MMC command and data traffic on the card bus according to the SD/MMC protocol. The control register values also decide whether the command and data traffic is directed to the CE-ATA card, and the SD/MMC controller controls the command and data path accordingly.†

The following list describes the CIU operation restrictions:†

- After a command is issued, the CIU accepts another command only to check read status or to stop the transfer.†
- Only one data transfer command can be issued at a time.†
- During an open-ended card write operation, if the card clock is stopped because the FIFO buffer is empty, the software must first fill the data into the FIFO buffer and start the card clock. It can then issue only an SD/SDIO STOP_TRANSMISSION (CMD12) command to the card.†
- During an SDIO/COMBO card transfer, if the card function is suspended and the software wants to resume the suspended transfer, it must first reset the FIFO buffer and start the resume command as if it were a new data transfer command.†
- When issuing SD/SDIO card reset commands (GO_IDLE_STATE, GO_INACTIVE_STATE or CMD52_reset) while a card data transfer is in progress, the software must set the stop abort command bit (`stop_abort_cmd`) in the `cmd` register to 1 so that the controller can stop the data transfer after issuing the card reset command.†
- If the card clock is stopped because the FIFO buffer is full during a card read, the software must read at least two FIFO buffer locations to start the card clock.†
- If CE-ATA card device interrupts are enabled (the `nIEN` bit is set to 0 in the ATA control register), a new RW_BLK command must not be sent to the same card device if there is a pending RW_BLK command in progress (the RW_BLK command used in this document is the RW_MULTIPLE_BLOCK MMC command defined by the CE-ATA specification). Only the Command Completion Signal Disable (CCSD) command can be sent while waiting for the Command Completion Signal (CCS).†
- For the same card device, a new command is allowed for reading status information, if interrupts are disabled in the CE-ATA card (the `nIEN` bit is set to 1 in the ATA control register).†
- Open-ended transfers are not supported for the CE-ATA card devices.†
- The `send_auto_stop` signal is not supported (software must not set the `send_auto_stop` bit in the `cmd` register) for CE-ATA transfers.†

The CIU consists of the following primary functional blocks:†

- Command path†
- Data path†
- Clock control†

Command Path

The command path performs the following functions:†

- Load card command parameters†
- Send commands to card bus†
- Receive responses from card bus†
- Send responses to BIU†
- Load clock parameters†
- Drives the P-bit on command pin†

A new command is issued to the controller by writing to the BIU registers and setting the `start_cmd` bit in the `cmd` register. The command path loads the new command (command, command argument, timeout) and sends an acknowledgement to the BIU.†

After the new command is loaded, the command path state machine sends a command to the card bus—including the internally generated seven-term CRC (CRC-7)—and receives a response, if any. The state machine then sends the received response and signals to the BIU that the command is done, and then waits for eight clock cycles before loading a new command. In CE-ATA data payload transfer (RW_MULTIPLE_BLOCK) commands, if the card device interrupts are enabled (the nIEN bit is set to 0 in the ATA control register), the state machine performs the following actions after receiving the response:†

- Does not drive the P-bit; it waits for CCS, decodes and goes back to idle state, and then drives the P-bit.†
- If the host wants to send the CCSD command and if eight clock cycles are expired after the response, it sends the CCSD pattern on the command pin.†

Load Command Parameters

Commands or responses are loaded in the command path in the following situations:†

- New command from BIU—When the BIU sends a new command to the CIU, the `start_cmd` bit is set to 1 in the `cmd` register.†
- Internally-generated `send_auto_stop`—When the data path ends, the SD/SDIO STOP command request is loaded.†
- Interrupt request (IRQ) response with relative card address (RCA) 0x000—When the command path is waiting for an IRQ response from the MMC and a “send irq response” request is signaled by the BIU, the send IRQ request bit (`send_irq_response`) is set to 1 in the `ctrl` register.†

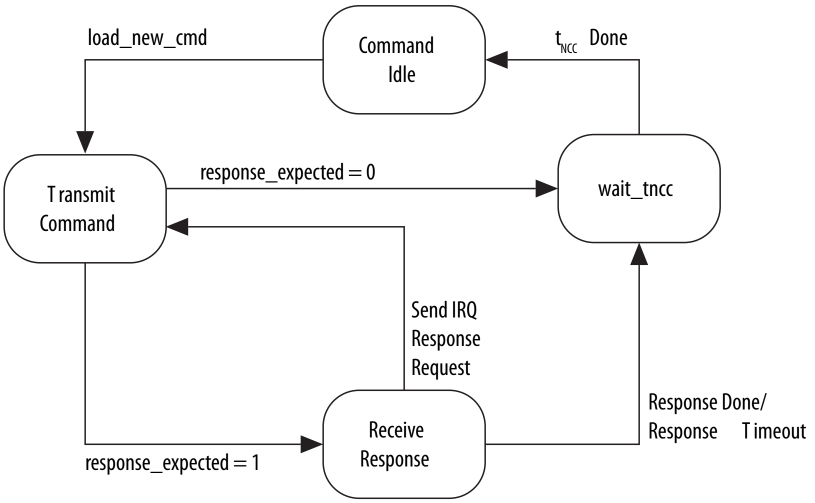
Loading a new command from the BIU in the command path depends on the following `cmd` register bit settings:†

- `update_clock_registers_only`—If this bit is set to 1 in the `cmd` register, the command path updates only the `clkena`, `clkdiv`, and `clksrc` registers. If this bit is set to 0, the command path loads the `cmd`, `cmdarg`, and `tmout` registers. It then processes the new command, which is sent to the card.†
- `wait_prvdata_complete`—If this bit is set to 1, the command path loads the new command under one of the following conditions:†
  - Immediately, if the data path is free (that is, there is no data transfer in progress), or if an open-ended data transfer is in progress (`bytcnt` = 0).†
  - After completion of the current data transfer, if a predefined data transfer is in progress.†

Send Command and Receive Response

After a new command is loaded in the command path (the `update_clock_registers_only` bit in the `cmd` register is set to 0), the command path state machine sends out a command on the card bus.†

Figure 14-7: Command Path State Machine†



The command path state machine performs the following functions, according to cmd register bit values:[†]

1. send_initialization—Initialization sequence of 80 clock cycles is sent before sending the command.[†]
2. response_expected—A response is expected for the command. After the command is sent out, the command path state machine receives a 48-bit or 136-bit response and sends it to the BIU. If the start bit of the card response is not received within the number of clock cycles (as set up in the tmout register), the rto bit and command done (CD) bit are set to 1 in the rintsts register, to signal to the BIU. If the response-expected bit is set to 0, the command path sends out a command and signals a response done to the BIU, which causes the cmd bit to be set to 1 in the rintsts register.[†]
3. response_length—If this bit is set to 1, a 136-bit long response is received; if it is set to 0, a 48-bit short response is received.[†]
4. check_response_crc—If this bit is set to 1, the command path compares CRC-7 received in the response with the internally-generated CRC-7. If the two do not match, the response CRC error is signaled to the BIU, that is, the rcrc bit is set to 1 in the rintsts register.[†]

Send Response to BIU

If the response_expected bit is set to 1 in the cmd register, the received response is sent to the BIU. Response register 0 (resp0) is updated for a short response, and the response register 3 (resp3), response register 2 (resp2), response register 1 (resp1), and resp0 registers are updated on a long response, after which the cmd bit is set to 1 in the rintsts register. If the response is for an AUTO_STOP command sent by the CIU, the response is written to the resp1 register, after which the auto command done bit (acd) is set to 1 in the rintsts register.[†]

The command path verifies the contents of the card response.

Table 14-10: Card Response Fields[†]

Field	Contents
Response transmission bit	0
Command index	Command index of the sent command
End bit	1

The command index is not checked for a 136-bit response or if the check_response_crc bit in the cmd register is set to 0. For a 136-bit response and reserved CRC 48-bit responses, the command index is reserved, that is, 0b111111.[†]

Related Information

www.sdcard.org

For more information about response values, refer to Physical Layer Simplified Specification, Version 3.01 as described on the SD Association website.

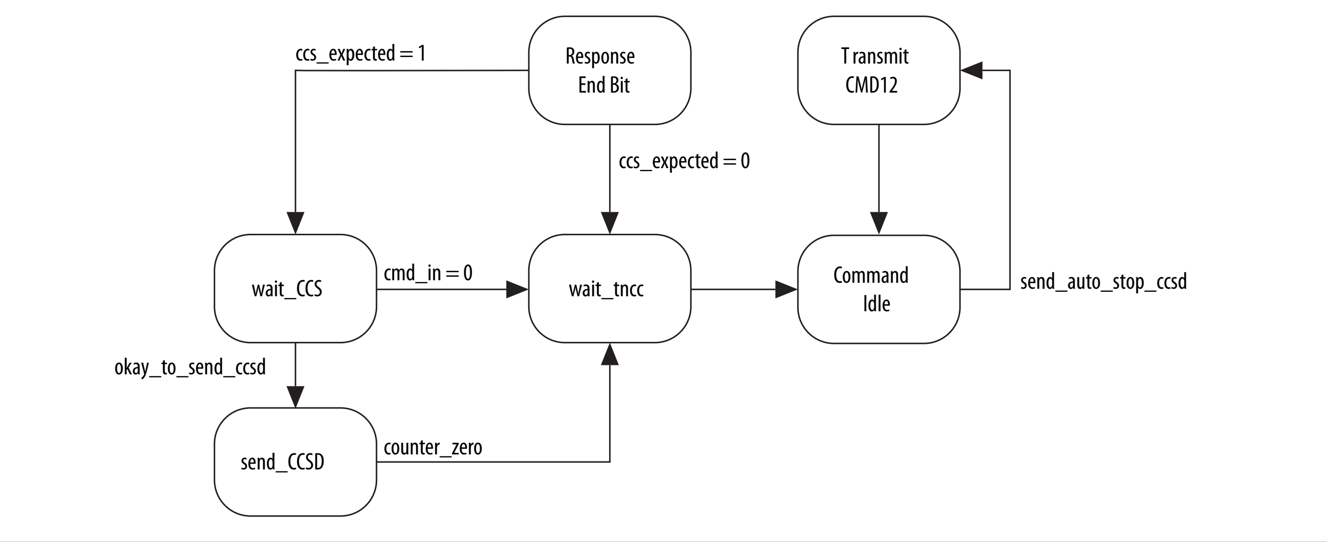
Driving P-bit to the CMD Pin

The command path drives a one-cycle pull-up bit (P-bit) to 1 on the CMD pin between two commands if a response is not expected. If a response is expected, the P-bit is driven after the response is received and before the start of the next command. While accessing a CE-ATA card device, for commands that expect a CCS, the P-bit is driven after the response only if the interrupts are disabled in the CE-ATA card (the nIEN bit is set to 1 in the ATA control register), that is, the CCS expected bit (ccs_expected) in the cmd register is set to 0. If the command expects the CCS, the P-bit is driven only after receiving the CCS.[†]

Polling the CCS

CE-ATA card devices generate the CCS to notify the host controller of the normal ATA command completion or ATA command termination. After receiving the response from the card, the command path state machine performs the functions illustrated in the following figure according to `cmd` register bit values.†

Figure 14-8: CE-ATA Command Path State Machine†



The above figure illustrates:

- Response end bit state—The state machine receives the end bit of the response from the card device. If the `ccs_expected` bit of the `cmd` register is set to 1, the state machine enters the wait CCS state.†
- Wait CCS—The state machine waits for the CCS from the CE-ATA card device. While waiting for the CCS, the following events can happen:†

  1. Software sets the send CCSD bit (`send_ccsd`) in the `ctrl` register, indicating not to wait for CCS and to send the CCSD pattern on the command line.†
  2. Receive the CCS on the CMD line.†
- Send CCSD command—Sends the CCSD pattern (0b00001) on the CMD line.†

CCS Detection and Interrupt to Host Processor

If the `ccs_expected` bit in the `cmd` register is set to 1, the CCS from the CE-ATA card device is indicated by setting the data transfer over bit (`dto`) in the `rintsts` register. The controller generates a DTO interrupt if this interrupt is not masked.†

For the RW_MULTIPLE_BLOCK commands, if the CE-ATA card device interrupts are disabled (the `nIEN` bit is set to 1 in the ATA control register)— that is, the `ccs_expected` bit is set to 0 in the `cmd` register—there are no CCSs from the card. When the data transfer is over—that is, when the requested number of bytes are transferred—the `dto` bit in the `rintsts` register is set to 1.†

CCS Timeout

If the command expects a CCS from the card device (the `ccs_expected` bit is set to 1 in the `cmd` register), the command state machine waits for the CCS and remains in the wait CCS state. If the CE-ATA card fails to send out the CCS, the host software must implement a timeout mechanism to free the command and data path. The controller does not implement a hardware timer; it is the responsibility of the host software to maintain a software timer.[†]

In the event of a CCS timeout, the host must issue a CCSD command by setting the `send_ccsd` bit in the `ctrl` register. The controller command path state machine sends the CCSD command to the CE-ATA card device and exits to an idle state. After sending the CCSD command, the host must also send an SD/SDIO STOP_TRANSMISSION command to the CE-ATA card to abort the outstanding ATA command.[†]

Send CCSD Command

If the `send_ccsd` bit in the `ctrl` register is set to 1, the controller sends a CCSD pattern on the CMD line. The host can send the CCSD command while waiting for the CCS or after a CCS timeout happens.[†]

After sending the CCSD pattern, the controller sets the `cmd` bit in the `rintsts` register and also generates an interrupt to the host if the Command Done interrupt is not masked.[†]

Note: Within the CIU block, if the `send_ccsd` bit in the `ctrl` register is set to 1 on the same clock cycle as CCS is sampled, the CIU block does not send a CCSD pattern on the CMD line. In this case, the `dto` and `cmd` bits in the `rintsts` register are set to 1.[†]

Note: Due to asynchronous boundaries, the CCS might have already happened and the `send_ccsd` bit is set to 1. In this case, the CCSD command does not go to the CE-ATA card device and the `send_ccsd` bit is not set to 0. The host must reset the `send_ccsd` bit to 0 before the next command is issued.[†]

If the send auto stop CCSD (`send_auto_stop_ccsd`) bit in the `ctrl` register is set to 1, the controller sends an internally generated STOP_TRANSMISSION command (CMD12) after sending the CCSD pattern. The controller sets the `acd` bit in the `rintsts` register.[†]

I/O transmission delay (N_{ACIO} Timeout)

The host software maintains the timeout mechanism for handling the I/O transmission delay (N_{ACIO} cycles) time-outs while reading from the CE-ATA card device. The controller neither maintains any timeout mechanism nor indicates that N_{ACIO} cycles are elapsed while waiting for the start bit of a data token. The I/O transmission delay is applicable for read transfers using the RW_REG and RW_BLK commands; the RW_REG and RW_BLK commands used in this document refer to the RW_MULTIPLE_REGISTER and RW_MULTIPLE_BLOCK MMC commands defined by the CE-ATA specification.[†]

Note: After the N_{ACIO} timeout, the application must abort the command by sending the CCSD and STOP commands, or the STOP command. The Data Read Timeout (DRTO) interrupt might be set to 1 while a STOP_TRANSMISSION command is transmitted out of the controller, in which case the data read timeout boot data start bit (`bds`) and the `dto` bit in the `rintsts` register are set to 1.[†]

Data Path

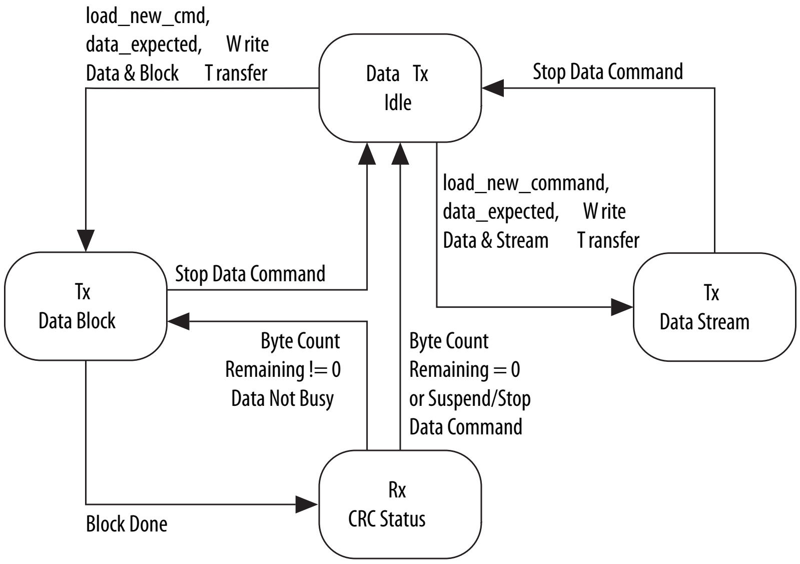
The data path block reads the data FIFO buffer and transmits data on the card bus during a write data transfer, or receives data and writes it to the FIFO buffer during a read data transfer. The data path loads new data parameters—data expected, read/write data transfer, stream/block transfer, block size, byte count, card type, timeout registers—whenever a data transfer command is not in progress. If the data transfer expected bit (`data_expected`) in the `cmd` register is set to 1, the new command is a data transfer command and the data path starts one of the following actions:[†]

- Transmits data if the read/write bit = 1[†]
- Receives data if read/write bit = 0[†]

Data Transmit

The data transmit state machine starts data transmission two clock cycles after a response for the data write command is received. This occurs even if the command path detects a response error or response CRC error. If a response is not received from the card because of a response timeout, data is not transmitted. Depending upon the value of the transfer mode bit (`transfer_mode`) in the `cmd` register, the data transmit state machine puts data on the card data bus in a stream or in blocks.†

Figure 14-9: Data Transmit State Machine†



Stream Data Transmit

If the `transfer_mode` bit in the `cmd` register is set to 1, the transfer is a stream-write data transfer. The data path reads data from the FIFO buffer from the BIU and transmits in a stream to the card data bus. If the FIFO buffer becomes empty, the card clock is stopped and restarted once data is available in the FIFO buffer.†

If the `bytcnt` register is reset to 0, the transfer is an open-ended stream-write data transfer. During this data transfer, the data path continuously transmits data in a stream until the host software issues an SD/SDIO STOP command. A stream data transfer is terminated when the end bit of the STOP command and end bit of the data match over two clock cycles.†

If the `bytcnt` register is written with a nonzero value and the `send_auto_stop` bit in the `cmd` register is set to 1, the STOP command is internally generated and loaded in the command path when the end bit of the STOP command occurs after the last byte of the stream write transfer matches. This data transfer can also terminate if the host issues a STOP command before all the data bytes are transferred to the card bus.†

Single Block Data

If the `transfer_mode` bit in the `cmd` register is set to 0 and the `bytcnt` register value is equal to the value of the `block_size` register, a single-block write-data transfer occurs. The data transmit state machine sends data in a single block, where the number of bytes equals the block size, including the internally-generated 16-term CRC (CRC-16).†

If the `ctype` register is set for a 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC-16 is separately generated and transmitted for 1, 4, or 8 data lines, respectively.†

After a single data block is transmitted, the data transmit state machine receives the CRC status from the card and signals a data transfer to the BIU. This happens when the dto bit in the rintsts register is set to 1.†

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the dcrc bit in the rintsts register.†

Additionally, if the start bit of the CRC status is not received by two clock cycles after the end of the data block, a CRC status start-bit error (SBE) is signaled to the BIU by setting the sbe bit in the rintsts register.†

Multiple Block Data

A multiple-block write-data transfer occurs if the transfer_mode bit in the cmd register is set to 0 and the value in the bytcnt register is not equal to the value of the block_size register. The data transmit state machine sends data in blocks, where the number of bytes in a block equals the block size, including the internally-generated CRC-16 value.†

If the ctype register is set to 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC-16 is separately generated and transmitted on 1, 4, or 8 data lines, respectively.†

After one data block is transmitted, the data transmit state machine receives the CRC status from the card. If the remaining byte count becomes 0, the data path signals to the BIU that the data transfer is done. This happens when the dto bit in the rintsts register is set to 1.†

If the remaining data bytes are greater than zero, the data path state machine starts to transmit another data block.†

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the dcrc bit in the rintsts register, and continues further data transmission until all the bytes are transmitted.†

If the CRC status start bit is not received by two clock cycles after the end of a data block, a CRC status SBE is signaled to the BIU by setting the ebe bit in the rintsts register and further data transfer is terminated.†

If the send_auto_stop bit is set to 1 in the cmd register, the SD/SDIO STOP command is internally generated during the transfer of the last data block, where no extra bytes are transferred to the card. The end bit of the STOP command might not exactly match the end bit of the CRC status in the last data block.†

If the block size is less than 4, 16, or 32 for card data widths of 1 bit, 4 bits, or 8 bits, respectively, the data transmit state machine terminates the data transfer when all the data is transferred, at which time the internally-generated STOP command is loaded in the command path.†

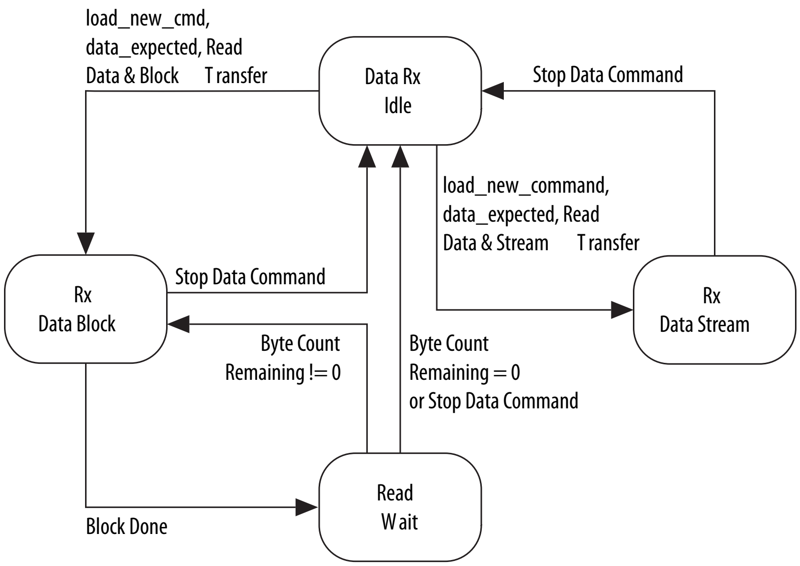
If the bytcnt is zero (the block size must be greater than zero) the transfer is an open-ended block transfer. The data transmit state machine for this type of data transfer continues the block-write data transfer until the host software issues an SD/SDIO STOP or STOP_TRANSMISSION (CMD12) command.†

Data Receive

The data-receive state machine receives data two clock cycles after the end bit of a data read command, even if the command path detects a response error or response CRC error. If a response is not received from the card because a response timeout occurs, the BIU does not receive a signal that the data transfer is complete. This happens if the command sent by the controller is an illegal operation for the card, which keeps the card from starting a read data transfer.†

If data is not received before the data timeout, the data path signals a data timeout to the BIU and an end to the data transfer done. Based on the value of the `transfer_mode` bit in the `cmd` register, the data-receive state machine gets data from the card data bus in a stream or block(s).†

Figure 14-10: Data Receive State Machine†



Stream Data Read

A stream-read data transfer occurs if the `transfer_mode` bit in the `cmd` register is set to 1, at which time the data path receives data from the card and writes it to the FIFO buffer. If the FIFO buffer becomes full, the card clock stops and restarts once the FIFO buffer is no longer full.†

An open-ended stream-read data transfer occurs if the `bytcnt` register is set to 0. During this type of data transfer, the data path continuously receives data in a stream until the host software issues an SD/SDIO STOP command. A stream data transfer terminates two clock cycles after the end bit of the STOP command.†

If the `bytcnt` register contains a nonzero value and the `send_auto_stop` bit in the `cmd` register is set to 1, a STOP command is internally generated and loaded into the command path, where the end bit of the STOP command occurs after the last byte of the stream data transfer is received. This data transfer can terminate if the host issues an SD/SDIO STOP or STOP_TRANSMISSION (CMD12) command before all the data bytes are received from the card.†

Single-block Data Read

If the `ctype` register is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC-16 is separately generated and checked for 1, 4, or 8 data lines, respectively. If there is a CRC-16 mismatch, the data path signals a data CRC error to the BIU. If the received end bit is not 1, the BIU receives an End-bit Error (EBE).†

Multiple-block Data Read

If the `transfer_mode` bit in the `cmd` register is set to 0 and the value of the `bytcnt` register is not equal to the value of the `block_size` register, the transfer is a multiple-block read-data transfer. The data-receive state machine receives data in blocks, where the number of bytes in a block is equal to the block size, including the internally-generated CRC-16.†

If the `ctype` register is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC-16 is separately generated and checked for 1, 4, or 8 data lines, respectively. After a data block is received, if the remaining byte count becomes zero, the data path signals a data transfer to the BIU.†

If the remaining data bytes are greater than zero, the data path state machine causes another data block to be received. If CRC-16 of a received data block does not match the internally-generated CRC-16, a data CRC error to the BIU and data reception continue further data transmission until all bytes are transmitted. Additionally, if the end of a received data block is not 1, data on the data path signals terminate the bit error to the CIU and the data-receive state machine terminates data reception, waits for data timeout, and signals to the BIU that the data transfer is complete.†

If the send_auto_stop bit in the cmd register is set to 1, the SD/SDIO STOP command is internally generated when the last data block is transferred, where no extra bytes are transferred from the card. The end bit of the STOP command might not exactly match the end bit of the last data block.†

If the requested block size for data transfers to cards is less than 4, 16, or 32 bytes for 1-bit, 4-bit, or 8-bit data transfer modes, respectively, the data-transmit state machine terminates the data transfer when all data is transferred, at which point the internally-generated STOP command is loaded in the command path. Data received from the card after that are then ignored by the data path.†

If the bytcnt register is 0 (the block size must be greater than zero), the transfer is an open-ended block transfer. For this type of data transfer, the data-receive state machine continues the block-read data transfer until the host software issues an SD/SDIO STOP or STOP_TRANSMISSION (CMD12) command.†

Auto-Stop

The controller internally generates an SD/SDIO STOP command and is loaded in the command path when the send_auto_stop bit in the cmd register is set to 1. The AUTO_STOP command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards. The software must set the send_auto_stop bit according to the following details: †

The following list describes conditions for the AUTO_STOP command:†

- Stream-read for MMC with byte count greater than zero—The controller generates an internal STOP command and loads it into the command path so that the end bit of the STOP command is sent when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than six (48 bits), a few extra data bytes are received from the card before the end bit of the STOP command is sent.†
- Stream-write for MMC with byte count greater than zero—The controller generates an internal STOP command and loads it into the command path so that the end bit of the STOP command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than six (48 bits), the data path transmits the data last to meet these condition.†
- Multiple-block read memory for SD card with byte count greater than zero—If the block size is less than four (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the AUTO_STOP command is loaded in the command path after all the bytes are read. Otherwise, the STOP command is loaded in the command path so that the end bit of the STOP command is sent after the last data block is received.†
- Multiple-block write memory for SD card with byte count greater than zero—If the block size is less than three (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the AUTO_STOP command is loaded in the command path after all data blocks are transmitted. Otherwise, the STOP command is loaded in the command path so that the end bit of the STOP command is sent after the end bit of the CRC status is received.†
- Precaution for host software during auto-stop—When an AUTO_STOP command is issued, the host software must not issue a new command to the controller until the AUTO_STOP command is sent by the controller and the data transfer is complete. If the host issues a new command during a data transfer with the AUTO_STOP command in progress, an AUTO_STOP command might be sent after the new command is sent and its response is received. This can delay sending the STOP command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the host wants to terminate the data transfer before the data transfer is complete, it can issue an SD/SDIO STOP or STOP_TRANSMISSION (CMD12) command, in which case the controller does not generate an AUTO_STOP command.†

Auto-Stop Generation for MMC Cards

Table 14-11: Auto-Stop Generation for MMC Cards†

Transfer Type	Byte Count	send_auto_stop bit set	Comments
Stream read	0	No	Open-ended stream
Stream read	>0	Yes	Auto-stop after all bytes transfer
Stream write	0	No	Open-ended stream
Stream write	>0	Yes	Auto-stop after all bytes transfer
Single-block read	>0	No	Byte count = 0 is illegal
Single-block write	>0	No	Byte count = 0 is illegal
Multiple-block read	0	No	Open-ended multiple block
Multiple-block read	>0	Yes †	Pre-defined multiple block
Multiple-block write	0	No	Open-ended multiple block
Multiple-block write	>0	Yes †	Pre-defined multiple block

Auto-Stop Generation for SD Cards

Table 14-12: Auto-Stop Generation for SD Cards[†]

Transfer Type	Byte Count	send_auto_stop bit set	Comments
Single-block read	>0	No	Byte count = 0 is illegal
Single-block write	>0	No	Byte count = 0 illegal
Multiple-block read	0	No	Open-ended multiple block
Multiple-block read	>0	Yes	Auto-stop after all bytes transfer
Multiple-block write	0	No	Open-ended multiple block
Multiple-block write	>0	Yes	Auto-stop after all bytes transfer

Auto-Stop Generation for SDIO Cards

Table 14-13: Auto-Stop Generation for SDIO Cards[†]

Transfer Type	Byte Count	send_auto_stop bit set	Comments
Single-block read	>0	No	Byte count = 0 is illegal
Single-block write	>0	No	Byte count = 0 illegal
Multiple-block read	0	No	Open-ended multiple block
Multiple-block read	>0	No	Pre-defined multiple block
Multiple-block write	0	No	Open-ended multiple block
Multiple-block write	>0	No	Pre-defined multiple block

Non-Data Transfer Commands that Use Data Path

Some SD/SDIO non-data transfer commands (commands other than read and write commands) also use the data path.[†]

[37] The condition under which the transfer mode is set to block transfer and byte_count is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If byte_count = n*block_size (n = 2, 3, …), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the host software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue CMD18/CMD25 commands without setting the send_auto_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send_auto_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.[†]

Table 14-14: Non-Data Transfer Commands and Requirements†

	PROGRAM_ CSD (CMD27)	SEND_ WRITE_PROT (CMD30)	LOCK_ UNLOCK (CMD42)	SD_ STATUS (ACMD13)	SEND_NUM_ WR_BLOCKS (ACMD22)	SEND_SCR (ACMD51)
cmd Register Setup						
Cmd_index	0x1B=27	0x1E=30	0x2A=42	0x0D=13	0x16=22	0x33=51
Response_ expect	1	1	1	1	1	1
Response_ length	0	0	0	0	0	0
Check_ response_crc	1	1	1	1	1	1
Data_expected	1	1	1	1	1	1
Read/write	1	0	1	0	0	0
Transfer_ mode	0	0	0	0	0	0
Send_auto_ stop	0	0	0	0	0	0
Wait_ prevdata_ complete	0	0	0	0	0	0
Stop_abort_ cmd	0	0	0	0	0	0
cmdarg Register Setup						
	Stuff bits	32-bit write protect data address	Stuff bits	Stuff bits	Stuff bits	Stuff bits
blksiz Register Setup						
	16	4	Num_ bytes [38] †	64	4	8
bytcnt Register Setup						

[38] Num_bytes = Number of bytes specified as per the lock card data structure. Refer to the SD specification and the MMC specification. †

	PROGRAM_CSD (CMD27)	SEND_WRITE_PROT (CMD30)	LOCK_UNLOCK (CMD42)	SD_STATUS (ACMD13)	SEND_NUM_WR_BLOCKS (ACMD22)	SEND_SCR (ACMD51)
	16	4	Num_bytes [38] †	64	4	8

Clock Control Block

The clock control block provides different clock frequencies required for SD/MMC/CE-ATA cards. The clock control block has one clock divider, which is used to generate different card clock frequencies.†

The clock frequency of a card depends on the following clock `ctrl` register settings:†

- `clkdiv` register—Internal clock dividers are used to generate different clock frequencies required for the cards. The division factor for the clock divider can be set by writing to the `clkdiv` register. The clock divider is an 8-bit value that provides a clock division factor from 1 to 510; a value of 0 represents a clock-divider bypass, a value of 1 represents a divide by 2, a value of 2 represents a divide by 4, and so on.†
- `clksrc` register—Set this register to 0 as clock is divided by clock divider 0.†
- `clkena` register—The `cclk_out` card output clock can be enabled or disabled under the following conditions:†
    - `cclk_out` is enabled when the `cclk_enable` bit in the `clkena` register is set to 1 and disabled when set to 0.†
    - Low-power mode can be enabled by setting the `cclk_low_power` bit of the `clkena` register to 1. If low-power mode is enabled to save card power, the `cclk_out` signal is disabled when the card is idle for at least eight card clock cycles. Low-power mode is enabled when a new command is loaded and the command path goes to a non-idle state.†

Under the following conditions, the card clock is stopped or disabled:†

- Clock can be disabled by writing to the `clkena` register.†
- When low-power mode is selected and the card is idle for at least eight clock cycles.†
- FIFO buffer is full, data path cannot accept more data from the card, and data transfer is incomplete—to avoid FIFO buffer overflow.†
- FIFO buffer is empty, data path cannot transmit more data to the card, and data transfer is incomplete—to avoid FIFO buffer underflow.†

Note: The card clock must be disabled through the `clkena` register before the host software changes the values of the `clkdiv` and `clksrc` registers.†

Error Detection

Errors can occur during card operations within the CIU in the following situations.

Response†

- Response timeout—did not receive the response expected with response start bit within the specified number of clock cycles in the timeout register.†
- Response CRC error—response is expected and check response CRC requested; response CRC-7 does not match with the internally-generated CRC-7.†
- Response error—response transmission bit is not 0, command index does not match with the command index of the send command, or response end bit is not 1.†

Data Transmit†

- No CRC status—during a write data transfer, if the CRC status start bit is not received for two clock cycles after the end bit of the data block is sent out, the data path performs the following actions:†
  - Signals no CRC status error to the BIU†
  - Terminates further data transfer†
  - Signals data transfer done to the BIU†
- Negative CRC—if the CRC status received after the write data block is negative (that is, not 0b010), the data path signals a data CRC error to the BIU and continues with the data transfer.†
- Data starvation due to empty FIFO buffer—if the FIFO buffer becomes empty during a write data transmission, or if the card clock stopped and the FIFO buffer remains empty for a data-timeout number of clock cycles, the data path signals a data-starvation error to the BIU and the data path continues to wait for data in the FIFO buffer.†

Data Receive

- Data timeout—during a read-data transfer, if the data start bit is not received before the number of clock cycles specified in the timeout register, the data path does the following action: †
  - Signals a data-timeout error to the BIU†
  - Terminates further data transfer†
  - Signals data transfer done to BIU†
- Data SBE—during a 4-bit or 8-bit read-data transfer, if the all-bit data line does not have a start bit, the data path signals a data SBE to the BIU and waits for a data timeout, after which it signals that the data transfer is done.†
- Data CRC error—during a read-data-block transfer, if the CRC-16 received does not match with the internally generated CRC-16, the data path signals a data CRC error to the BIU and continues with the data transfer.†
- Data EBE—during a read-data transfer, if the end bit of the received data is not 1, the data path signals an EBE to the BIU, terminates further data transfer, and signals to the BIU that the data transfer is done.†
- Data starvation due to FIFO buffer full—during a read data transmission and when the FIFO buffer becomes full, the card clock stops. If the FIFO buffer remains full for a data-timeout number of clock cycles, the data path signals a data starvation error to the BIU, by setting the data starvation host timeout bit (hto) in rintsts register to 1, and the data path continues to wait for the FIFO buffer to empty.†

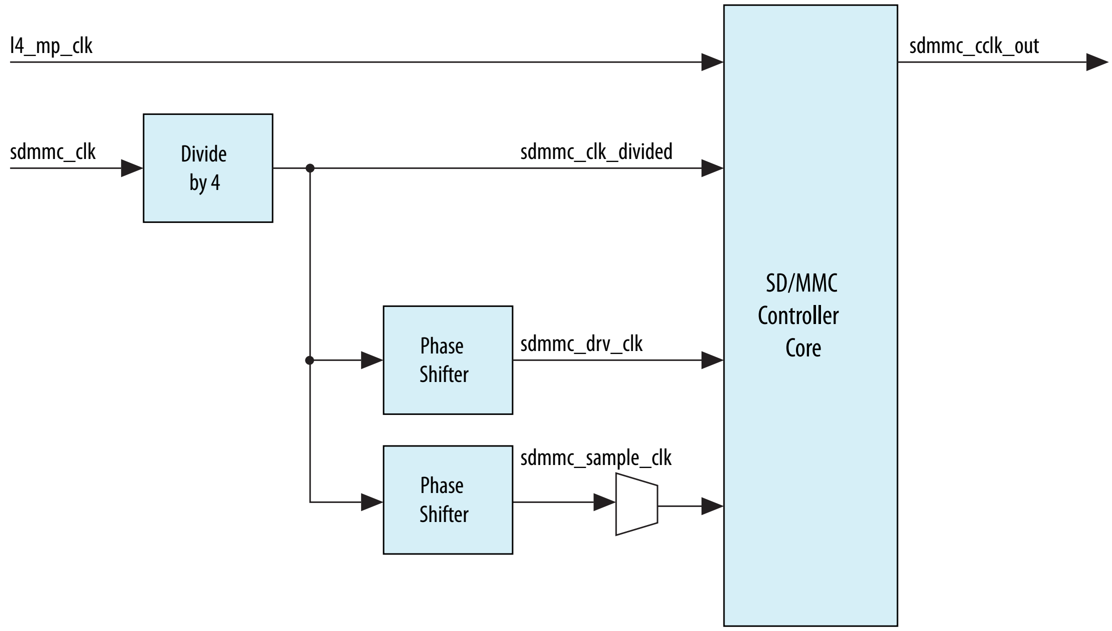
Clocks

Table 14-15: SD/MMC Controller Clocks

Clock Name	Direction	Description
l4_mp_clk	In	Clock for SD/MMC controller BIU
sdmmc_clk	In	Clock for SD/MMC controller
sdmmc_cclk_out	Out	Generated output clock for card
sdmmc_clk_divided	Internal	Divide-by-four clock of sdmmc_clk

Clock Name	Direction	Description
`sdmmc_sample_clk`	Internal	Phase-shifted clock of `sdmmc_clk_divided` used to sample the command and data from the card
`sdmmc_drv_clk`	Internal	Phase-shifted clock of `sdmmc_clk_divided` for controller to drive command and data to the card to meet hold time requirements

Figure 14-11: SD/MMC Controller Clock Connections



The *sdmmc_clk* clock from the clock manager is divided by four and becomes the *sdmmc_clk_divided* clock before passing to the phase shifters and the SD/MMC controller CIU. The phase shifters are used to generate the *sdmmc_drv_clk* and *sdmmc_sample_clk* clocks. These phase shifters provide up to eight phases shift which include 0, 45, 90, 135, 180, 225, 270, and 315 degrees. The *sdmmc_sample_clk* clock can be driven by the output from the phase shifter.

Note: The selections of phase shift degree and *sdmmc_sample_clk* source are done in the system manager. For information about setting the phase shift and selecting the source of the *sdmmc_sample_clk* clock, refer to the *Clock Setup* section within this document.

The controller generates the *sdmmc_cclk_out* clock, which is driven to the card. For more information about the generation of the *sdmmc_cclk_out* clock, refer to the *Clock Control Block* section within this document.

Related Information

- Clock Setup
  Refer to this section for information about setting the phase shift.
- Clock Control Block on page 14-32
  Refer to this section for information about the generation of the `sdmmc_cclk_out`clock.

Resets

The SD/MMC controller has one reset signal. The reset manager drives this signal to the SD/MMC controller on a cold or warm reset.

Related Information

Reset Manager on page 3-1

Interface Signals

Table 14-16: SD/MMC Controller Interface I/O Pins

Signal	Width	Direction	Description
`sdmmc_cclk_out`	1	Out	Clock from controller to the card
`sdmmc_cmd`	1	In/Out	Card command
`sdmmc_pwren`	1	Out	External device power enable
`sdmmc_data`	8	In/Out	Card data

SD/MMC Controller Programming Model

Software and Hardware Restrictions†

Only one data transfer command should be issued at one time. For CE-ATA devices, if CE-ATA device interrupts are enabled (nIEN=0), only one RW_MULTIPLE_BLOCK command (RW_BLK) should be issued; no other commands (including a new RW_BLK) should be issued before the Data Transfer Over status is set for the outstanding RW_BLK.†

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.†

If the card is enumerated in SDR50, SDR104, or DDR50 mode, then the application must program the use_hold_reg bit[29] in the CMD register to 1’b0 (phase shift of cclk_in_drv = 0) or 1'b1 (phase shift of cclk_in_drv > 0). If the card is enumerated in SDR12 or SDR25 mode, the application must program the use_hold_reg bit[29] in the CMD register to 1’b1.†

This programming should be done for all data transfer commands and non-data commands that are sent to the card. When the use_hold_reg bit is programmed to 1’b0, the DWC_mobile_storage bypasses the Hold Registers in the transmit path. The value of this bit should not be changed when a Command or Data Transfer is in progress.†

For more information on using the `use_hold_reg` and the implementation requirements for meeting the card input hold time, refer to the latest version of the Synopsys DesignWare Cores Mobile Storage Host Databook.

Avoiding Glitches in the Card Clock Outputs†

To avoid glitches in the card clock outputs (`cclk_out`), the software should use the following steps when changing the card clock frequency:†

1. Before disabling the clocks, ensure that the card is not busy due to any previous data command. To determine this, check for 0 in bit 9 of the STATUS register.†
2. Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:†

   - `start_cmd` bit†
   - "update clock registers only" bits†
   - "wait_previous data complete"†

   Note: Wait for the CIU to take the command by polling for 0 on the `start_cmd` bit.†
3. Set the `start_cmd` bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.†
4. Set `start_cmd` to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.†

Reading from a Card in Non-DMA Mode†

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. While using the external DMA interface for reading from a card, the DTO interrupt occurs only after all the data is flushed to memory by the DMA Interface unit.†

Writing to a Card in External DMA Mode†

While writing to a card in external DMA mode, if an undefined-length transfer is selected by setting the Byte Count register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.†

Software Issues a `Controller_Reset` Command†

If the software issues a `controller_reset` command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the host. In addition to a card-reset, if a FIFO reset is also issued, then:†

- Any pending DMA transfer on the bus completes correctly†
- DMA data read is ignored†
- Write data is unknown (x)†

Additionally, if dma_reset is also issued, any pending DMA transfer is abruptly terminated. When the DW-DMA/Non-DW-DMA is used, the DMA controller channel should also be reset and reprogrammed.†

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.†

Data-Transfer Requirement Between the FIFO and Host†

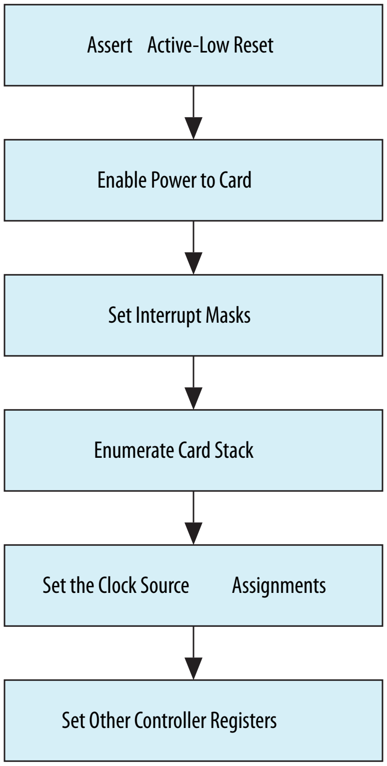
One data-transfer requirement between the FIFO and host is that the number of transfers should be a multiple of the FIFO data width (F_DATA_WIDTH). For example, if F_DATA_WIDTH = 32 and you want to write only 15 bytes to an SD_MMC_CEATA card (BYTCNT), the host should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers, if external DMA mode is enabled. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the host should still read all 16 bytes from the FIFO.†

It is recommended that you not change the FIFO threshold register in the middle of data transfers when DW-DMA/Non-DW-DMA mode is chosen.†

Initialization†

After the power and clock to the controller are stable, the controller active-low reset is asserted. The reset sequence initializes the registers, FIFO buffer pointers, DMA interface controls, and state machines in the controller.†

Figure 14-12: SD/MMC Controller Initialization Sequence†



Power-On Reset Sequence

Software must perform the following steps after the power-on-reset:

1. Before enabling power to the card, confirm that the voltage setting to the voltage regulator is correct. †
2. Enable power to the card by setting the power enable bit (power_enable) in the power enable register (pwren) to 1. Wait for the power ramp-up time before proceeding to the next step.†
3. Set the interrupt masks by resetting the appropriate bits to 0 in the intmask register.†
4. Set the int_enable bit of the ctrl register to 1.†

Note: Altera recommends that you write 0xFFFFFFFF to the `rintsts` register to clear any pending interrupts before setting the `int_enable` bit to 1.†

5. Discover the card stack according to the card type. For discovery, you must restrict the clock frequency to 400 kHz in accordance with SD/MMC/CE-ATA standards. For more information, refer to *Enumerated Card Stack*.†
6. Set the clock source assignments. Set the card frequency using the `clkdiv` and `clksrc` registers of the controller. For more information, refer to *Clock Setup*.†
7. The following common registers and fields can be set during initialization process:†
    - The response timeout field (`response_timeout`) of the `tmout` register. A typical value is 0x64.†
    - The data timeout field (`data_timeout`) of the `tmout` register, highest of the following:†
        - 10 * N_{AC} †

          N_{AC} = card device total access time†

          = 10 * ((TAAC * F_{OP}) + (100 * NSAC))†

          where:†

          TAAC = Time-dependent factor of the data access time†

          F_{OP} = The card clock frequency used for the card operation†

          NSAC = Worst-case clock rate-dependent factor of the data access time†
        - Host FIFO buffer latency†

          On read: Time elapsed before host starts reading from a full FIFO buffer†

          On write: Time elapsed before host starts writing to an empty FIFO buffer†
    - Debounce counter register (`debnce`). A typical debounce value is 25 ms.†
    - TX watermark field (`tx_wmark`) of the FIFO threshold watermark register (`fifoth`). Typically, the threshold value is set to 512, which is half the FIFO buffer depth.†
    - RX watermark field (`rx_wmark`) of the `fifoth` register. Typically, the threshold value is set to 511.†

These registers do not need to be changed with every SD/MMC/CE-ATA command. Set them to a typical value according to the SD/MMC/CE-ATA specifications.

Related Information

- **Clock Setup** on page 14-41
  Refer to this section for information on setting the clock source assignments.
- **Enumerated Card Stack** on page 14-38
  Refer to this section for information on discovering the card stack according to the card type.

Enumerated Card Stack

The card stack performs the following tasks:

- Discovers the connected card †
- Sets the relative Card Address Register (RCA) in the connected card†
- Reads the card specific information†
- Stores the card specific information locally†

The card connected to the controller can be an MMC, CE-ATA, SD or SDIO (including IO ONLY, MEM ONLY and COMBO) card.

Identifying the Connected Card Type

To identify the connected card type, the following discovery sequence is needed:

1. Reset the card width 1 or 4 bit (`card_width2`) and card width 8 bit (`card_width1`) fields in the `ctype` register to 0.
2. Identify the card type as SD, MMC, SDIO or SDIO-COMBO:
    a. Send an SD/SDIO IO_SEND_OP_COND (CMD5) command with argument 0 to the card.
    b. Read `resp0` on the controller. The response to the IO_SEND_OP_COND command gives the voltage that the card supports.
    c. Send the IO_SEND_OP_COND command, with the desired voltage window in the arguments. This command sets the voltage window and makes the card exit the initialization state.
    d. Check bit 27 in `resp0`:
        - If bit 27 is 0, the SDIO card is IO ONLY. In this case, proceed to **step 5**.
        - If bit 27 is 1, the card type is SDIO COMBO. Continue with the following steps.
3. Only continue with this step if the SDIO card type is COMBO or there is no response received from the previous IO_SEND_OP_COND command. Otherwise, skip to **step 5**.
    a. Send the SD/SDIO SEND_IF_COND (CMD8) command with the following arguments:
        - Bit[31:12] = 0x0 (reserved bits)†
        - Bit[11:8] = 0x1 (supply voltage value)†
        - Bit[7:0] = 0xAA (preferred check pattern by SD memory cards compliant with SDIO Simplified Specification Version 2.00 and later.)†

       Refer to *SDIO Simplified Specification Version 2.00* as described on the SD Association website.
    b. If a response is received to the previous SEND_IF_COND command, the card supports SD High-Capacity, compliant with SD Specifications, Part 1, Physical Layer Simplified Specification Version 2.00.

       If no response is received, proceed to **step e**.
    c. Send the SD_SEND_OP_COND (ACMD41) command with the following arguments:
        - Bit[31] = 0x0 (reserved bits)†
        - Bit[30] = 0x1 (high capacity status)†
        - Bit[29:25] = 0x0 (reserved bits)†
        - Bit[24] = 0x1 (S18R --supports voltage switching for 1.8V)†
        - Bit[23:0] = supported voltage range†
    d. If a response is received to the previous SD_SEND_OP_COND command, the card type is SDHC. Otherwise, the card is MMC or CE-ATA. In either case, skip the following steps and proceed to **step 5**.
    e. If a response is not received to the initial SEND_IF_COND command, the card does not support High Capacity SD2.0. Next, issue the GO_IDLE_STATE command followed by the SD_SEND_OP_COND command with the following arguments:
        - Bit[31] = 0x0 (reserved bits)†
        - Bit[30] = 0x0 (high capacity status)†
        - Bit[29:24] = 0x0 (reserved bits)†
        - Bit[23:0] = supported voltage range†
    f. If a response is received to the previous SD_SEND_OP_COND command, the card is SD type. Otherwise, the card is MMC or CE-ATA.

       Note: You must issue the SEND_IF_COND command prior to the first SD_SEND_OP_COND command, to initialize the High Capacity SD memory card. The card returns busy as a response to the SD_SEND_OP_COND command when any of the following conditions are true:

- The card executes its internal initialization process.
- A SEND_IF_COND command is not issued before the SD_SEND_OP_COND command.
- The ACMD41 command is issued. In the command argument, the Host Capacity Support (HCS) bit is set to 0, for a high capacity SD card.

4. Use the following sequence to determine whether the card is a CE-ATA 1.1, CE-ATA 1.0, or MMC device:

   a. Determine whether the card is a CE-ATA v1.1 card device by attempting to select ATA mode. Send the SD/SDIO SEND_IF_COND command, querying byte 504 (S_CMD_SET) of the EXT_CSD register block in the external card.

   - If bit 4 is set to 1, the card device supports ATA mode. Send the SWITCH_FUNC (CMD6) command, setting the ATA bit (bit 4) of the EXT_CSD register slice 191 (CMD_SET) to 1. This command selects ATA mode and activates the ATA command set.

     You can verify the currently selected mode by reading it back from byte 191 of the EXT_CSD register.

     Skip to **step 5**.
   - If the card device does not support ATA mode, it might be an MMC card or a CE-ATA v1.0 card. Continue to **Step b**.

   b. Determine whether the card is a CE-ATA 1.0 card device or an MMC card device by sending the RW_REG command. If a response is received and the response data contains the CE-ATA signature, the card is a CE-ATA 1.0 card device. Otherwise, the card is an MMC card device.
5. At this point, the software has determined the card type as SD/SDHC, SDIO or SDIO-COMBO. Now it must enumerate the card stack according to the type that has been discovered.
6. Set the card clock source frequency to the frequency of identification clock rate, 400 KHz. Use one of the following discovery command sequences:

   - For an SD card or an SDIO memory section, send the following SD/SDIO command sequence:

     - GO_IDLE_STATE
     - SEND_IF_COND
     - SD_SEND_OP_COND (ACMD41)
     - ALL_SEND_CID (CMD2)
     - SEND_RELATIVE_ADDR (CMD3)
   - For an SDIO card, send the following command sequence:

     - IO_SEND_OP_COND.
     - If the function count is valid, send the SEND_RELATIVE_ADDR command.
   - For an MMC, send the following command sequence:

     - GO_IDLE_STATE
     - SEND_OP_COND (CMD1)
     - ALL_SEND_CID
     - SEND_RELATIVE_ADDR
7. You can change the card clock frequency after discovery by writing a value to the `clkdiv` register that divides down the `sdmmc_clk` clock.

The following list shows typical clock frequencies for various types of cards:

- SD memory card, 25 MHz†
- MMC card device, 12.5 MHz†
- Full speed SDIO, 25 MHz†
- Low speed SDIO, 400 kHz†

Related Information

www.sdcard.org
To learn more about how SD technology works, visit the SD Association website.

Clock Setup

The following registers of the SD/MMC controller allow software to select the desired clock frequency for the card:

- `clksrc`
- `clkdiv`
- `clkena`

The controller loads these registers when it receives an update clocks command.

Changing the Card Clock Frequency

To change the card clock frequency, perform the following steps:

1. Before disabling the clocks, ensure that the card is not busy with any previous data command. To do so, verify that the `data_busy` bit of the status register (`status`) is 0.
2. Reset the `cclk_enable` bit of the `clkena` register to 0, to disable the card clock generation.
3. Reset the `clksrc` register to 0.
4. Set the following bits in the `cmd` register to 1:
   - `update_clk_regs_only`—Specifies the update clocks command†
   - `wait_prvdata_complete`—Ensures that clock parameters do not change until any ongoing data transfer is complete†
   - `start_cmd`—Initiates the command†
5. Wait until the `start_cmd` and `update_clk_regs_only` bits change to 0. There is no interrupt when the clock modification completes. The controller does not set the `command_done` bit in the `rintsts` register upon command completion. The controller might signal a hardware lock error if it already has another command in the queue. In this case, return to **Step 4**.

   For information about hardware lock errors, refer to *Interrupt and Error Handling*.
6. Reset the `sdmmc_clk_enable` bit to 0 in the `enable` register of the clock manager peripheral PLL group (`perpllgrp`).
7. In the control register (`ctrl`) of the SDMMC controller group (`sdmmcgrp`) in the system manager, set the drive clock phase shift select (`drvsel`) and sample clock phase shift select (`smplsel`) bits to specify the required phase shift value.
8. Set the `sdmmc_clk_enable` bit in the `Enable` register of the clock manager `perpllgrp` group to 1.
9. Set the `clkdiv` register of the controller to the correct divider value for the required clock frequency.
10. Set the `cclk_enable` bit of the `clkena` register to 1, to enable the card clock generation.

You can also use the clkena register to enable low-power mode, which automatically stops the sdmmc_cclk_out clock when the card is idle for more than eight clock cycles.

Related Information

Interrupt and Error Handling on page 14-68
Refer to this section for information about hardware lock errors.

Controller/DMA/FIFO Buffer Reset Usage

The following list shows the effect of reset on various parts in the SD/MMC controller:†

- Controller reset—resets the controller by setting the controller_reset bit in the ctrl register to 1. Controller reset resets the CIU and state machines, and also resets the BIU-to-CIU interface. Because this reset bit is self-clearing, after issuing the reset, wait until this bit changes to 0.†
- FIFO buffer reset—resets the FIFO buffer by setting the FIFO reset bit (fifo_reset) in the ctrl register to 1. FIFO buffer reset resets the FIFO buffer pointers and counters in the FIFO buffer. Because this reset bit is self-clearing, after issuing the reset, wait until this bit changes to 0.†
- DMA reset—resets the internal DMA controller logic by setting the DMA reset bit (dma_reset) in the ctrl register to 1, which immediately terminates any DMA transfer in progress. Because this reset bit is self-clearing, after issuing the reset, wait until this bit changes to 0.†

Note: Ensure that the DMA is idle before performing a DMA reset. Otherwise, the L3 interconnect might be left in an indeterminate state.†

Altera recommends setting the controller_reset, fifo_reset, and dma_reset bits in the ctrl register to 1 first, and then resetting the rintsts register to 0 using another write, to clear any resultant interrupt.

Enabling FIFO Buffer ECC

To protect the FIFO buffer data with ECC, you must enable the ECC feature before performing any operations with the SD/MMC controller. Perform the following steps to enable the FIFO buffer ECC feature:

1. Verify there are no commands committed to the controller.
2. Ensure that the FIFO buffer is initialized. Initialize the FIFO buffer by writing 0 to all 1024 FIFO buffer locations. A FIFO buffer write to any address from 0x200 to the maximum FIFO buffer size is valid.
3. Set the SDMMC RAM ECC single and double, correctable error interrupt status bits (serrporta, derrporta, serrportb, and derrportb) to 1 in the sdmmc register in the eccgrp group of the system manager, to clear any previously-detected ECC errors.
4. Reset the FIFO buffer by setting the fifo_reset bit to 1 in the ctrl register. This action resets pointers and counters in the FIFO buffer. This reset bit is self-clearing, so after issuing the reset, wait until the bit changes to 0.
5. Set the en bit in sdmmc register in eccgrp group of the system manager to 1, to enable ECC for the FIFO buffer in SD/MMC controller.

Non-Data Transfer Commands

To send any non-data transfer command, the software needs to write the cmd register and the cmdarg register with appropriate parameters. Using these two registers, the controller forms the command and sends it to the CMD pin. The controller reports errors in the command response through the error bits of the rintsts register.†

When a response is received—either erroneous or valid—the controller sets the command_done bit in the rintsts register to 1. A short response is copied to resp0, while a long response is copied to all four response registers (resp0, resp1, resp2, and resp3).† For long responses, bit 31 of resp3 represents the MSB and bit 0 of resp0 represents the LSB.†

For basic and non-data transfer commands, perform the following steps:

1. Write the cmdarg register with the appropriate command argument parameter.†
2. Write the cmd register with the settings in *Register Settings for Non-Data Transfer Command*.†
3. Wait for the controller to accept the command. The start_cmd bit changes to 0 when the command is accepted.†

   The following actions occur when the command is loaded into the controller:†

   - If no previous command is being processed, the controller accepts the command for execution and resets the start_cmd bit in the cmd register to 0. If a previous command is being processed, the controller loads the new command in the command buffer.†
   - If the controller is unable to load the new command—that is, a command is already in progress, a second command is in the buffer, and a third command is attempted—the controller generates a hardware lock error.†
4. Check if there is a hardware lock error.†
5. Wait for command execution to complete. After receiving either a response from a card or response timeout, the controller sets the command_done bit in the rintsts register to 1. Software can either poll for this bit or respond to a generated interrupt (if enabled).†
6. Check if the response timeout boot acknowledge received (bar), rcrc, or re bit is set to 1. Software can either respond to an interrupt raised by these errors or poll the re, rcrc, and bar bits of the rintsts register. If no response error is received, the response is valid. If required, software can copy the response from the response registers.†

Note: Software cannot modify clock parameters while a command is being executed.†

Related Information

cmd Register Settings for Non-Data Transfer Command† on page 14-43
Refer to this table for information about Non-Data Transfer commands.

cmd Register Settings for Non-Data Transfer Command†

Table 14-17: Default

Parameter	Value	Comment
start_cmd	1	This bit resets itself to 0 after the command is committed.
use_hold_reg	1 or 0	Choose the value based on the speed mode used.
update_clk_regs_only	0	Indicates that the command is not a clock update command
data_expected	0	Indicates that the command is not a data command
card_number	1	For one card

Parameter	Value	Comment
`cmd_index`	Command Index	Set this parameter to the command number. For example, set to 8 for the SD/SDIO SEND_IF_COND (CMD8) command.
`send_initialization`	0 or 1	1 for card reset commands such as the SD/SDIO GO_IDLE_STATE command 0 otherwise
`stop_abort_cmd`	0 or 1	1 for a command to stop data transfer, such as the SD/SDIO STOP_TRANSMISSION command 0 otherwise
`response_length`	0 or 1	1 for R2 (long) response 0 for short response
`response_expect`	0 or 1	0 for commands with no response, such as SD/SDIO GO_IDLE_STATE, SET_DSR (CMD4), or GO_INACTIVE_STATE (CMD15). 1 otherwise

Table 14-18: User Selectable

Parameter	Value	Comment
`wait_prvdata_complete`	1	Before sending a command on the command line, the host must wait for completion of any data command already in process. Altera recommends that you set this bit to 1, unless the current command is to query status or stop data transfer when transfer is in progress.
`check_response_crc`	1 or 0	1 if the response includes a valid CRC, and the software is required to crosscheck the response CRC bits. 0 otherwise

Data Transfer Commands

Data transfer commands transfer data between the memory card and the controller. To issue a data command, the controller requires a command argument, total data size, and block size. Data transferred to or from the memory card is buffered by the controller FIFO buffer.†

Confirming Transfer State

Before issuing a data transfer command, software must confirm that the card is not busy and is in a transfer state, by performing the following steps:†

1. Issue an SD/SDIO SEND_STATUS (CMD13) command. The controller sends the status of the card as the response to the command.†
2. Check the card's busy status.†
3. Wait until the card is not busy.†
4. Check the card's transfer status. If the card is in the stand-by state, issue an SD/SDIO SELECT/DESELECT_CARD (CMD7) command to place it in the transfer state.†

Busy Signal After CE-ATA RW_BLK Write Transfer

During CE-ATA RW_BLK write transfers, the MMC busy signal might be asserted after the last block. If the CE-ATA card device interrupt is disabled (the nIEN bit in the card device's ATA control register is set to 1), the `dto` bit in the `rintsts` register is set to 1 even though the card sends MMC BUSY. The host cannot issue the CMD60 command to check the ATA busy status after a CMD61 command. Instead, the host must perform one of the following actions:†

- Issue the SEND_STATUS command and check the MMC busy status before issuing a new CMD60 command†
- Issue the CMD39 command and check the ATA busy status before issuing a new CMD60 command†

For the data transfer commands, software must set the `ctype` register to the bus width that is programmed in the card.†

Data Transfer Interrupts

The controller generates an interrupt for different conditions during data transfer, which are reflected in the following `rintsts` register bits:†

1. `dto`—Data transfer is over or terminated. If there is a response timeout error, the controller does not attempt any data transfer and the Data Transfer Over bit is never set.†
2. Transmit FIFO data request bit (`txdr`)—The FIFO buffer threshold for transmitting data is reached; software is expected to write data, if available, into the FIFO buffer.†
3. Receive FIFO data request bit (`rxdr`)—The FIFO buffer threshold for receiving data is reached; software is expected to read data from the FIFO buffer.†
4. `hto`—The FIFO buffer is empty during transmission or is full during reception. Unless software corrects this condition by writing data for empty condition, or reading data for full condition, the controller cannot continue with data transfer. The clock to the card is stopped.†
5. `bds`—The card has not sent data within the timeout period.†
6. `dcrc`—A CRC error occurred during data reception.†
7. `sbe`—The start bit is not received during data reception.†
8. `ebe`—The end bit is not received during data reception, or for a write operation. A CRC error is indicated by the card.†

`dcrc`, `sbe`, and `ebe` indicate that the received data might have errors. If there is a response timeout, no data transfer occurs.†

Single-Block or Multiple-Block Read

To implement a single-block or multiple-block read, the software performs the following steps:†

1. Write the data size in bytes to the `bytcnt` register. For a multi-block read, `bytcnt` must be a multiple of the block size.†
2. Write the block size in bytes to the `blksiz` register. The controller expects data to return from the card in blocks of size `blksiz`.†
3. If the read round trip delay, including the card delay, is greater than half of `sdmmc_clk_divided`, write to the card threshold control register (`cardthrctl`) to ensure that the card clock does not stop in the

middle of a block of data being transferred from the card to the host. For more information, refer to *Card Read Threshold*.†

Note: If the card read threshold enable bit (cardrdthren) is 0, the host system must ensure that the RX FIFO buffer does not become full during a read data transfer by ensuring that the RX FIFO buffer is read at a rate faster than that at which data is written into the FIFO buffer. Otherwise, an overflow might occur.†

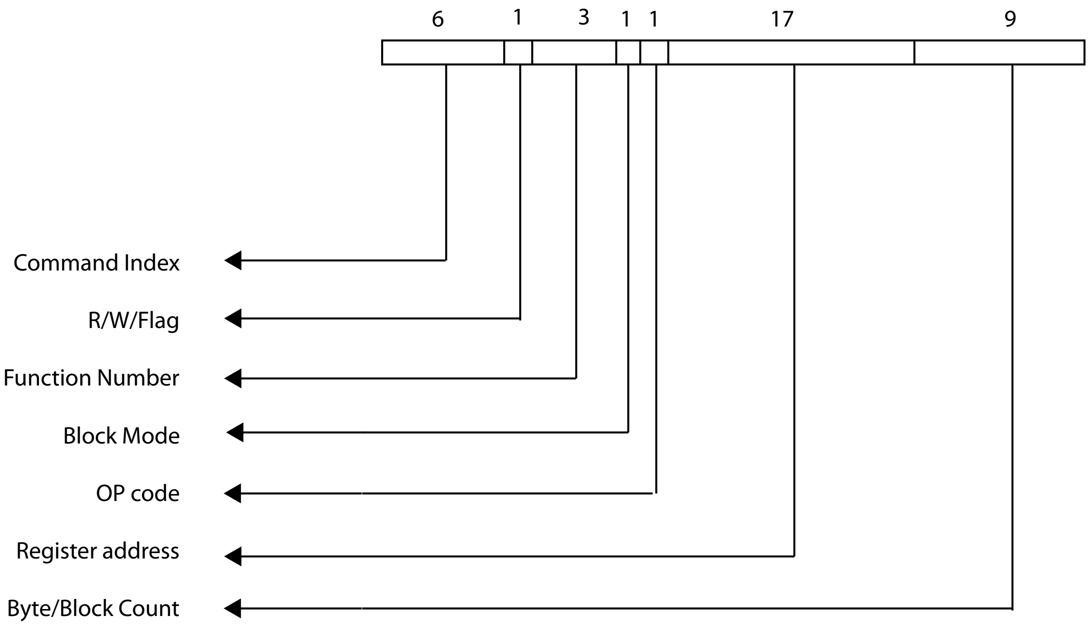
4. Write the cmdarg register with the beginning data address for the data read.†
5. Write the cmd register with the parameters listed in *cmd Register Settings for Single-Block and Multiple-Block Reads*. For SD and MMC cards, use the SD/SDIO READ_SINGLE_BLOCK (CMD17) command for a single-block read and the READ_MULTIPLE_BLOCK (CMD18) command for a multiple-block read. For SDIO cards, use the IO_RW_EXTENDED (CMD53) command for both single-block and multiple-block transfers. The command argument for (CMD53) is shown in the figure, below. After writing to the cmd register, the controller starts executing the command. When the command is sent to the bus, the Command Done interrupt is generated.†
6. Software must check for data error interrupts, reported in the dcrc, bds, sbe, and ebe bits of the rintsts register. If required, software can terminate the data transfer by sending an SD/SDIO STOP command.†
7. Software must check for host timeout conditions in the rintsts register:†

    - Receive FIFO buffer data request†
    - Data starvation from host—the host is not reading from the FIFO buffer fast enough to keep up with data from the card. To correct this condition, software must perform the following steps:†

        - Read the fifo_count field of the status register†
        - Read the corresponding amount of data out of the FIFO buffer†

    In both cases, the software must read data from the FIFO buffer and make space in the FIFO buffer for receiving more data.†
8. When a DTO interrupt is received, the software must read the remaining data from the FIFO buffer.†

Figure 14-13: Command Argument for IO_RW_EXTENDED (CMD53)[†]



Related Information

- Card Read Threshold on page 14-65
  Refer to this section for information about the thresholds for a card read.
- cmd Register Settings for Single-Block and Multiple-Block Reads[†] on page 14-47
  Refer to this table for information about the settings for Single-Block and Multiple-Block Reads.

cmd Register Settings for Single-Block and Multiple-Block Reads[†]

Table 14-19: cmd Register Settings for Single-Block and Multiple-Block Reads (Default)

Parameter	Value	Comment
`start_cmd`	1	This bit resets itself to 0 after the command is committed.
`use_hold_reg`	1 or 0	Choose the value based on speed mode used.
`update_clk_regs_only`	0	Does not need to update clock parameters
`data_expected`	1	Data command
`card_number`	1	For one card
`transfer_mode`	0	Block transfer

Parameter	Value	Comment
`send_initialization`	0	1 for a card reset command such as the SD/SDIO GO_IDLE_STATE command 0 otherwise
`stop_abort_cmd`	0	1 for a command to stop data transfer such as the SD/SDIO STOP_TRANSMISSION command 0 otherwise
`send_auto_stop`	0 or 1	Refer to *Auto Stop* for information about how to set this parameter.
`read_write`	0	Read from card
`response_length`	0	1 for R2 (long) response 0 for short response
`response_expect`	1 or 0	0 for commands with no response, such as SD/SDIO GO_IDLE_STATE, SET_DSR, and GO_INACTIVE_STATE. 1 otherwise

Table 14-20: cmd Register Settings for Single-Block and Multiple-Block Reads (User Selectable)

Parameter	Value	Comment
`wait_prvdata_complete`	1 or 0	0 - sends command to CIU immediately 1 - sends command after previous data transfer ends
`check_response_crc`	1 or 0	0 - Controller must not check response CRC 1 - Controller must check responce CRC
`cmd_index`	Command Index	Set this parameter to the command number. For example, set to 17 or 18 for SD/SDIO READ_SINGLE_BLOCK (CMS17) or READ_MULTIPLE_BLOCK (CMD18)

Related Information

Auto-Stop on page 14-28
Refer to this table for information about setting the `send_auto_stop` parameter.

Single-Block or Multiple-Block Write

The following steps comprise a single-block or multiple-block write:

1. Write the data size in bytes to the bytcnt register. For a multi-block write, bytcnt must be a multiple of the block size.†
2. Write the block size in bytes to the blksiz register. The controller sends data in blocks of size blksiz each.†
3. Write the cmdarg register with the data address to which data must be written.†
4. Write data into the FIFO buffer. For best performance, the host software should write data continuously until the FIFO buffer is full.†
5. Write the cmd register with the parameters listed in *cmd Register Settings for Single-Block and Multiple-Block Write*. For SD and MMC cards, use the SD/SDIO WRITE_BLOCK (CMD24) command for a single-block write and the WRITE_MULTIPLE_BLOCK (CMD25) command for a multiple-block writes. For SDIO cards, use the IO_RW_EXTENDED command for both single-block and multiple-block transfers.†

   After writing to the cmd register, the controller starts executing a command if there is no other command already being processed. When the command is sent to the bus, a Command Done interrupt is generated.†
6. Software must check for data error interrupts; that is, for dcrc, bds, and ebe bits of the rintsts register. If required, software can terminate the data transfer early by sending the SD/SDIO STOP command.†
7. Software must check for host timeout conditions in the rintsts register: †

   - Transmit FIFO buffer data request.†
   - Data starvation by the host—the controller wrote data to the card faster than the host could supply the data.†

   In both cases, the software must write data into the FIFO buffer.†

   There are two types of transfers: open-ended and fixed length.†

   - Open-ended transfers—For an open-ended block transfer, the byte count is 0. At the end of the data transfer, software must send the STOP_TRANSMISSION command (CMD12).†
   - Fixed-length transfers—The byte count is nonzero. You must already have written the number of bytes to the bytcnt register. The controller issues the STOP command for you if you set the send_auto_stop bit of the cmd register to 1. After completion of a transfer of a given number of bytes, the controller sends the STOP command. Completion of the AUTO_STOP command is reflected by the Auto Command Done interrupt. A response to the AUTO_STOP command is written to the resp1 register. If software does not set the send_auto_stop bit in the cmd register to 1, software must issue the STOP command just like in the open-ended case.†

When the dto bit of the rintsts register is set, the data command is complete.†

cmd Register Settings for Single-Block and Multiple-Block Write

Table 14-21: cmd Register Settings for Single-Block and Multiple-Block Write (Default)†

Parameter	Value	Comment
start_cmd	1	This bit resets itself to 0 after the command is committed (accepted by the BIU).
use_hold_reg	1 or 0	Choose the value based on speed mode used.
update_clk_regs_only	0	Does not need to update clock parameters

Parameter	Value	Comment
`data_expected`	1	Data command
`card_number`	1	For one card
`transfer_mode`	0	Block transfer
`send_initialization`	0	Can be 1, but only for card reset commands such as SD/SDIO GO_IDLE_STATE
`stop_abort_cmd`	0	Can be 1 for commands to stop data transfer such as SD/SDIO STOP_TRANSMISSION
`send_auto_stop`	0 or 1	Refer to *Auto Stop* for information about how to set this parameter.
`read_write`	1	Write to card
`response_length`	0	Can be 1 for R2 (long) responses
`response_expect`	1	Can be 0 for commands with no response. For example, SD/SDIO GO_IDLE_STATE, SET_DSR, GO_INACTIVE_STATE etc.

Table 14-22: cmd Register Settings for Single-Block and Multiple-Block Write (User Selectable)†

Parameter	Value	Comment
`wait_prvdata_complete`	1	0—Sends command to the CIU immediately 1—Sends command after previous data transfer ends
`check_response_crc`	1	0—Controller must not check response CRC 1—Controller must check response CRC
`cmd_index`	Command Index	Set this parameter to the command number. For example, set to 24 for SD/SDIO WRITE_BLOCK (CMD24) or 25 for WRITE_MULTIPLE_BLOCK (CMD25).

Related Information

Auto-Stop on page 14-28
Refer to this table for information about setting the `send_auto_stop` parameter.

Stream Read and Write

In a stream transfer, if the byte count is equal to 0, the software must also send the SD/SDIO STOP command. If the byte count is not 0, when a given number of bytes completes a transfer, the controller sends the STOP command automatically. Completion of this AUTO_STOP command is reflected by the

Auto_command_done interrupt. A response to an AUTO_STOP command is written to the `resp1` register. A stream transfer is allowed only for card interfaces with a 1-bit data bus.†

A stream read requires the same steps as the block read described in *Single-Block or Multiple-Block Read*, except for the following bits in the `cmd` register:†

- `transfer_mode` = 0x1 (for stream transfer)†
- `cmd_index` = 20 (SD/SDIO CMD20)†

A stream write requires the same steps as the block write mentioned in *Single-Block or Multiple-Block Write*, except for the following bits in the `cmd` register:†

- `transfer_mode` = 0x1 (for stream transfer)†
- `cmd_index` = 11 (SD/SDIO CMD11)†

Related Information

- **Single-Block or Multiple-Block Read** on page 14-45
  Refer to this section for more information about a stream read.
- **Single-Block or Multiple-Block Write** on page 14-48
  Refer to this section for more information about a stream write.

Packed Commands

To reduce overhead, read and write commands can be packed in groups of commands—either all read or all write—that transfer the data for all commands in the group in one transfer on the bus. Use the SD/SDIO SET_BLOCK_COUNT (CMD23) command to state ahead of time how many blocks will be transferred. Then issue a single READ_MULTIPLE_BLOCK or WRITE_MULTIPLE_BLOCK command to read or write multiple blocks.†

- SET_BLOCK_COUNT—set block count (number of blocks transferred using the READ_MULTIPLE_BLOCK or WRITE_MULTIPLE_BLOCK command) †
- READ_MULTIPLE_BLOCK—multiple-block read command †
- WRITE_MULTIPLE_BLOCK—multiple-block write command†

Packed commands are organized in packets by the application software and are transparent to the controller.†

Related Information

www.jedec.org
For more information about packed commands, refer to JEDEC Standard No. 84-A441, available on the JEDEC website.

Transfer Stop and Abort Commands

This section describes stop and abort commands. The SD/SDIO STOP_TRANSMISSION command can terminate a data transfer between a memory card and the controller. The ABORT command can terminate an I/O data transfer for only an SDIO card. †

STOP_TRANSMISSION (CMD12)

The host can send the STOP_TRANSMISSION (CMD12) command on the CMD pin at any time while a data transfer is in progress. Perform the following steps to send the STOP_TRANSMISSION command to the SD/SDIO card device:†

1. Set the `wait_prvdata_complete` bit of the `cmd` register to 0.†
2. Set the `stop_abort_cmd` in the `cmd` register to 1, which ensures that the CIU stops.†

The STOP_TRANSMISSION command is a non-data transfer command.†

Related Information

Non-Data Transfer Commands on page 14-42
Refer to this section for information on the STOP_TRANSMISSION command.

ABORT

The ABORT command can only be used with SDIO cards. To abort the function that is transferring data, program the ABORT function number in the ASx[2:0] bits at address 0x06 of the card common control register (CCCR) in the card device, using the IO_RW_DIRECT (CMD52) command. The CCCR is located at the base of the card space 0x00 – 0xFF.†

Note: The ABORT command is a non-data transfer command.†

Related Information

Non-Data Transfer Commands on page 14-42
Refer to this section for information on the ABORT command.

Sending the ABORT Command

Perform the following steps to send the ABORT command to the SDIO card device:†

1. Set the `cmdarg` register to include the appropriate command argument parameters listed in *cmdarg Register Settings for SD/SDIO ABORT Command*.†
2. Send the IO_RW_DIRECT command by setting the following fields of the `cmd` register:†
   - Set the command index to 0x52 (IO_RW_DIRECT).†
   - Set the `stop_abort_cmd` bit of the `cmd` register to 1 to inform the controller that the host aborted the data transfer.†
   - Set the `wait_prvdata_complete` bit of the `cmd` register to 0.†
3. Wait for the `cmd` bit in the `rintsts` register to change to 1.†
4. Read the response to the IO_RW_DIRECT command (R5) in the response registers for any errors.†

For more information about response values, refer to the Physical Layer Simplified Specification, Version 3.01, available on the SD Association website.

Related Information

www.sdcard.org
To learn more about how SD technology works, visit the SD Association website.

cmdarg Register Settings for SD/SDIO ABORT Command†

Table 14-23: cmdarg Register Settings for SD/SDIO ABORT Command

Bits	Contents	Value
31	R/W flag	1
30:28	Function number	0, for access to the CCCR in the card device

Bits	Contents	Value
27	RAW flag	1, if needed to read after write
26	Don't care	-
25:9	Register address	0x06
8	Don't care	-
7:0	Write data	Function number to abort

Internal DMA Controller Operations

For better performance, you can use the internal DMA controller to transfer data between the host and the controller. This section describes the internal DMA controller’s initialization process, and transmission sequence, and reception sequence.

Internal DMA Controller Initialization

To initialize the internal DMA controller, perform the following steps:†

1. Set the required `bmod` register bits: †
   - If the internal DMA controller enable bit (`de`) of the `bmod` register is set to 0 during the middle of a DMA transfer, the change has no effect. Disabling only takes effect for a new data transfer command.†
   - Issuing a software reset immediately terminates the transfer. Prior to issuing a software reset, Altera recommends the host reset the DMA interface by setting the `dma_reset` bit of the `ctrl` register to 1.†
   - The `pbl` field of the `bmod` register is read-only and a direct reflection of the contents of the DMA multiple transaction size field (`dw_dma_multiple_transaction_size`) in the `fifoth` register.†
   - The `fb` bit of the `bmod` register has to be set appropriately for system performance.†
2. Write to the `idinten` register to mask unnecessary interrupt causes according to the following guidelines:†
   - When a Descriptor Unavailable interrupt is asserted, the software needs to form the descriptor, appropriately set its own bit, and then write to the poll demand register (`pldmnd`) for the internal DMA controller to re-fetch the descriptor.†
   - It is always appropriate for the software to enable abnormal interrupts because any errors related to the transfer are reported to the software.†
3. Populate either a transmit or receive descriptor list in memory. Then write the base address of the first descriptor in the list to the internal DMA controller’s descriptor list base address register (`dbaddr`). The DMA controller then proceeds to load the descriptor list from memory. *Internal DMA Controller Transmission Sequences* and *Internal DMA Controller Reception Sequences* describe this step in detail. †

Related Information

- **Internal DMA Controller Transmission Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Transmission Sequences.
- **Internal DMA Controller Reception Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Reception Sequences.

Internal DMA Controller Transmission Sequences

To use the internal DMA controller to transmit data, perform the following steps:

1. The host sets up the Descriptor fields (DES0—DES3) for transmission and sets the OWN bit (DES0[31]) to 1. The host also loads the data buffer in system memory with the data to be written to the SD card.†
2. The host writes the appropriate write data command (SD/SDIO WRITE_BLOCK or WRITE_MULTIPLE_BLOCK) to the `cmd` register. The internal DMA controller determines that a write data transfer needs to be performed.†
3. The host sets the required transmit threshold level in the `tx_wmark` field in the `fifoth` register.†
4. The internal DMA controller engine fetches the descriptor and checks the OWN bit. If the OWN bit is set to 0, the host owns the descriptor. In this case, the internal DMA controller enters the suspend state and asserts the Descriptor Unable interrupt. The host then needs to set the descriptor OWN bit to 1 and release the DMA controller by writing any value to the `pldmnd` register.†
5. The host must write the descriptor base address to the `dbaddr` register.†
6. The internal DMA controller waits for the Command Done (`CD`) bit in the `rintsts` register to be set to 1, with no errors from the BIU. This condition indicates that a transfer can be done.†
7. The internal DMA controller engine waits for a DMA interface request from BIU. The BIU divides each transfer into smaller chunks. Each chunk is an internal request to the DMA. This request is generated based on the transmit threshold value.†
8. The internal DMA controller fetches the transmit data from the data buffer in the system memory and transfers the data to the FIFO buffer in preparation for transmission to the card.†
9. When data spans across multiple descriptors, the internal DMA controller fetches the next descriptor and continues with its operation with the next descriptor. The Last Descriptor bit in the descriptor DES0 field indicates whether the data spans multiple descriptors or not.†
10. When data transmission is complete, status information is updated in the `idsts` register by setting the `ti` bit to 1, if enabled. Also, the OWN bit is set to 0 by the DMA controller by updating the DES0 field of the descriptor.†

Internal DMA Controller Reception Sequences

To use the internal DMA controller to receive data, perform the following steps:

1. The host sets up the descriptor fields (DES0—DES3) for reception, sets the OWN (DES0 [31]) to 1.†
2. The host writes the read data command to the `cmd` register in BIU. The internal DMA controller determines that a read data transfer needs to be performed.†
3. The host sets the required receive threshold level in the `rx_wmark` field in the `fifoth` register.†
4. The internal DMA controller engine fetches the descriptor and checks the OWN bit. If the OWN bit is set to 0, the host owns the descriptor. In this case, the internal DMA controller enters suspend state and asserts the Descriptor Unable interrupt. The host then needs to set the descriptor OWN bit to 1 and release the DMA controller by writing any value to the `pldmnd` register.†
5. The host must write the descriptor base address to the `dbaddr` register.†
6. The internal DMA controller waits for the `CD` bit in the `rintsts` register to be set to 1, with no errors from the BIU. This condition indicates that a transfer can be done.†
7. The internal DMA controller engine waits for a DMA interface request from the BIU. The BIU divides each transfer into smaller chunks. Each chunk is an internal request to the DMA. This request is generated based on the receive threshold value.†

8. The internal DMA controller fetches the data from the FIFO buffer and transfers the data to system memory.†
9. When data spans across multiple descriptors, the internal DMA controller fetches the next descriptor and continues with its operation with the next descriptor. The Last Descriptor bit in the descriptor indicates whether the data spans multiple descriptors or not.†
10. When data reception is complete, status information is updated in the `idsts` register by setting the `ri` bit to 1, if enabled. Also, the OWN bit is set to 0 by the DMA controller by updating the DES0 field of the descriptor.†

Commands for SDIO Card Devices

This section describes the commands to temporarily halt the transfers between the controller and SDIO card device.

Suspend and Resume Sequence

For SDIO cards, a data transfer between an I/O function and the controller can be temporarily halted using the SUSPEND command. This capability might be required to perform a high-priority data transfer with another function. When desired, the suspended data transfer can be resumed using the RESUME command.†

The SUSPEND and RESUME operations are implemented by writing to the appropriate bits in the CCCR (Function 0) of the SDIO card. To read from or write to the CCCR, use the controller's IO_RW_DIRECT command.†

Suspend

To suspend data transfer, perform the following steps:†

1. Check if the SDIO card supports the SUSPEND/RESUME protocol. This can be done through the SBS bit in the CCCR at offset 0x08 of the card.†
2. Check if the data transfer for the required function number is in process. The function number that is currently active is reflected in the function select bits (FSx) of the CCCR, bits 3:0 at offset 0x0D of the card.†

   Note: If the bus status bit (BS), bit 0 at address 0xC, is 1, only the function number given by the FSx bits is valid.†
3. To suspend the transfer, set the bus release bit (BR), bit 2 at address 0xC, to 1.†
4. Poll the BR and BS bits of the CCCR at offset 0x0C of the card until they are set to 0. The BS bit is 1 when the currently-selected function is using the data bus. The BR bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function is suspended.†
5. During a read-data transfer, the controller can be waiting for the data from the card. If the data transfer is a read from a card, the controller must be informed after the successful completion of the SUSPEND command. The controller then resets the data state machine and comes out of the wait state. To accomplish this, set the abort read data bit (`abort_read_data`) in the `ctrl` register to 1.†
6. Wait for data completion, by polling until the `dto` bit is set to 1 in the `rintsts` register. To determine the number of pending bytes to transfer, read the transferred CIU card byte count (`tcbcnt`) register of the controller. Subtract this value from the total transfer size. You use this number to resume the transfer properly.†

Resume

To resume the data transfer, perform the following steps:†

1. Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.†
2. If the card is in a disconnect state, select it using the SD/SDIO SELECT/DESELECT_CARD command. The card status can be retrieved in response to an IO_RW_DIRECT or IO_RW_EXTENDED command.†
3. Check that a function to be resumed is ready for data transfer. Determine this state by reading the corresponding RF<*n*> flag in CCCR at offset 0x0F of the card. If RF<*n*> = 1, the function is ready for data transfer.†

   Note: For detailed information about the RF<*n*> flags, refer to SDIO Simplified Specification Version 2.00, available on the SD Association website.†
4. To resume transfer, use the IO_RW_DIRECT command to write the function number at the FSx bits in the CCCR, bits 3:0 at offset 0x0D of the card. Form the command argument for the IO_RW_DIRECT command and write it to the `cmdarg` register. Bit values are listed in the following table.†

Table 14-24: cmdarg Bit Values for RESUME Command†

Bits	Content	Value
31	R/W flag	1
30:28	Function number	0, for CCCR access
27	RAW flag	1, read after write
26	Don't care	-
25:9	Register address	0x0D
8	Don't care	-
7:0	Write data	Function number that is to be resumed

5. Write the block size value to the `blksiz` register. Data is transferred in units of this block size.†
6. Write the byte count value to the `bytcnt` register. Specify the total size of the data that is the remaining bytes to be transferred. It is the responsibility of the software to handle the data.†

   To determine the number of pending bytes to transfer, read the transferred CIU card byte count register (`tcbcnt`). Subtract this value from the total transfer size to calculate the number of remaining bytes to transfer.†
7. Write to the `cmd` register similar to a block transfer operation. When the `cmd` register is written, the command is sent and the function resumes data transfer. For more information, refer to *Single-Block or Multiple-Block Read* and *Single-Block or Multiple-Block Write*.†
8. Read the resume data flag (DF) of the SDIO card device. Interpret the DF flag as follows:†

- DF=1—The function has data for the transfer and begins a data transfer as soon as the function or memory is resumed.†
- DF=0—The function has no data for the transfer. If the data transfer is a read, the controller waits for data. After the data timeout period, it issues a data timeout error.†

Related Information

- **www.sdcard.org**
  To learn more about how SD technology works, visit the SD Association website.
- **Single-Block or Multiple-Block Read** on page 14-45
  Refer to this section for more information about writing to the `cmd` register.
- **Single-Block or Multiple-Block Write** on page 14-48
  Refer to this section for more information about writing to the `cmd` register.

Read-Wait Sequence

Read_wait is used with SDIO cards only. It temporarily stalls the data transfer, either from functions or memory, and allows the host to send commands to any function within the SDIO card device. The host can stall this transfer for as long as required. The controller provides the facility to signal this stall transfer to the card.†

Signalling a Stall

To signal the stall, perform the following steps:†

1. Check if the card supports the read_wait facility by reading the SDIO card's SRW bit, bit 2 at offset 0x8 in the CCCR.†
2. If this bit is 1, all functions in the card support the read_wait facility. Use the SD/SDIO IO_RW_DIRECT command to read this bit.†
3. If the card supports the `read_wait` signal, assert it by setting the read wait bit (`read_wait`) in the `ctrl` register to 1.†
4. Reset the `read_wait` bit to 0 in the `ctrl` register.†

CE-ATA Data Transfer Commands

This section describes CE-ATA data transfer commands.

Related Information

Data Transfer Commands on page 14-44
Refer to this section for information about the basic settings and interrupts generated for different conditions.

Reset and Card Device Discovery Overview

Before starting any CE-ATA operations, the host must perform a MMC reset and initialization procedure. The host and card device must negotiate the MMC transfer (MMC TRAN) state before the card enters the MMC TRAN state.†

The host must follow the existing MMC discovery procedure to negotiate the MMC TRAN state. After completing normal MMC reset and initialization procedures, the host must query the initial ATA task file values using the RW_REG or CMD39 command.†

By default, the MMC block size is 512 bytes—indicated by bits 1:0 of the srcControl register inside the CE-ATA card device. The host can negotiate the use of a 1 KB or 4 KB MMC block sizes. The card indicates MMC block sizes that it can support through the srcCapabilities register in the MMC; the host reads this register to negotiate the MMC block size. Negotiation is complete when the host controller writes the MMC block size into the srcControl register bits 1:0 of the card.†

Related Information

www.jedec.org
For information about the (MMC TRAN) state, MMC reset and initialization, refer to JEDEC Standard No. 84-A441, available on the JEDEC website.

ATA Task File Transfer Overview

ATA task file registers are mapped to addresses 0x00h through 0x10h in the MMC register space. The RW_REG command is used to issue the ATA command, and the ATA task file is transmitted in a single RW_REG MMC command sequence.†

The host software stack must write the task file image to the FIFO buffer before setting the `cmdarg` and `cmd` registers in the controller. The host processor then writes the address and byte count to the `cmdarg` register before setting the `cmd` register bits.†

For the RW_REG command, there is no CCS from the CE-ATA card device. †

ATA Task File Transfer Using the RW_MULTIPLE_REGISTER (RW_REG) Command

This command involves data transfer between the CE-ATA card device and the controller. To send a data command, the controller needs a command argument, total data size, and block size. Software receives or sends data through the FIFO buffer.†

Implementing ATA Task File Transfer

To implement an ATA task file transfer (read or write), perform the following steps:†

1. Write the data size in bytes to the `bytcnt` register. `bytcnt` must equal the block size, because the controller expects a single block transfer.†
2. Write the block size in bytes to the `blksiz` register.†
3. Write the `cmdarg` register with the beginning register address.†

You must set the `cmdarg`, `cmd`, `blksiz`, and `bytcnt` registers according to the tables in *Register Settings for ATA Task File Transfer*.†

Related Information

Register Settings for ATA Task File Transfer on page 14-58
Refer to this table for information on how to set these registers.

Register Settings for ATA Task File Transfer

Table 14-25: cmdarg Register Settings for ATA Task File Transfer†

Bit	Value	Comment
31	1 or 0	Set to 0 for read operation or set to 1 for write operation
30:24	0	Reserved (bits set to 0 by host processor)
23:18	0	Starting register address for read or write (DWORD aligned)
17:16	0	Register address (DWORD aligned)

Bit	Value	Comment
15:8	0	Reserved (bits set to 0 by host processor)
7:2	16	Number of bytes to read or write (integral number of DWORD)
1:0	0	Byte count in integral number of DWORD

Table 14-26: cmd Register Settings for ATA Task File Transfer†

Bit	Value	Comment
`start_cmd`	1	
`ccs_expected`	0	CCS is not expected
`read_ceata_device`	0 or 1	Set to 1 if RW_BLK or RW_REG read
`update_clk_regs_only`	0	No clock parameters update command
`card_num`	0	
`send_initialization`	0	No initialization sequence
`stop_abort_cmd`	0	
`send_auto_stop`	0	
`transfer_mode`	0	Block transfer mode. Block size and byte count must match number of bytes to read or write
`read_write`	1 or 0	1 for write and 0 for read
`data_expected`	1	Data is expected
`response_length`	0	
`response_expect`	1	
`cmd_index`	Command index	Set this parameter to the command number. For example, set to 24 for SD/SDIO WRITE_BLOCK (CMD24) or 25 for WRITE_MULTIPLE_BLOCK (CMD25).
`wait_prvdata_complete`	1	• 0 for send command immediately • 1 for send command after previous DTO interrupt
`check_response_crc`	1	• 0 for not checking response CRC • 1 for checking response CRC

Table 14-27: blksiz Register Settings for ATA Task File Transfer†

Bit	Value	Comment
31:16	0	Reserved bits set to 0
15:0 (`block_size`)	16	For accessing entire task file (16, 8-bit registers). Block size of 16 bytes

Table 14-28: bytcnt Register Settings for ATA Task File Transfer

Bit	Value	Comment
31:0	16	For accessing entire task file (16, 8-bit registers). Byte count value of 16 is used with the block size set to 16.

ATA Payload Transfer Using the RW_MULTIPLE_BLOCK (RW_BLK) Command

This command involves data transfer between the CE-ATA card device and the controller. To send a data command, the controller needs a command argument, total data size, and block size. Software receives or sends data through the FIFO buffer. †

Implementing ATA Payload Transfer

To implement an ATA payload transfer (read or write), perform the following steps:†

1. Write the data size in bytes to the `bytcnt` register.†
2. Write the block size in bytes to the `blksiz` register. The controller expects a single/multiple block transfer.†
3. Write to the `cmdarg` register to indicate the data unit count.†

Register Settings for ATA Payload Transfer

You must set the `cmdarg`, `cmd`, `blksiz`, and `bytcnt` registers according to the following tables.†

Table 14-29: cmdarg Register Settings for ATA Payload Transfer†

Bits	Value	Comment
31	1 or 0	Set to 0 for read operation or set to 1 for write operation
30:24	0	Reserved (bits set to 0 by host processor)
23:16	0	Reserved (bits set to 0 by host processor)
15:8	Data count	Data Count Unit [15:8]
7:0	Data count	Data Count Unit [7:0]

Table 14-30: cmd Register Settings for ATA Payload Transfer†

Bits	Value	Comment
start_cmd	1	-
ccs_expected	1	CCS is expected. Set to 1 for the RW_BLK command if interrupts are enabled in CE-ATA card device (the nIEN bit is set to 0 in the ATA control register)
read_ceata_device	0 or 1	Set to 1 for a RW_BLK or RW_REG read command
update_clk_regs_only	0	No clock parameters update command
card_num	0	-
send_initialization	0	No initialization sequence
stop_abort_cmd	0	-
send_auto_stop	0	-
transfer_mode	0	Block transfer mode. Byte count must be integer multiple of 4kB. Block size can be 512, 1k or 4k bytes
read_write	1 or 0	1 for write and 0 for read
data_expected	1	Data is expected
response_length	0	-
response_expect	1	-
cmd_index	Command index	Set this parameter to the command number. For example, set to 24 for SD/SDIO WRITE_BLOCK (CMD24) or 25 for WRITE_MULTIPLE_BLOCK (CMD25).
wait_prvdata_complete	1	• 0 for send command immediately • 1 for send command after previous DTO interrupt
check_response_crc	1	• 0 for not checking response CRC • 1 for checking response CRC

Table 14-31: blksiz Register Settings for ATA Payload Transfer†

Bits	Value	Comment
31:16	0	Reserved bits set to 0
15:0 (block_size)	512, 1024 or 4096	MMC block size can be 512, 1024 or 4096 bytes as negotiated by host

Table 14-32: bytcnt Register Settings for ATA Payload Transfer

Bits	Value	Comment
31:0	<n>*block_size	Byte count must be an integer multiple of the block size. For ATA media access commands, byte count must be a multiple of 4 KB. (<*n*>*block_size = <*x*>*4 KB, where <*n*> and <*x*> are integers)

CE-ATA CCS

This section describes disabling the CCS, recovery after CCS timeout, and recovery after I/O read transmission delay (N_{ACIO}) timeout. †

Disabling the CCS

While waiting for the CCS for an outstanding RW_BLK command, the host can disable the CCS by sending a CCSD command:†

- Send a CCSD command—the controller sends the CCSD command to the CE-ATA card device if the send_ccsd bit is set to 1 in the ctrl register of the controller. This bit can be set only after a response is received for the RW_BLK command.†
- Send an internal stop command—send an internally-generated SD/SDIO STOP_TRANSMISSION (CMD12) command after sending the CCSD pattern. If the send_auto_stop_ccsd bit of the ctrl register is also set to 1 when the controller is set to send the CCSD pattern, the controller sends the internally-generated STOP command to the CMD pin. After sending the STOP command, the controller sets the acd bit in the rintsts register to 1.†

Recovery after CCS Timeout

If a timeout occurs while waiting for the CCS, the host needs to send the CCSD command followed by a STOP command to abort the pending ATA command. The host can set up the controller to send an internally-generated STOP command after sending the CCSD pattern:†

- Send CCSD command—set the `send_ccsd` bit in the `ctrl` register to 1.†
- Send external STOP command—terminate the data transfer between the CE-ATA card device and the controller. For more information about sending the STOP command, refer to *Transfer Stop and Abort Commands*.†
- Send internal STOP command—set the `send_auto_stop_ccsd` bit in the `ctrl` register to 1, which tells the controller to send the internally-generated STOP command. After sending the STOP command, the controller sets the `acd` bit in the `rintsts` register to 1. The `send_auto_stop_ccsd` bit must be set to 1 along with setting the `send_ccsd` bit.†

Related Information

Transfer Stop and Abort Commands on page 14-51
Refer to this section for more information about sending the STOP command.

Recovery after I/O Read Transmission Delay (N_{ACIO}) Timeout

If the I/O read transmission delay (N_{ACIO}) timeout occurs for the CE-ATA card device, perform one of the following steps to recover from the timeout:†

- If the CCS is expected from the CE-ATA card device (that is, the `ccs_expected` bit is set to 1 in the `cmd` register), follow the steps in *Recovery after CCS Timeout*.†
- If the CCS is not expected from the CE-ATA card device, perform the following steps: †

  1. Send an external STOP command. †
  2. Terminate the data transfer between the controller and CE-ATA card device. †

Related Information

Recovery after CCS Timeout on page 14-62
For more information about what steps to take if the CCS is expected from the CE-ATA card device.

Reduced ATA Command Set

It is necessary for the CE-ATA card device to support the reduced ATA command subset. This section describes the reduced command set.†

The IDENTIFY DEVICE Command

The IDENTIFY DEVICE command returns a 512-byte data structure to the host that describes device-specific information and capabilities. The host issues the IDENTIFY DEVICE command only if the MMC block size is set to 512 bytes. Any other MMC block size has indeterminate results.†

The host issues a RW_REG command for the ATA command, and the data is retrieved with the RW_BLK command.†

The host controller uses the following settings while sending a RW_REG command for the IDENTIFY DEVICE ATA command. The following list shows the primary bit settings:†

- `cmd` register setting: `data_expected` bit set to 0†
- `cmdarg` register settings: †
  - Bit [31] set to 0†
  - Bits [7:2] set to 128 †
  - All other bits set to 0†
- Task file settings: †
  - Command field of the ATA task file set to 0xEC†
  - Reserved fields of the task file set to 0†
- `bytcnt` register and `block_size` field of the `blksiz` register: set to 16†

The host controller uses the following settings for data retrieval (RW_BLK command):†

- `cmd` register settings:†
  - `ccs_expected` set to 1†
  - `data_expected` set to 1†
- `cmdarg` register settings: †
  - Bit [31] set to 0 (read operation) †
  - Bits [15:0] set to 1 (data unit count = 1)†
  - All other bits set to 0†
- `bytcnt` register and `block_size` field of the `blksiz` register: set to 512†

The READ DMA EXT Command

The READ DMA EXT command reads a number of logical blocks of data from the card device using the Data-In data transfer protocol. The host uses a RW_REG command to issue the ATA command and the RW_BLK command for the data transfer.†

The WRITE DMA EXT Command

The WRITE DMA EXT command writes a number of logical blocks of data to the card device using the Data-Out data transfer protocol. The host uses a RW_REG command to issue the ATA command and the RW_BLK command for the data transfer.†

The STANDBY IMMEDIATE Command

This ATA command causes the card device to immediately enter the most aggressive power management mode that still retains internal device context. No data transfer (RW_BLK) is expected for this command.†

For card devices that do not provide a power savings mode, the STANDBY IMMEDIATE command returns a successful status indication. The host issues a RW_REG command for the ATA command, and the status is retrieved with the SD/SDIO CMD39 or RW_REG command. Only the status field of the ATA task file contains the success status; there is no error status.†

The host controller uses the following settings while sending the RW_REG command for the STANDBY IMMEDIATE ATA command: †

- `cmd` register setting: `data_expected` bit set to 0†
- `cmdarg` register settings: †
  - Bit [31] set to 1 †
  - Bits [7:2] set to 4 †
  - All other bits set to 0 †
- Task file settings: †
  - Command field of the ATA task file set to 0xE0†
  - Reserved fields of the task file set to 0†
- `bytcnt` register and `block_size` field of the `blksiz` register: set to 16 †

The FLUSH CACHE EXT Command

For card devices that buffer/cache written data, the FLUSH CACHE EXT command ensures that buffered data is written to the card media. For cards that do not buffer written data, the FLUSH CACHE EXT command returns a success status. No data transfer (RW_BLK) is expected for this ATA command. †

The host issues a RW_REG command for the ATA command, and the status is retrieved with the SD/SDIO CMD39 or RW_REG command. There can be error status for this ATA command, in which case fields other than the status field of the ATA task file are valid.†

The host controller uses the following settings while sending the RW_REG command for the STANDBY IMMEDIATE ATA command:†

- cmd register setting: data_expected bit set to 0 †
- cmdarg register settings: †
    - Bit [31] set to 1 †
    - Bits [7:2] set to 4†
    - All other bits set to 0†
- Task file settings: †
    - Command field of the ATA task file set to 0xEA †
    - Reserved fields of the task file set to 0†
- bytcnt register and block_size field of the blksiz register: set to 16 †

Card Read Threshold

When an application needs to perform a single or multiple block read command, the application must set the cardthrctl register with the appropriate card read threshold size in the card read threshold field (cardrdthreshold) and set the cardrdthren bit to 1. This additional information specified in the controller ensures that the controller sends a read command only if there is space equal to the card read threshold available in the RX FIFO buffer. This in turn ensures that the card clock is not stopped in the middle a block of data being transmitted from the card. Set the card read threshold to the block size of the transfer to guarantee there is a minimum of one block size of space in the RX FIFO buffer before the controller enables the card clock. †

The card read threshold is required when the round trip delay is greater than half of sdmmc_clk_divided.†

Table 14-33: Card Read Threshold Guidelines†

Bus Speed Modes	Round Trip Delay (Delay_R) [39]	Is Stopping of Card Clock Allowed?	Card Read Threshold Required?
SDR25	Delay_R > 0.5 * (sdmmc_clk/4)	No	Yes
	Delay_R < 0.5 * (sdmmc_clk/4)	Yes	No
SDR12	Delay_R > 0.5 * (sdmmc_clk/4)	No	Yes
	Delay_R < 0.5 * (sdmmc_clk/4)	Yes	No

Related Information

http://www.altera.com/literature/hb/cyclone-v/cv_51002.pdf

[39] Delay_R = Delay_O + tODLY + Delay_I †

Where: †

Delay_O = sdmmc_clk to sdmmc_cclk_out delay (including I/O pin delay) †

Delay_I = Input I/O pin delay + routing delay to the input register †

tODLY = sdmmc_cclk_out to card output delay (varies across card manufactures and speed modes) †

For the delay numbers needed for above calculation, refer to Arria 10 Datasheet. †

Recommended Usage Guidelines for Card Read Threshold

1. The `cardthrctl` register must be set before setting the `cmd` register for a data read command.†
2. The `cardthrctl` register must not be set while a data transfer command is in progress.†
3. The `cardrdthreshold` field of the `cardthrctl` register must be set to at the least the block size of a single or multiblock transfer. A `cardrdthreshold` field setting greater than or equal to the block size of the read transfer ensures that the card clock does not stop in the middle of a block of data.†
4. If the round trip delay is greater than half of the card clock period, card read threshold must be enabled and the card threshold must be set as per guideline 3 to guarantee that the card clock does not stop in the middle of a block of data.†
5. If the `cardrdthreshold` field is set to less than the block size of the transfer, the host must ensure that the receive FIFO buffer never overflows during the read transfer. Overflow can cause the card clock from the controller to stop. The controller is not able to guarantee that the card clock does not stop during a read transfer.†

Note: If the `cardrdthreshold` field of the `cardthrctl` register, and the `rx_wmark` and `dw_dma_multiple_transaction_size` fields of the `fifoth` register are set incorrectly, the card clock might stop indefinitely, with no interrupts generated by the controller.†

Card Read Threshold Programming Sequence

Most cards, such as SDHC or SDXC, support block sizes that are either specified in the card or are fixed to 512 bytes. For SDIO, MMC, and standard capacity SD cards that support partial block read (READ_BL_PARTIAL set to 1 in the CSD register of the card device), the block size is variable and can be chosen by the application.†

To use the card read threshold feature effectively and to guarantee that the card clock does not stop because of a FIFO Full condition in the middle of a block of data being read from the card, follow these steps:†

1. Choose a block size that is a multiple of four bytes.†
2. Enable card read threshold feature. The card read threshold can be enabled only if the block size for the given transfer is less than or equal to the total depth of the FIFO buffer:†

   (block size / 4) ≤ 1024†
3. Choose the card read threshold value: †
   - If (block size / 4) ≥ 512, choose `cardrdthreshold` such that:†
     - `cardrdthreshold` ≤ (block size / 4) in bytes†
   - If (block size / 4) < 512, choose `cardrdthreshold` such that:†
     - `cardrdthreshold` = (block size / 4) in bytes†
4. Set the `dw_dma_multiple_transaction_size` field in the `fifoth` register to the number of transfers that make up a DMA transaction. For example, size = 1 means 4 bytes are moved. The possible values for the size are 1, 4, 8, 16, 32, 64, 128, and 256 transfers. Select the size so that the value (block size / 4) is evenly divided by the size.†
5. Set the `rx_wmark` field in the `fifoth` register to the size – 1.†

For example, if your block size is 512 bytes, legal values of `dw_dma_multiple_transaction_size` and `rx_wmark` are listed in the following table.

Table 14-34: Legal Values of dw_dma_multiple_transaction_size and rx_wmark for Block Size = 512†

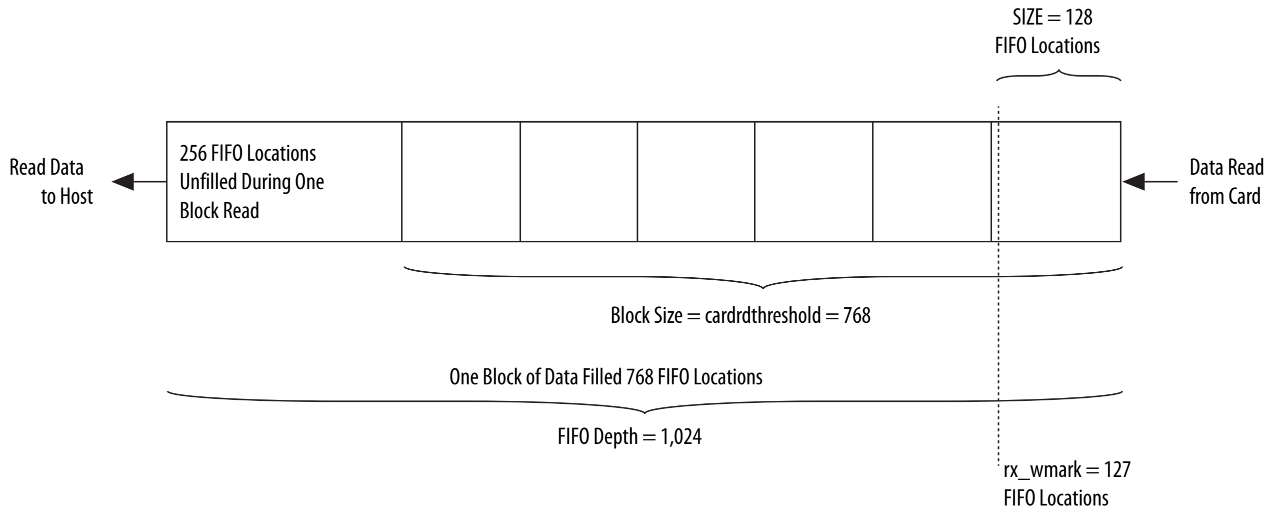
Block Size	dw_dma_multiple_transaction_size	rx_wmark
512	1	0
512	4	3
512	8	7
512	16	15
512	32	31
512	64	63
512	128	127

Card Read Threshold Programming Examples

This section shows examples of how to program the card read threshold.†

- Choose a block size that is a multiple of 4 (the number of bytes per FIFO location), and less than 4096 (1024 FIFO locations). For example, a block size of 3072 bytes is legal, because 3072 / 4 = 768 FIFO locations.†
- For DMA mode, choose the size so that block size is a multiple of the size. For example size = 128, where block size%size = 0.†
- Set the `rx_wmark` field = size – 1. For example, the `rx_wmark` field = 128 – 1 = 127.†
- Because block size > ½ FifoDepth, set the `cardrdthreshold` field to the block size. For example, the `cardrdthreshold` field = 3072 bytes.†

Figure 14-14: FIFO Buffer content when Card Read Threshold is set to 768†



Interrupt and Error Handling

This section describes how to use interrupts to handle errors. On power-on or reset, interrupts are disabled (the `int_enable` bit in the `ctrl` register is set to 0), and all the interrupts are masked (the `intmask` register default is 0). The controller error handling includes the following types of errors:

- Response and data timeout errors—For response time-outs, the host software can retry the command. For data time-outs, the controller has not received the data start bit from the card, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the `tcbcnt` register later, the software can decide how many bytes remain to be copied (read). †
- Response errors—Set to 1 when an error is received during response reception. If the response received is invalid, the software can retry the command. †
- Data errors—Set to 1 when a data receive error occurs. Examples of data receive errors: †
  - Data CRC†
  - Start bit not found †
  - End bit not found †

  These errors can be occur on any block. On receipt of an error, the software can issue an SD/SDIO STOP or SEND_IF_COND command, and retry the command for either the whole data or partial data.†
- Hardware locked error—Set to 1 when the controller cannot load a command issued by software. When software sets the `start_cmd` bit in the `cmd` register to 1, the controller tries to load the command. If the command buffer already contains a command, this error is raised, and the new command is discarded, requiring the software to reload the command.†
- FIFO buffer underrun/overrun error—If the FIFO buffer is full and software tries to write data to the FIFO buffer, an overrun error is set. Conversely, if the FIFO buffer is empty and the software tries to read data from the FIFO buffer, an underrun error is set. Before reading or writing data in the FIFO buffer, the software must read the FIFO buffer empty bit (`fifo_empty`) or FIFO buffer full bit (`fifo_full`) in the `status` register.†
- Data starvation by host timeout—This condition occurs when software does not service the FIFO buffer fast enough to keep up with the controller. Under this condition and when a read transfer is in process, the software must read data from the FIFO buffer, which creates space for further data reception. When a transmit operation is in process, the software must write data to fill the FIFO buffer so that the controller can write the data to the card.†
- CE-ATA errors†
- CRC error on command—If a CRC error is detected for a command, the CE-ATA card device does not send a response, and a response timeout is expected from the controller. The ATA layer is notified that an MMC transport layer error occurred.

- CRC error on command—If a CRC error is detected for a command, the CE-ATA card device does not send a response, and a response timeout is expected from the controller. The ATA layer is notified that an MMC transport layer error occurred.†
- Write operation—Any MMC transport layer error known to the card device causes an outstanding ATA command to be terminated. The ERR bits are set in the ATA status registers and the appropriate error code is sent to the Error Register (Error) on the ATA card device.†

  If the device interrupt bit of the CE-ATA card (the nIEN bit in the ATA control register) is set to 0, the CCS is sent to the host.†

  If the device interrupt bit is set to 1, the card device completes the entire data unit count if the host controller does not abort the ongoing transfer.†

  Note: During a multiple-block data transfer, if a negative CRC status is received from the card device, the data path signals a data CRC error to the BIU by setting the `dcrc` bit in the `rintsts` register to 1. It then continues further data transmission until all the bytes are transmitted.†
- Read operation—If MMC transport layer errors are detected by the host controller, the host completes the ATA command with an error status. The host controller can issue a CCSD command followed by a STOP_TRANSMISSION (CMD12) command to abort the read transfer. The host can also transfer the entire data unit count bytes without aborting the data transfer.†

Booting Operation for eMMC and MMC

This section describes how to set up the controller for eMMC and MMC boot operation.

Boot Operation by Holding Down the CMD Line

The controller can boot from MMC4.3, MMC4.4, and MMC4.41 cards by holding down the CMD line.

For information about this boot method, refer to the following specifications, available on the JEDEC website:

- JEDEC Standard No. 84-A441
- JEDEC Standard No. 84-A44
- JEDEC Standard No. JESD84-A43

Related Information

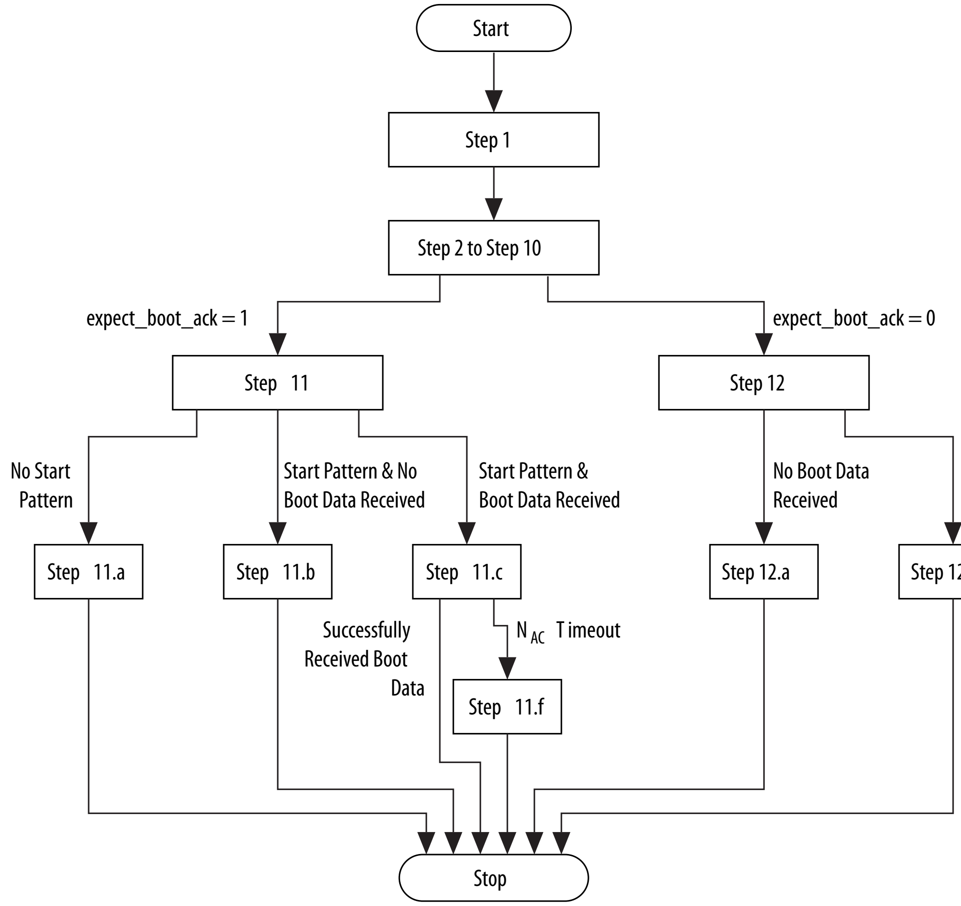
www.jedec.org

For more information about this boot method, refer to the following JEDEC Standards available on the JEDEC website: No. 84-A441, No. 84-A44, and No. JESD84-A43.

Boot Operation for eMMC Card Device

The following figure illustrates the steps to perform the boot process for eMMC card devices. The detailed steps are described following the flow chart.

Figure 14-15: Flow for eMMC Boot Operation†



1. The software driver performs the following checks: †
   - If the eMMC card device supports boot operation (the BOOT_PARTITION_ENABLE bit is set to 1 in the EXT_CSD register of the eMMC card).†
   - The BOOT_SIZE_MULT and BOOT_BUS_WIDTH values in the EXT_CSD register, to be used during the boot process.†
2. The software sets the following bits: †
   - Sets masks for interrupts, by setting the appropriate bits to 0 in the `intmask` register.†
   - Sets the global `int_enable` bit of the `ctrl` register to 1. Other bits in the `ctrl` register must be set to 0. †

     Note: Altera recommends that you write 0xFFFFFFFF to the `rintsts` and `idsts` registers to clear any pending interrupts before setting the `int_enable` bit. For internal DMA controller mode, the software driver needs to unmask all the relevant fields in the `idinten` register.†
3. If the software driver needs to use the internal DMA controller to transfer the boot data received, it must perform the following steps: †

- Set up the descriptors as described in *Internal DMA Controller Transmission Sequences* and *Internal DMA Controller Reception Sequences*". †
- Set the use_internal_dmac bit of the ctrl register to 1.†

4. Set the card device frequency to 400 kHz using the clkdiv registers. For more information, refer to Clock Setup.†
5. Set the data_timeout field of the tmout register equal to the card device total access time, N_{AC}. †
6. Set the blksiz register to 0x200 (512 bytes). †
7. Set the bytcnt register to a multiple of 128 KB, as indicated by the BOOT_SIZE_MULT value in the card device.†
8. Set the rx_wmark field in the fifoth register. Typically, the threshold value can be set to 512, which is half the FIFO buffer depth.†
9. Set the following fields in the cmd register:†
    - Initiate the command by setting start_cmd = 1†
    - Enable boot (enable_boot) = 1†
    - Expect boot acknowledge (expect_boot_ack): †
        - If a start-acknowledge pattern is expected from the card device, set expect_boot_ack to 1.†
        - If a start-acknowledge pattern is not expected from the card device, set expect_boot_ack to 0.†
    - Card number (card_number) = 0†
    - data_expected = 1†
    - Reset the remainder of cmd register bits to 0†
10. If no start-acknowledge pattern is expected from the card device (expect_boot_ack set to 0) proceed to **step 12**.†
11. This step handles the case where a start-acknowledge pattern is expected (expect_boot_ack was set to 1 in **step 9**).†

    a. If the Boot ACK Received interrupt is not received from the controller within 50 ms of initiating the command (**step 9**), the software driver must set the following cmd register fields: †

    - start_cmd = 1†
    - Disable boot (disable_boot)= 1†
    - card_number = 0 †
    - All other fields = 0†

    The controller generates a Command Done interrupt after deasserting the CMD pin of the card interface.†

    If internal DMA controller mode is used for the boot process, the controller performs the following steps after the Boot ACK Received timeout:†

    - The DMA descriptor is closed.†
    - The ces bit in the idsts register is set, indicating the Boot ACK Received timeout.†
    - The ri bit of the idsts register is not set.†

    b. If the Boot ACK Received interrupt is received, the software driver must clear this interrupt by writing 1 to the ces bit in the idsts register.†

    Within 0.95 seconds of the Boot ACK Received interrupt, the Boot Data Start interrupt must be received from the controller. If this does not occur, the software driver must write the following cmd register fields:†

    - start_cmd = 1†
    - disable_boot = 1†
    - card_number = 0†
    - All other fields = 0†

The controller generates a Command Done interrupt after deasserting the `CMD` pin of the card interface.†

If internal DMA controller mode is used for the boot process, the controller performs the following steps after the Boot ACK Received timeout:†

- The DMA descriptor is closed†
- The `ces` bit in the `idsts` register is set, indicating Boot Data Start timeout†
- The `ri` bit of the `idsts` register is not set†

c. If the Boot Data Start interrupt is received, it indicates that the boot data is being received from the card device. When the DMA engine is not in internal DMA controller mode, the software driver can then initiate a data read from the controller based on the `rxdr` interrupt bit in the `rintsts` register.†

In internal DMA controller mode, the DMA engine starts transferring the data from the FIFO buffer to the system memory as soon as the level set in the `rx_wmark` field of the `fifoth` register is reached.†

At the end of a successful boot data transfer from the card, the following interrupts are generated:†

- The `cmd` bit and `dto` bit in the `rintsts` register†
- The `ri` bit in the `idsts` register, in internal DMA controller mode only†

d. If an error occurs in the boot ACK pattern (0b010) or an EBE occurs: †

- The controller automatically aborts the boot process by pulling the CMD line high†
- The controller generates a Command Done interrupt†
- The controller does not generate a Boot ACK Received interrupt†
- The application aborts the boot transfer†

e. In internal DMA controller mode:†

- If the software driver creates more descriptors than required by the received boot data, the extra descriptors are not closed by the controller. Software cannot reuse the descriptors until they are closed.†
- If the software driver creates fewer descriptors than required by the received boot data, the controller generates a Descriptor Unavailable interrupt and does not transfer any further data to system memory.†

f. If N_{AC} is violated between data block transfers, the DRTO interrupt is asserted. In addition, if there is an error associated with the start or end bit, the SBE or EBE interrupt is also generated.†

The boot operation for eMMC card devices is complete. Do not execute the remaining (**step 12**).†

12. This step handles the case where no start-acknowledge pattern is expected (`expect_boot_ack` was set to 0 in **step 9**).†

a. If the Boot Data Start interrupt is not received from the controller within 1 second of initiating the command (**step 9**), the software driver must write the `cmd` register with the following fields:†

- `start_cmd` = 1†
- `disable_boot` = 1†
- `card_number` = 0†
- All other fields = 0†

The controller generates a Command Done interrupt after deasserting the CMD line of the card. In internal DMA controller mode, the descriptor is closed and the `ces` bit in the `idsts` register is set to 1, indicating a Boot Data Start timeout.†

b. If a Boot Data Start interrupt is received, it indicates that the boot data is being received from the card device. When the DMA engine is not in internal DMA controller mode, the software driver can then initiate a data read from the controller based on the `rxdr` interrupt bit in the `rintsts` register.†

In internal DMA controller mode, the DMA engine starts transferring the data from the FIFO buffer to the system memory as soon as the level specified in the `rx_wmark` field of the `fifoth` register is reached.†

At the end of a successful boot data transfer from the card, the following interrupts are generated:†

- The `cmd` bit and `dto` bit in the `rintsts` register†
- The `ri` bit in the `idsts` register, in internal DMA controller mode only†

c. In internal DMA controller mode:†

- If the software driver creates more descriptors than required by the received boot data, the extra descriptors are not closed by the controller.†
- If the software driver creates fewer descriptors than required by the received boot data, the controller generates a Descriptor Unavailable interrupt and does not transfer any further data to system memory.†

The boot operation for eMMC card devices is complete.†

Related Information

- **Clock Setup** on page 14-41
  Refer to this section for information on how to set the card device frequency.
- **Internal DMA Controller Transmission Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Transmission Sequences.
- **Internal DMA Controller Reception Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Reception Sequences.

Boot Operation for Removable MMC4.3, MMC4.4 and MMC4.41 Cards

Removable MMC4.3, MMC4.4, and MMC4.41 Differences

Removable MMC4.3, MMC4.4, and MMC4.41 cards differ with respect to eMMC in that the controller is not aware whether these cards support the boot mode of operation when plugged in. Thus, the controller must: †

1. Discover these cards as it would discover MMC4.0/4.1/4.2 cards for the first time†
2. Know the card characteristics †
3. Decide whether to perform a boot operation or not†

Booting Removable MMC4.3, MMC4.4 and MMC4.41 Cards

For removable MMC4.3, MMC4.4 and MMC4.41 cards, the software driver must perform the following steps:†

1. Discover the card as described in *Enumerated Card Stack*.†
2. Read the EXT_CSD register of the card and examine the following fields: †
   - BOOT_PARTITION_ENABLE †
   - BOOT_SIZE_MULT†
   - BOOT_INFO †
3. If necessary, the software can manipulate the boot information in the card. †

   Note: For more information, refer to "Access to Boot Partition" in the following specifications available on the JEDEC website:

- JEDEC Standard No. 84-A441
- JEDEC Standard No. 84-A44
- JEDEC Standard No. JESD84-A43

4. If the host processor needs to perform a boot operation at the next power-up cycle, it can manipulate the EXT_CSD register contents by using a SWITCH_FUNC command. †
5. After this step, the software driver must power down the card by writing to the `pwren` register. †
6. From here on, use the same steps as in *Alternative Boot Operation for eMMC Card Devices*.†

Related Information

- **Enumerated Card Stack** on page 14-38
  Refer to this section for more information on discovering removable MMC cards.
- **http://www.jedec.org**
- **Alternative Boot Operation for eMMC Card Devices** on page 14-74
  Refer to this section for information about alternative boot operation steps.

Alternative Boot Operation

The alternative boot operation differs from the previous boot operation in that software uses the SD/SDIO GO_IDLE_STATE command to boot the card, rather than holding down the CMD line of the card. The alternative boot operation can be performed only if bit 0 in the BOOT_INFO register is set to 1. BOOT_INFO is located at offset 228 in the EXT_CSD registers. †

For detailed information about alternative boot operation, refer to the following specifications available on the JEDEC website:

- JEDEC Standard No. 84-A441
- JEDEC Standard No. 84-A44
- JEDEC Standard No. JESD84-A43

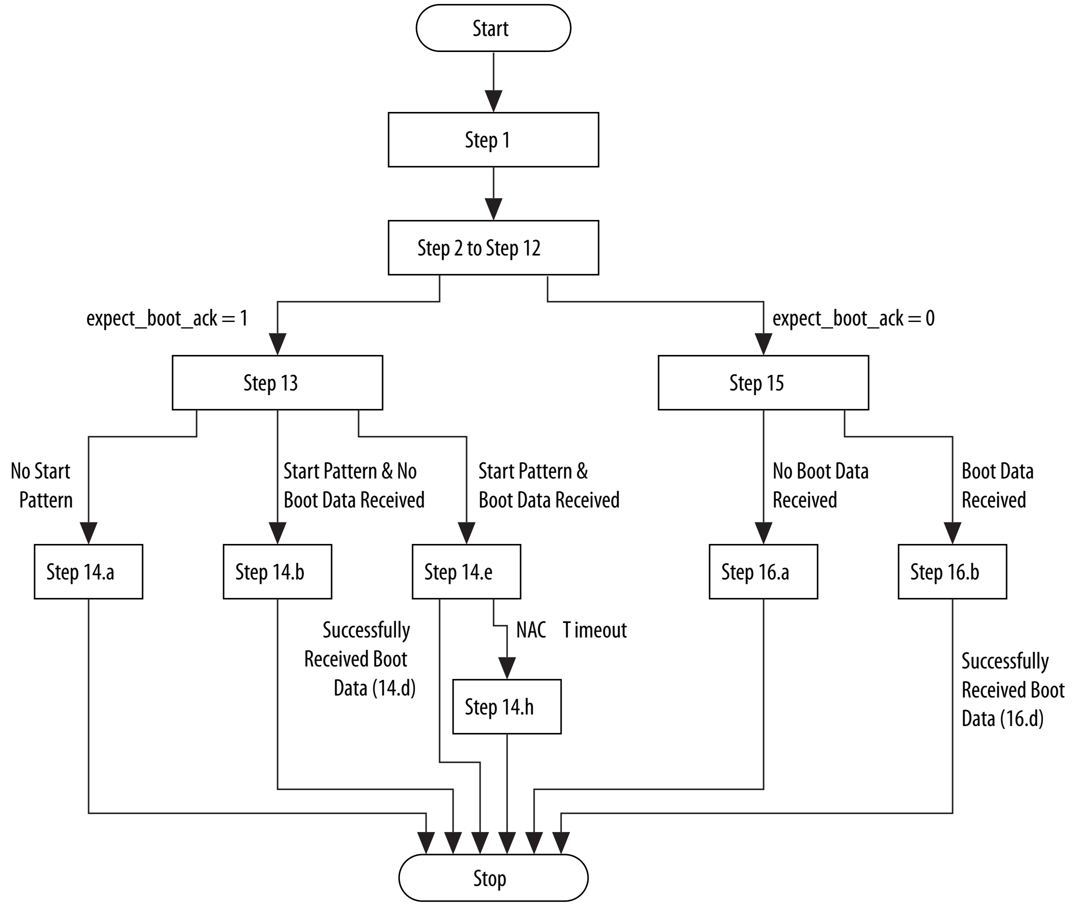
Related Information

www.jedec.org
For more information about alternative boot operation, refer to the following JEDEC Standards available on the JEDEC website: No. 84-A441, No. 84-A44, and No. JESD84-A43.

Alternative Boot Operation for eMMC Card Devices

The following figure illustrates the sequence of steps required to perform the alternative boot operation for eMMC card devices. The detailed steps are described following the flow chart.

Figure 14-16: Flow for eMMC Alternative Boot Operation[†]



1. The software driver checks:[†]
   - If the eMMC card device supports alternative boot operation (the BOOT_INFO bit is set to 1 in the eMMC card).[†]
   - The BOOT_SIZE_MULT and BOOT_BUS_WIDTH values in the card device to use during the boot process.[†]
2. The software sets the following bits: [†]
   - Sets masks for interrupts by resetting the appropriate bits to 0 in the `intmask` register.[†]
   - Sets the `int_enable` bit of the `ctrl` register to 1. Other bits in the `ctrl` register must be set to 0. [†]

   Note: Altera recommends writing 0xFFFFFFFF to the `rintsts` register and `idsts` register to clear any pending interrupts before setting the `int_enable` bit. For internal DMA controller mode, the software driver needs to unmask all the relevant fields in the `idinten` register.[†]
3. If the software driver needs to use the internal DMA controller to transfer the boot data received, it must perform the following actions: [†]
   - Set up the descriptors as described in *Internal DMA Controller Transmission Sequences* and *Internal DMA Controller Reception Sequences.* [†]
   - Set the use internal DMAC bit (`use_internal_dmac`) of the `ctrl` register to 1. [†]
4. Set the card device frequency to 400 kHz using the `clkdiv` registers. For more information, refer to *Clock Setup*. Ensure that the card clock is running.[†]

5. Wait for a time that ensures that at least 74 card clock cycles have occurred on the card interface.†
6. Set the `data_timeout` field of the `tmout` register equal to the card device total access time, N_{AC}. †
7. Set the `blksiz` register to 0x200 (512 bytes).†
8. Set the `bytcnt` register to multiples of 128K bytes, as indicated by the BOOT_SIZE_MULT value in the card device.†
9. Set the `rx_wmark` field in the `fifoth` register. Typically, the threshold value can be set to 512, which is half the FIFO buffer depth.†
10. Set the `cmdarg` register to 0xFFFFFFFA. †
11. Initiate the command, by setting the `cmd` register with the following fields: †

    - `start_cmd` = 1†
    - `enable_boot` = 1†
    - `expect_boot_ack`:†
        - If a start-acknowledge pattern is expected from the card device, set `expect_boot_ack` to 1.†
        - If a start-acknowledge pattern is not expected from the card device, set `expect_boot_ack` to 0. †
    - `card_number` = 0 †
    - `data_expected` = 1†
    - `cmd_index` = 0†
    - Set the remainder of `cmd` register bits to 0.†
12. If no start-acknowledge pattern is expected from the card device (`expect_boot_ack` set to 0) jump to **step 15**. †
13. Wait for the Command Done interrupt.†
14. This step handles the case where a start-acknowledge pattern is expected (`expect_boot_ack` was set to 1 in **step 11**).†

    a. If the Boot ACK Received interrupt is not received from the controller within 50 ms of initiating the command (**step 11**), the start pattern was not received. The software driver must discontinue the boot process and start with normal discovery. †

    If internal DMA controller mode is used for the boot process, the controller performs the following steps after the Boot ACK Received timeout:†

    - The DMA descriptor is closed.†
    - The `ces` bit in the `idsts` register is set to 1, indicating the Boot ACK Received timeout.†
    - The `ri` bit of the `idsts` register is not set.†

    b. If the Boot ACK Received interrupt is received, the software driver must clear this interrupt by writing 1 to it. †

    Within 0.95 seconds of the Boot ACK Received interrupt, the Boot Data Start interrupt must be received from the controller. If this does not occur, the software driver must discontinue the boot process and start with normal discovery.†

    If internal DMA controller mode is used for the boot process, the controller performs the following steps after the Boot ACK Received timeout: †

    - The DMA descriptor is closed.†
    - The `ces` bit in the `idsts` register is set to 1, indicating Boot Data Start timeout.†
    - The `ri` bit of the `idsts` register is not set.†

    c. If the Boot Data Start interrupt is received, it indicates that the boot data is being received from the card device. When the DMA engine is not in internal DMA controller mode, the software driver can then initiate a data read from the controller based on the `rxdr` interrupt bit in the `rintsts` register.†

In internal DMA controller mode, the DMA engine starts transferring the data from the FIFO buffer to the system memory as soon as the level specified in the `rx_wmark` field of the `fifoth` register is reached. †

d. The software driver must terminate the boot process by instructing the controller to send the SD/SDIO GO_IDLE_STATE command:†

  - Reset the `cmdarg` register to 0.†
  - Set the `start_cmd` bit of the `cmd` register to 1, and all other bits to 0.†

e. At the end of a successful boot data transfer from the card, the following interrupts are generated: †

  - The `cmd` bit and `dto` bit in the `rintsts` register†
  - The `ri` bit in the `idsts` register, in internal DMA controller mode only†

f. If an error occurs in the boot ACK pattern (0b010) or an EBE occurs: †

  - The controller does not generate a Boot ACK Received interrupt. †
  - The controller detects Boot Data Start and generates a Boot Data Start interrupt. †
  - The controller continues to receive boot data. †
  - The application must abort the boot process after receiving a Boot Data Start interrupt.†

g. In internal DMA controller mode: †

  - If the software driver creates more descriptors than required by the received boot data, the extra descriptors are not closed by the controller. †
  - If the software driver creates fewer descriptors than required by the received boot data, the controller generates a Descriptor Unavailable interrupt and does not transfer any further data to system memory.†

h. If N_{AC} is violated between data block transfers, a DRTO interrupt is asserted. Apart from this, if there is an error associated with the start or end bit, the SBE or EBE interrupt is also generated.†

The alternative boot operation for eMMC card devices is complete. Do not execute the remaining steps (15 and 16). †

15. Wait for the Command Done interrupt.†
16. This step handles the case where a start-acknowledge pattern is not expected (`expect_boot_ack` was set to 0 in **step 11**). †

a. If the Boot Data Start interrupt is not received from the controller within 1 second of initiating the command (**step 11**), the software driver must discontinue the boot process and start with normal discovery. † In internal DMA controller mode:†

  - The DMA descriptor is closed.†
  - The `ces` bit in the `idsts` register is set to 1, indicating Boot Data Start timeout.†
  - The `ri` bit of the `idsts` register is not set.†

b. If a Boot Data Start interrupt is received, the boot data is being received from the card device. When the DMA engine is not in internal DMA controller mode, the software driver can then initiate a data read from the controller based on the `rxdr` interrupt bit in the `rintsts` register.†

In internal DMA controller mode, the DMA engine starts transferring the data from the FIFO buffer to the system memory as soon as the level specified in the `rx_wmark` field of the `fifoth` register is reached.†

c. The software driver must terminate the boot process by instructing the controller to send the SD/SDIO GO_IDLE_STATE (CMD0) command: †

  - Reset the `cmdarg` register to 0.†
  - Set the `start_cmd` bit in the `cmd` register to 1, and all other bits to 0.†

d. At the end of a successful boot data transfer from the card, the following interrupts are generated: †

- The `cmd` bit and `dto` bit in the `rintsts` register†
- The `ri` bit in the `idsts` register, in internal DMA controller mode only †

e. In internal DMA controller mode: †

- If the software driver creates more descriptors than required by the received boot data, the extra descriptors are not closed by the controller. †
- If the software driver creates fewer descriptors than required by the received boot data, the controller generates a Descriptor Unavailable interrupt and does not transfer any further data to system memory.†

The alternative boot operation for eMMC card devices is complete.†

Related Information

- **Clock Setup** on page 14-41
  Refer to this section for information on how to set the card device frequency.
- **Internal DMA Controller Transmission Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Transmission Sequences.
- **Internal DMA Controller Reception Sequences** on page 14-54
  Refer to this section for information about the Internal DMA Controller Reception Sequences.

Alternative Boot Operation for MMC4.3 Cards

Removable MMC4.3 Boot Mode Support

Removable MMC4.3 cards differ with respect to eMMC in that the controller is not aware whether these cards support the boot mode of operation. Thus, the controller must: †

1. Discover these cards as it would discover MMC4.0/4.1/4.2 cards for the first time †
2. Know the card characteristics †
3. Decide whether to perform a boot operation or not†

Discovering Removable MMC4.3 Boot Mode Support

For removable MMC4.3 cards, the software driver must perform the following steps: †

1. Discover the card as described in *Enumerated Card Stack*.†
2. Read the MMC card device's EXT_CSD registers and examine the following fields: †
   - BOOT_PARTITION_ENABLE †
   - BOOT_SIZE_MULT †
   - BOOT_INFO †

   Note: For more information, refer to "Access to Boot Partition" in JEDEC Standard No. JESD84-A43, available on the JEDEC website.†
3. If the host processor needs to perform a boot operation at the next power-up cycle, it can manipulate the contents of the EXT_CSD registers in the MMC card device, by using a SWITCH_FUNC command. †
4. After this step, the software driver must power down the card by writing to the `pwren` register. †
5. From here on, use the same steps as in *Alternative Boot Operation for eMMC Card Devices*. †

   Note: Ignore the EBE if it is generated during an abort scenario.

   If a boot acknowledge error occurs, the boot acknowledge received interrupt times out. †

In internal DMA controller mode, the application needs to depend on the descriptor close interrupt instead of the data done interrupt. †

Related Information

- **Enumerated Card Stack** on page 14-38
  Refer to this section for more information on discovering removable MMC cards.
- **www.jedec.org**
  For more information, refer to "Access to Boot Partition" in JEDEC Standard No. JESD84-A43, available on the JEDEC website.
- **Alternative Boot Operation for eMMC Card Devices** on page 14-74
  Refer to this section for information about alternative boot operation steps.

Voltage Switches

This section describes the general steps to switch voltage level.

The SD/MMC cards support various operating voltages, for example 1.8V and 3.3V. If you have a card which is at 1.8V and you eject it and replace it with another card, which is 3.3V, then voltage switching is required.

In order to have the right voltage level to power the card, separate devices on the board are required: voltage translation transceiver and power regulator/supply. When the software is aware that voltage switching is needed, it should control the power regulator to supply another voltage level to the card (i.e. switching between 1.8V and 3.3V).

However, for the HPS (or Altera device), the I/O pins are connected to the SD/MMC card running at 3.3V. If your card runs at 1.8V, the voltage-translation transceiver is needed for voltage translation between the HPS and the SD/MMC card.

The general steps to switch voltage level requires you to use a SD/MMC voltage-translation transceiver in between the HPS and the SD/MMC card.

1. Power the HPS I/O pins for the SD/MMC controller to 3.3V.
   a. Connect the same power supply to one of the transceiver voltage input pins.
2. Power the other transceiver voltage input pin with another power supply.
   a. Connect this power supply with the SD/MMC card.
3. The SD/MMC will send predefined commands to check if the card supports dual voltage. The response from the card will indicate if dual voltage is supported
4. Software stops all SD/MMC activity. If the card does not support dual voltage, do not perform the remaining steps.
5. The GPIO on HPS sends a signal to the external power supply on the board notifying it to switch its voltage value.
6. After voltage switching is completed, resume the SD/MMC activity.

SD/MMC Controller Address Map and Register Definitions

The address map and register definitions for the SD/MMC Controller consists of the following region:

- SD/MMC Module

SDMMC Module Address Map

Registers in the SD/MMC module

Base Address: `0xFF704000`

SDMMC Module

Register	Offset	Width	Acces s	Reset Value	Description
ctrl on page 14-81	0x0	32	RW	0x0	Control Register
pwren on page 14-85	0x4	32	RW	0x0	Power Enable Register
clkdiv on page 14-86	0x8	32	RW	0x0	Clock Divider Register
clksrc on page 14-86	0xC	32	RW	0x0	SD Clock Source Register
clkena on page 14-87	0x10	32	RW	0x0	Clock Enable Register
tmout on page 14-88	0x14	32	RW	0xFFFFFF40	Timeout Register
ctype on page 14-89	0x18	32	RW	0x0	Card Type Register
blksiz on page 14-89	0x1C	32	RW	0x200	Block Size Register
bytcnt on page 14-90	0x20	32	RW	0x200	Byte Count Register
intmask on page 14-91	0x24	32	RW	0x0	Interrupt Mask Register
cmdarg on page 14-94	0x28	32	RW	0x0	Command Argument Register
cmd on page 14-95	0x2C	32	RW	0x20000000	Command Register
resp0 on page 14-101	0x30	32	RO	0x0	Response Register 0
resp1 on page 14-102	0x34	32	RO	0x0	Response Register 1
resp2 on page 14-102	0x38	32	RO	0x0	Response Register 2
resp3 on page 14-103	0x3C	32	RO	0x0	Response Register 3
mintsts on page 14-103	0x40	32	RO	0x0	Masked Interrupt Status Register
rintsts on page 14-107	0x44	32	RW	0x0	Raw Interrupt Status Register
status on page 14-111	0x48	32	RO	0x106	Status Register
fifoth on page 14-114	0x4C	32	RW	0x3FF0000	FIFO Threshold Watermark Register
cdetect on page 14-116	0x50	32	RO	0x1	Card Detect Register
wrtprt on page 14-117	0x54	32	RO	0x1	Write Protect Register
tcbcnt on page 14-117	0x5C	32	RO	0x0	Transferred CIU Card Byte Count Register
tbbcnt on page 14-118	0x60	32	RO	0x0	Transferred Host to BIU-FIFO Byte Count Register
debnce on page 14-118	0x64	32	RW	0xFFFFFF	Debounce Count Register
usrid on page 14-119	0x68	32	RW	0x7967797	User ID Register

Register	Offset	Width	Acces s	Reset Value	Description
verid on page 14-119	0x6C	32	RO	0x5342240A	Version ID Register
hcon on page 14-120	0x70	32	RO	0xC43081	Hardware Configuration Register
uhs_reg on page 14-122	0x74	32	RW	0x0	UHS-1 Register
rst_n on page 14-123	0x78	32	RW	0x1	Hardware Reset Register
bmod on page 14-124	0x80	32	RW	0x0	Bus Mode Register
pldmnd on page 14-126	0x84	32	WO	0x0	Poll Demand Register
dbaddr on page 14-126	0x88	32	RW	0x0	Descriptor List Base Address Register
idsts on page 14-127	0x8C	32	RW	0x0	Internal DMAC Status Register
idinten on page 14-130	0x90	32	RW	0x0	Internal DMAC Interrupt Enable Register
dscaddr on page 14-132	0x94	32	RO	0x0	Current Host Descriptor Address Register
bufaddr on page 14-132	0x98	32	RO	0x0	Current Buffer Descriptor Address Register
cardthrctl on page 14-133	0x100	32	RW	0x0	Card Threshold Control Register
back_end_power_r on page 14-134	0x104	32	RW	0x0	Back End Power Register
data on page 14-134	0x200	32	RW	0x0	Data FIFO Access

ctrl

Sets various operating condiitions.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						use_internal_dmac RW 0x0	Reserved								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ceata_device_interrupt_status RW 0x0	send_auto_stop_ccsd RW 0x0	send_ccsd RW 0x0	abort_read_data RW 0x0	send_irq_response RW 0x0	read_wait RW 0x0	Reserved	int_enable RW 0x0	Reserved	dma_reset RW 0x0	fifo_reset RW 0x0	controller_reset RW 0x0

ctrl Fields

Bit	Name	Description	Access	Reset
25	use_internal_dmac	Enable and Disable Internal DMA transfers. **Value** **Description** 0x0 The host performs data transfers thru slave interface 0x1 Internal DMAC used for data transfer	RW	0x0
11	ceata_device_ interrupt_status	Software should appropriately write to this bit after power-on reset or any other reset to CE-ATA device. After reset, usually CE-ATA device interrupt is disabled (nIEN = 1). If the host enables CE-ATA device interrupt, then software should set this bit. **Value** **Description** 0x0 Interrupts not enabled in CE-ATA device 0x1 Interrupts are enabled in CE-ATA device	RW	0x0

Bit	Name	Description	Access	Reset
10	`send_auto_stop_ccsd`	Always set send_auto_stop_ccsd and send_ccsd bits together; send_auto_stop_ccsd should not be set independent of send_ccsd. When set, SD/MMC automatically sends internally generated STOP command (CMD12) to CE-ATA device. After sending internally-generated STOP command, Auto Command Done (ACD) bit in RINTSTS is set and generates interrupt to host if Auto CommandDone interrupt is not masked. After sending the CCSD, SD/MMC automatically clears send_auto_stop_ccsd bit. **Value** **Description** 0x0 Clear bit if SD/MMC does not reset the bit 0x1 Send internally generated STOP.	`RW`	`0x0`
9	`send_ccsd`	When set, SD/MMC sends CCSD to CE-ATA device. Software sets this bit only if current command is expecting CCS (that is, RW_BLK) and interrupts are enabled in CE-ATA device. Once the CCSD pattern is sent to device, SD/MMC automatically clears send_ccsd bit. It also sets Command Done (CD) bit in RINTSTS register and generates interrupt to host if Command Done interrupt is not masked. NOTE: Once send_ccsd bit is set, it takes two card clock cycles to drive the CCSD on the CMD line. Due to this, during the boundary conditions it may happen that CCSD is sent to the CE-ATA device, even if the device signalled CCS. **Value** **Description** 0x0 Clear bit if SD/MMC does not reset the bit 0x1 Send Command Completion Signal Disable (CCSD) to CE-ATA device	`RW`	`0x0`
8	`abort_read_data`	After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data statemachine resets to idle. Used in SDIO card suspend sequence. **Value** **Description** 0x0 No change 0x1 Abort Read	`RW`	`0x0`

Bit	Name	Description	Access	Reset
7	`send_irq_response`	Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and SD/MMC waits for interrupt response from MMC card(s). In meantime, if host wants SD/MMC to exit waiting for interrupt state, it can set this bit, at which time SD/MMC command state-machine sends CMD40 response on bus and returns to idle state. **Value** **Description** 0x0 No change 0x1 Send auto IRQ response	RW	0x0
6	`read_wait`	For sending read-wait to SDIO cards. **Value** **Description** 0x0 Read Wait 0x1 Assert Read Wait	RW	0x0
4	`int_enable`	This bit enables and disable interrupts if one or more unmasked interrupts are set. **Value** **Description** 0x0 Disable Interrupts 0x1 Enable interrupts	RW	0x0
2	`dma_reset`	This bit resets the DMA interface control logic **Value** **Description** 0x0 No change 0x1 Reset internal DMA interface control logic	RW	0x0
1	`fifo_reset`	This bit resets the FIFO. This bit is auto-cleared after completion of reset operation. **Value** **Description** 0x0 No change 0x1 Reset to data FIFO To reset FIFO pointers	RW	0x0

Bit	Name	Description	Access	Reset
0	`controller_reset`	This bit resets the controller. This bit is auto-cleared after two l4_mp_clk and two sdmmc_clk clock cycles. This resets: - BIU/CIU interface - CIU and state machines - abort_read_data, send_irq_response, and read_wait bits of control register -start_cmd bit of command register Does not affect any registers, DMA interface, FIFO or host interrupts. **Value** **Description** 0x0 No change -default 0x1 Reset SD/MMC controller	RW	0x0

pwren

Power on/off switch for card; once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704004

Offset: `0x4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															power_enable RW 0x0

pwren Fields

Bit	Name	Description	Access	Reset
0	`power_enable`	Power on/off switch for one card; for example, bit[0] controls the card. Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card. **Value** **Description** 0x0 Power Off 0x1 Power On	RW	0x0

clkdiv

Divides Clock sdmmc_clk.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								clk_divider0 RW 0x0							

clkdiv Fields

Bit	Name	Description	Access	Reset
7:0	clk_divider0	Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2*0 = 0 (no division, bypass), value of 1 means divide by 2*1 = 2, value of ff means divide by 2*255 = 510, and so on.	RW	0x0

clksrc

Selects among available clock dividers. The sdmmc_cclk_out is always from clock divider 0.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70400C

Offset: 0xC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														clk_source RW 0x0	

clksrc Fields

Bit	Name	Description	Access	Reset
1:0	`clk_source`	Selects among available clock dividers. The SD/MMC module is configured with just one clock divider so this register should always be set to choose clkdiv0. **Value** **Description** 0x0 Clock divider 0	`RW`	`0x0`

clkena

Controls external SD/MMC Clock Enable.

Module Instance	Base Address	Register Address
`sdmmc`	`0xFF704000`	`0xFF704010`

Offset: `0x10`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															cclk_low_power RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															cclk_enable RW 0x0

clkena Fields

Bit	Name	Description	Access	Reset
16	`cclk_low_power`	In low-power mode, stop sdmmc_cclk_out when card in IDLE (should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped). **Value** — **Description** 0x0 — Non-low-power mode 0x1 — Low-power mode	RW	0x0
0	`cclk_enable`	Enables sdmmc_cclk_out. **Value** — **Description** 0x0 — SD/MMC Disable 0x1 — SD/MMC Enable	RW	0x0

tmout

Sets timeout values

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704014

Offset: `0x14`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data_timeout RW 0xFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data_timeout RW 0xFFFFFF								response_timeout RW 0x40							

tmout Fields

Bit	Name	Description	Access	Reset
31:8	`data_timeout`	Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. Value is in number of card output clocks sdmmc_cclk_out of selected card.	RW	0xFFFFFF
7:0	`response_timeout`	Response timeout value. Value is in number of card output clocks sdmmc_cclk_out.	RW	0x40

ctype

Describes card formats.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															card_width1 RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															card_width2 RW 0x0

ctype Fields

Bit	Name	Description	Access	Reset
16	card_width1	Indicates if card is 8 bit or othersize. If not 8-bit, card_width2 specifies the width. **Value** / **Description** 0x0 / Non 8-bit mode 0x1 / 8-bit mode	RW	0x0
0	card_width2	Ignored if card_width1 is MODE8BIT. **Value** / **Description** 0x0 / 1-bit mode 0x1 / 4-bit mode	RW	0x0

blksiz

The Block Size.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70401C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
block_size RW 0x200															

blksiz Fields

Bit	Name	Description	Access	Reset
15:0	block_size	The size of a block in bytes.	RW	0x200

bytcnt

The number of bytes to be transferred.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
byte_count RW 0x200															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
byte_count RW 0x200															

bytcnt Fields

Bit	Name	Description	Access	Reset
31:0	byte_count	This value should be an integer multiple of the Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer. Note: In SDIO mode, if a single transfer is greater than 4 bytes and non-DWORD-aligned, the transfer should be broken where only the last transfer is non-DWORD-aligned and less than 4 bytes. For example, if a transfer of 129 bytes must occur, then the driver should start at least two transfers; one with 128 bytes and the other with 1 byte.	RW	0x200

intmask

Allows Masking of Various Interrupts

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704024

Offset: 0x24

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															sdio_int_mask RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ebe RW 0x0	acd RW 0x0	sbe RW 0x0	hle RW 0x0	frun RW 0x0	hto RW 0x0	drt RW 0x0	rto RW 0x0	dcrc RW 0x0	rcrc RW 0x0	rxdr RW 0x0	txdr RW 0x0	dto RW 0x0	cmd RW 0x0	re RW 0x0	cd RW 0x0

intmask Fields

Bit	Name	Description	Access	Reset
16	sdio_int_mask	In current application, MMC-Ver3.3 only Bit 16 of this field is used. Bits 17 to 31 are unused and return 0 **Value** — **Description** 0x0 — SDIO Mask Interrupt Disabled 0x1 — SDIO Interrupt Enabled	RW	0x0
15	ebe	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** — **Description** 0x0 — End-bit error Mask 0x1 — End-bit error No Mask	RW	0x0

Bit	Name	Description	Access	Reset
14	`acd`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Auto command done Mask 0x1 Auto command done No Mask	`RW`	`0x0`
13	`sbe`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Start-bit error Mask 0x1 Start-bit error No Mask	`RW`	`0x0`
12	`hle`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Hardware locked write error Mask 0x1 Hardware locked write error No Mask	`RW`	`0x0`
11	`frun`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 FIFO underrun/overrun error Mask 0x1 FIFO underrun/overrun error No Mask	`RW`	`0x0`
10	`hto`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Data starvation by host timeout Mask 0x1 Data starvation by host timeout No Mask	`RW`	`0x0`
9	`drt`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Data read timeout Mask 0x1 Data read timeout No Mask	`RW`	`0x0`

Bit	Name	Description	Access	Reset
8	`rto`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Response timeout Mask 0x1 Response timeout No Mask	RW	0x0
7	`dcrc`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Data CRC error Mask 0x1 Data CRC error No Mask	RW	0x0
6	`rcrc`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Response CRC error Mask 0x1 Response CRC error No Mask	RW	0x0
5	`rxdr`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Receive FIFO data request Mask 0x1 Receive FIFO data request No Mask	RW	0x0
4	`txdr`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Transmit FIFO data request Mask 0x1 Transmit FIFO data request No Mask	RW	0x0
3	`dto`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Data transfer over Mask 0x1 Data transfer over No Mask	RW	0x0

Bit	Name	Description	Access	Reset
2	`cmd`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Command Done Mask 0x1 Command Done No Mask	`RW`	`0x0`
1	`re`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Response error Mask 0x1 Response error No Mask	`RW`	`0x0`
0	`cd`	Bits used to mask unwanted interrupts. Value of 0 masks interrupts, value of 1 enables interrupt. **Value** **Description** 0x0 Card Detected Mask 0x1 Card Detect No Mask	`RW`	`0x0`

cmdarg

See Field Description.

Module Instance	Base Address	Register Address
`sdmmc`	`0xFF704000`	`0xFF704028`

Offset: `0x28`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">cmd_arg
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">cmd_arg
RW 0x0</td></tr>
</table>

cmdarg Fields

Bit	Name	Description	Access	Reset
31:0	`cmd_arg`	Values indicates command argument to be passed to card.	`RW`	`0x0`

cmd

This register issues various commands.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70402C

Offset: 0x2C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
start_cmd RW 0x0	Reserved	use_hold_reg RW 0x1	volt_switch RW 0x0	boot_mode RW 0x0	disable_boot RW 0x0	expect_boot_ack RW 0x0	enable_boot RW 0x0	ccs_expected RW 0x0	read_ceata_device RW 0x0	update_clock_registers_only RW 0x0	card_number RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
send_initialization RW 0x0	stop_abort_cmd RW 0x0	wait_prvdata_complete RW 0x0	send_auto_stop RW 0x0	transfer_mode RW 0x0	read_write RW 0x0	data_expected RW 0x0	check_response_crc RW 0x0	response_length RW 0x0	response_expect RW 0x0	cmd_index RW 0x0					

cmd Fields

Bit	Name	Description	Access	Reset
31	start_cmd	Once command is taken by CIU, bit is cleared. If Start Cmd issued host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register. Once command is sent and response is received from SD_MMC_CEATA cards, Command Done bit is set in raw interrupt register. **Value** — **Description** 0x0 — No Start Cmd 0x1 — Start Cmd Issued	RW	0x0
29	use_hold_reg	Set to one for SDR12 and SDR25 (with non-zero phase-shifted cclk_in_drv); zero phase shift is not allowed in these modes. -Set to 1'b0 for SDR50, SDR104, and DDR50 (with zero phase-shifted cclk_in_drv). -Set to 1'b1 for SDR50, SDR104, and DDR50 (with non-zero phase-shifted cclk_in_drv). **Value** — **Description** 0x0 — CMD and DATA sent to card bypassing HOLD Register 0x1 — CMD and DATA sent to card through the HOLD Register	RW	0x1
28	volt_switch	Voltage switch bit. When set must be set for CMD11 only. **Value** — **Description** 0x0 — No voltage switching - default 0x1 — Voltage switching enabled	RW	0x0
27	boot_mode	Type of Boot Mode. **Value** — **Description** 0x0 — Mandatory Boot Operation 0x1 — Alternate Boot Operation	RW	0x0

Bit	Name	Description	Access	Reset
26	`disable_boot`	When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together. **Value** **Description** 0x0 Boot not Terminated 0x1 Terminate Boot	RW	0x0
25	`expect_boot_ack`	When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card. **Value** **Description** 0x0 No Boot ACK 0x1 Expect Boot ACK	RW	0x0
24	`enable_boot`	This bit should be set only for mandatory boot mode. When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together **Value** **Description** 0x0 Disable Boot 0x1 Enable Boot	RW	0x0
23	`ccs_expected`	If the command expects Command Completion Signal (CCS) from the CE-ATA device, the software should set this control bit. SD/MMC sets Data Transfer Over (DTO) bit in RINTSTS register and generates interrupt to host if Data Transfer Over interrupt is not masked. **Value** **Description** 0x0 Interrupts are not enabled in CE-ATA device (nIEN = 1 in ATA control register), or command does not expect CCS from device 0x1 Interrupts are enabled in CE-ATA device (nIEN = 0), and RW_BLK command expects command completion signal from CE-ATA device	RW	0x0

Bit	Name	Description	Access	Reset
22	`read_ceata_device`	Software should set this bit to indicate that CE-ATA device is being accessed for read transfer. This bit is used to disable read data timeout indication while performing CE-ATA read transfers. Maximum value of I/O transmission delay can be no less than 10 seconds. SD/MMC should not indicate read data timeout while waiting for data from CE-ATA device. **Value** **Description** 0x0 Host is not performing read access (RW_REG or RW_BLK) towards CE-ATA device 0x1 Host is performing read access (RW_REG or RW_BLK) towards CE-ATA device	RW	0x0
21	`update_clock_registers_only`	Following register values transferred into card clock domain: CLKDIV, CLRSRC, CLKENA. Changes card clocks (change frequency, truncate off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only = 0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card(s). When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC_CEATA cards. **Value** **Description** 0x0 Normal command sequence 0x1 Do not send commands, just update clock register value into card clock domain	RW	0x0
20:16	`card_number`	Card number in use must always be 0.	RW	0x0

Bit	Name	Description	Access	Reset
15	`send_initialization`	After power on, 80 clocks must be sent to the card for initialization before sending any commands to card. Bit should be set while sending first command to card so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory). **Value** **Description** 0x0 Do not send initialization sequence (80 clocks of 1) before sending this command 0x1 Send initialization sequence before sending this command	RW	0x0
14	`stop_abort_cmd`	When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/ data state-machines of CIU can return correctly to idle state. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = disable_boot. Note: If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0. **Value** **Description** 0x0 Don't stop or abort command to stop current data transfer in progress 0x1 Stop or Abort command, intended to stop current data transfer in progress	RW	0x0
13	`wait_prvdata_complete`	Determines when command is sent. The send command at once option is typically used to query status of card during data transfer or to stop current data transfer. **Value** **Description** 0x0 Send command at once 0x1 Wait for previous data transfer completion	RW	0x0

Bit	Name	Description	Access	Reset
12	`send_auto_stop`	When set, SD/MMC sends stop command to SD_MMC_CEATA cards at end of data transfer. Determine the following: *-when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands. *-open-ended transfers that software should explicitly send to stop command. Additionally, when resume is sent to resume-suspended memory access of SD-Combo card, bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card. **Value** / **Description** 0x0 No stop command sent at end of data transfer 0x1 Send stop command at end of data transfer	RW	0x0
11	`transfer_mode`	Block transfer command. Don't care if no data expected **Value** / **Description** 0x0 Block data transfer command 0x1 Stream data transfer command	RW	0x0
10	`read_write`	Read/Write from card. Don't care if no data transfer expected. **Value** / **Description** 0x0 Read from card 0x1 Write to card	RW	0x0
9	`data_expected`	Set decision on data transfer expecetd or not. **Value** / **Description** 0x0 No data transfer expected (read/write) 0x1 Data transfer expected (read/write)	RW	0x0

Bit	Name	Description	Access	Reset
8	`check_response_crc`	Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller. **Value** — **Description** 0x0 — Do not check response CRC 0x1 — Check Response CRC	RW	0x0
7	`response_length`	Provides long and short response **Value** — **Description** 0x0 — Short response expected from card 0x1 — Long response expected from card	RW	0x0
6	`response_expect`	Response expected from card. **Value** — **Description** 0x0 — No response expected from card 0x1 — Response expected from card	RW	0x0
5:0	`cmd_index`	Tracks the command index number. Values from 0-31.	RW	0x0

resp0

Preserves previous command.

Module Instance	Base Address	Register Address
`sdmmc`	`0xFF704000`	`0xFF704030`

Offset: `0x30`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">response0
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">response0
RO 0x0</td></tr>
</table>

resp0 Fields

Bit	Name	Description	Access	Reset
31:0	`response0`	Bit[31:0] of response.	RO	0x0

resp1

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704034

Offset: `0x34`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
response1 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
response1 RO 0x0															

resp1 Fields

Bit	Name	Description	Access	Reset
31:0	`response1`	Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always short for them.	RO	0x0

resp2

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704038

Offset: `0x38`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
response2 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
response2 RO 0x0															

resp2 Fields

Bit	Name	Description	Access	Reset
31:0	`response2`	Bit[95:64] of long response	RO	0x0

resp3

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70403C

Offset: `0x3C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
response3 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
response3 RO 0x0															

resp3 Fields

Bit	Name	Description	Access	Reset
31:0	`response3`	Bit[127:96] of long response	RO	0x0

mintsts

Describes state of Masked Interrupt Register.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704040

Offset: `0x40`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															sdio_interrupt RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ebe RO 0x0	acd RO 0x0	strerr RO 0x0	hlwerr RO 0x0	fifoovunerr RO 0x0	dshto RO 0x0	datardto RO 0x0	respto RO 0x0	datacrcerr RO 0x0	respcrcerr RO 0x0	rxfifodr RO 0x0	dttxfifodr RO 0x0	dt RO 0x0	cmd_done RO 0x0	resp RO 0x0	cd RO 0x0

mintsts Fields

Bit	Name	Description	Access	Reset
16	sdio_interrupt	Interrupt from SDIO card: one bit for each card. Bit[16] is for Card[0]. SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register (mask bit 1 enables interrupt; 0 masks interrupt). In MMC-Ver3.3-only mode, bits always 0. **Value** **Description** 0x1 SDIO interrupt from card 0x0 No SDIO interrupt from card	RO	0x0
15	ebe	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 End-bit error Mask 0x1 End-bit error No Mask	RO	0x0
14	acd	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Auto command done Mask 0x1 Auto command done No Mask	RO	0x0

Bit	Name	Description	Access	Reset
13	`strerr`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Start-bit error Mask 0x1 Start-bit error No Mask	RO	0x0
12	`hlwerr`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Hardware locked write error Mask 0x1 Hardware locked write error No Mask	RO	0x0
11	`fifoovunerr`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 FIFO underrun/overrun error Mask 0x1 FIFO underrun/overrun error No Mask	RO	0x0
10	`dshto`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Data starvation by host timeout Mask 0x1 Data starvation by host timeout No Mask	RO	0x0
9	`datardto`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Data read timeout Mask 0x1 Data read timeout No Mask	RO	0x0
8	`respto`	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Response timeout Mask 0x1 Response timeout No Mask	RO	0x0

Bit	Name	Description	Access	Reset
7	datacrcerr	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Data CRC error Mask 0x1 Data CRC error No Mask	RO	0x0
6	respcrcerr	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Response CRC error Mask 0x1 Response CRC error No Mask	RO	0x0
5	rxfifodr	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Receive FIFO data request Mask 0x1 Receive FIFO data request No Mask	RO	0x0
4	dttxfifodr	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Transmit FIFO data request Mask 0x1 Transmit FIFO data request No Mask	RO	0x0
3	dt	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Data transfer over Mask 0x1 Data transfer over No Mask	RO	0x0
2	cmd_done	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Command Done Mask 0x1 Command Done No Mask	RO	0x0

Bit	Name	Description	Access	Reset
1	resp	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Response error Mask 0x1 Response error No Mask	RO	0x0
0	cd	Interrupt enabled only if corresponding bit in interrupt mask register is set. **Value** **Description** 0x0 Card Detected Mask 0x1 Card Detected No Mask	RO	0x0

rintsts

Interrupt Status Before Masking.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704044

Offset: 0x44

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															sdio_interrupt RW 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ebe RW 0x0	acd RW 0x0	sbe RW 0x0	hle RW 0x0	frun RW 0x0	hto RW 0x0	bds RW 0x0	bar RW 0x0	dcrc RW 0x0	rcrc RW 0x0	rxdr RW 0x0	txdr RW 0x0	dto RW 0x0	cmd RW 0x0	re RW 0x0	cd RW 0x0

rintsts Fields

Bit	Name	Description	Access	Reset
16	sdio_interrupt	Interrupt from SDIO card. **Value** **Description** 0x1 SDIO interrupt from card bit 0x0 No SDIO interrupt from card bi	RW	0x0
15	ebe	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 End-bit error (read)/write no CRC (EBE) 0x1 Clears End-bit error (read)/write no CRC (EBE)	RW	0x0
14	acd	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Auto command done (ACD) 0x1 Clear Auto command done (ACD	RW	0x0
13	sbe	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Start-bit error (SBE) 0x1 Clears Start-bit error (SBE)	RW	0x0
12	hle	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Hardware locked write error (HLE) 0x1 Clears Hardware locked write error (HLE)	RW	0x0

Bit	Name	Description	Access	Reset
11	`frun`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 FIFO underrun/overrun error (FRUN) 0x1 Clear FIFO underrun/overrun error (FRUN)	`RW`	`0x0`
10	`hto`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Data starvation-by-host timeout (HTO) / Volt_switch_int 0x1 Clears Data starvation-by-host timeout (HTO) /Volt_switch_int	`RW`	`0x0`
9	`bds`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Data read timeout (DRTO)/Boot Data Start (BDS) 0x1 Clears Data read timeout (DRTO)/Boot Data Start (BDS)	`RW`	`0x0`
8	`bar`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Response timeout (RTO)/Boot Ack Received (BAR) 0x1 Clears Response timeout (RTO)/Boot Ack Received (BAR)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
7	dcrc	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Data CRC error (DCRC) 0x1 Clears Data CRC error (DCRC)	RW	0x0
6	rcrc	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Response CRC error (RCRC) 0x1 Clears Response CRC error (RCRC)	RW	0x0
5	rxdr	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Receive FIFO data request (RXDR) 0x1 Clears Receive FIFO data request (RXDR)	RW	0x0
4	txdr	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Transmit FIFO data request (TXDR) 0x1 Clears Transmit FIFO data request (TXDR)	RW	0x0
3	dto	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** **Description** 0x0 Data transfer over (DTO) 0x1 Clears Data transfer over (DTO)	RW	0x0

Bit	Name	Description	Access	Reset
2	`cmd`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** / **Description** 0x0 Command done (CD) 0x1 Clears Command done (CD)	`RW`	`0x0`
1	`re`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** / **Description** 0x0 Response error (RE) 0x1 Clears Response error (RE)	`RW`	`0x0`
0	`cd`	Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. **Value** / **Description** 0x0 Card detect (CD) 0x1 Clears Card detect (CD)	`RW`	`0x0`

status

Reports various operting status conditions.

Module Instance	Base Address	Register Address
`sdmmc`	`0xFF704000`	`0xFF704048`

Offset: `0x48`

Access: `RO`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="13">fifo_count
RO 0x0</td><td>response _index
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">response_index
RO 0x0</td><td>data_state_mc_busy
RO 0x0</td><td>data_busy
RO 0x0</td><td>data_3_status
RO 0x1</td><td colspan="4">command_fsm_states
RO 0x0</td><td>fifo_full
RO 0x0</td><td>fifo_empty
RO 0x1</td><td>fifo_tx_watermark
RO 0x1</td><td>fifo_rx_watermark
RO 0x0</td></tr>
</table>

status Fields

Bit	Name	Description	Access	Reset
29:17	fifo_count	FIFO count - Number of filled locations in FIFO	RO	0x0
16:11	response_index	Index of previous response, including any auto-stop sent by core	RO	0x0
10	data_state_mc_busy	Data transmit or receive state-machine is busy. **Value** **Description** 0x1 Data State MC busy 0x0 Data State MC not busy	RO	0x0
9	data_busy	Inverted version of raw selected card_data[0]. The default can be cardpresent or not present depend on cdata_in. **Value** **Description** 0x1 card data busy 0x0 card data not busy	RO	0x0
8	data_3_status	Raw selected card_data[3]; checks whether card is present. The default can be cardpresent or not present depend on cdata_in. **Value** **Description** 0x1 Card Present 0x0 Card Not Present	RO	0x1

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>7:4</td><td><code>command_fsm_states</code></td><td>The command FSM state.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Idle, Wait for CCS, Send CCSD, or Boot Mode</td></tr>
<tr><td>0x1</td><td>Send init sequence</td></tr>
<tr><td>0x2</td><td>Tx cmd start bit</td></tr>
<tr><td>0x3</td><td>Tx cmd tx bit</td></tr>
<tr><td>0x4</td><td>Tx cmd index + arg</td></tr>
<tr><td>0x5</td><td>Tx cmd crc7</td></tr>
<tr><td>0x6</td><td>Tx cmd end bit</td></tr>
<tr><td>0x7</td><td>Rx resp start bit</td></tr>
<tr><td>0x8</td><td>Rx resp IRQ response</td></tr>
<tr><td>0x9</td><td>Rx resp tx bit</td></tr>
<tr><td>0xa</td><td>Rx resp cmd idx</td></tr>
<tr><td>0xb</td><td>Rx resp data</td></tr>
<tr><td>0xc</td><td>Rx resp crc7</td></tr>
<tr><td>0xd</td><td>Rx resp end bit</td></tr>
<tr><td>0xe</td><td>Cmd path wait NCC</td></tr>
<tr><td>0xf</td><td>Wait: CMD-to-reponse turnaround</td></tr>
</table></td><td><code>RO</code></td><td><code>0x0</code></td></tr>
<tr><td>3</td><td><code>fifo_full</code></td><td>FIFO is full status.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>FIFO is full</td></tr>
<tr><td>0x1</td><td>FIFO is not full</td></tr>
</table></td><td><code>RO</code></td><td><code>0x0</code></td></tr>
<tr><td>2</td><td><code>fifo_empty</code></td><td>FIFO is empty status.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>FIFO is empty</td></tr>
<tr><td>0x0</td><td>FIFO not empty</td></tr>
</table></td><td><code>RO</code></td><td><code>0x1</code></td></tr>
</table>

Bit	Name	Description	Access	Reset
1	`fifo_tx_watermark`	FIFO reached Transmit watermark level; not qualified with data transfer. **Value** **Description** 0x1 FIFO reached transmit watermark level: not qualified with data transfer. 0x0 FIFO not at transmit watermark Level	RO	0x1
0	`fifo_rx_watermark`	FIFO reached Receive watermark level; not qualified with data transfer **Value** **Description** 0x0 FIFO reached watermark level; not qualified with data transfer. 0x1 FIFO not at watermark Level	RO	0x0

fifoth

DMA and FIFO Control Fields.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70404C

Offset: `0x4C`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="3">dw_dma_multiple_transaction_size
RW 0x0</td><td colspan="12">rx_wmark
RW 0x3FF</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="12">tx_wmark
RW 0x0</td></tr>
</table>

fifoth Fields

Bit	Name	Description	Access	Reset
30:28	dw_dma_multiple_ transaction_size	Burst size of multiple transaction; should be programmed same as DMA controller multiple-transaction-size SRC/DEST_MSIZE. The units for transfers is 32 bits. A single transfer would be signalled based on this value. Value should be sub-multiple of 512. Allowed combinations for MSize and TX_WMark. **Value** — **Description** 0x0 — Msize 1 and TX_WMARK 1-1023 0x1 — Msize 4 and TX_WMARK 256 0x2 — Msize 8 and TX_WMARK 128 0x3 — Msize 16 and TX_WMARK 64 0x5 — Msize 1 and RX_WMARK 512 0x6 — Msize 4 and RX_WMARK 128 0x7 — Msize 8 and RX_WMARK 64	RW	0x0
27:16	rx_wmark	FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold (RXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. 12 bits - 1 bit less than FIFO-count of status register, which is 13 bits. Limitation: RX_WMark <= 1022 Recommended: 511; means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.	RW	0x3FF

Bit	Name	Description	Access	Reset
11:0	tx_wmark	FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If Interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated, regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold (TXDR) interrupt is enabled, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes (not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. 12 bits - 1 bit less than FIFO-count of status register, which is 13 bits. Limitation: TX_WMark >= 1; Recommended: FIFO_DEPTH/2 = 512; (means less than or equal to 512)	RW	0x0

cdetect

Determines if card is present.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704050

Offset: 0x50

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															card_detect_n RO 0x1

cdetect Fields

Bit	Name	Description	Access	Reset
0	card_detect_n	Value on sdmmc_cd_i input port. **Value** **Description** 0x1 Card not Detected 0x0 Card Detected	RO	0x1

wrtprt

See Field Description.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704054

Offset: 0x54

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															write_protect RO 0x1

wrtprt Fields

Bit	Name	Description	Access	Reset
0	write_protect	Value on sdmmc_wp_i input port. **Value** **Description** 0x1 Write Protect Enabled 0x0 Write Protect Disabled	RO	0x1

tcbcnt

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70405C

Offset: 0x5C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
trans_card_byte_count RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trans_card_byte_count RO 0x0															

tcbcnt Fields

Bit	Name	Description	Access	Reset
31:0	trans_card_byte_count	Number of bytes transferred by CIU unit to card.	RO	0x0

tbbcnt

Tracks number of bytes transferred between Host and FIFO.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704060

Offset: 0x60

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
trans_fifo_byte_count RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trans_fifo_byte_count RO 0x0															

tbbcnt Fields

Bit	Name	Description	Access	Reset
31:0	trans_fifo_byte_count	Number of bytes transferred between Host/DMA memory and BIU FIFO. In 32-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. Both TCBCNT and TBBCNT share same coherency register.	RO	0x0

debnce

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704064

Offset: `0x64`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								debounce_count RW 0xFFFFFF							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
debounce_count RW 0xFFFFFF															

debnce Fields

Bit	Name	Description	Access	Reset
23:0	debounce_count	Number of host clocks l4_mp_clk used by debounce filter logic; typical debounce time is 5-25 ms.	RW	0xFFFFFF

usrid

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704068

Offset: `0x68`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
usr_id RW 0x7967797															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usr_id RW 0x7967797															

usrid Fields

Bit	Name	Description	Access	Reset
31:0	usr_id	User identification field; Value is 0x7967797.	RW	0x7967797

verid

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF70406C

Offset: `0x6C`

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ver_id RO 0x5342240A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ver_id RO 0x5342240A															

verid Fields

Bit	Name	Description	Access	Reset
31:0	ver_id	Synopsys version id. Current value is 32'h5342240a	RO	0x53422 40A

hcon

Hardware configurations registers. Register can be used to develop configuration-independent software drivers.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704070

Offset: 0x70

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					aro RO 0x0	ncd RO 0x0		scfp RO 0x1	ihr RO 0x1	rios RO 0x0	dmadatawidth RO 0x1			dmaintf RO 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
haddrwidth RO 0xC						hdatawidth RO 0x1			hbus RO 0x0	nc RO 0x0					ct RO 0x1

hcon Fields

Bit	Name	Description	Access	Reset
26	aro	Area optimized **Value** **Description** 0x0 Not Optimized For Area	RO	0x0

Bit	Name	Description	Access	Reset
25:24	ncd	Number of clock dividers less one **Value** **Description** 0x0 One Clock Divider	RO	0x0
23	scfp	Clock False Path **Value** **Description** 0x1 Clock False Path Set	RO	0x1
22	ihr	Implement hold register **Value** **Description** 0x1 Implements Hold Register	RO	0x1
21	rios	FIFO RAM location **Value** **Description** 0x0 FIFO RAM Outside IP Core	RO	0x0
20:18	dmadatawidth	Encodes bit width of external DMA controller interface. Doesn't apply to the SD/MMC because it has no external DMA controller interface. **Value** **Description** 0x1 32-bits wide	RO	0x1
17:16	dmaintf	DMA interface type **Value** **Description** 0x0 No External DMA Controller Interface (SD/MMC has its own internal DMA Controller	RO	0x0
15:10	haddrwidth	Slave bus address width less one **Value** **Description** 0xc Width 13 Bits	RO	0xC
9:7	hdatawidth	Slave bus data width **Value** **Description** 0x1 Width 32 Bits	RO	0x1

Bit	Name	Description	Access	Reset
6	hbus	Slave bus type. **Value** **Description** 0x0 APB Bus	RO	0x0
5:1	nc	Maximum number of cards less one **Value** **Description** 0x0 1 Card	RO	0x0
0	ct	Supported card types **Value** **Description** 0x1 Card Type SD/MMC	RO	0x1

uhs_reg

UHS-1 Register

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704074

Offset: 0x74

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="15">Reserved</td><td>ddr_reg
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>volt_reg
RW 0x0</td></tr>
</table>

uhs_reg Fields

Bit	Name	Description	Access	Reset
16	ddr_reg	Determines the voltage fed to the buffers by an external voltage regulator. **Value** / **Description** 0x0 Non-DDR mode 0x1 DDR mode	RW	0x0
0	volt_reg	Determines the voltage fed to the buffers by an external voltage regulator. These bits function as the output of the host controller and are fed to an external voltage regulator. The voltage regulator must switch the voltage of the buffers of a particular card to either 3.3V or 1.8V, depending on the value programmed in the register. **Value** / **Description** 0x0 Buffers supplied with 3.3V Vdd 0x1 Buffers supplied with 1.8V Vdd	RW	0x0

rst_n

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704078

Offset: 0x78

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															card_reset RW 0x1

rst_n Fields

Bit	Name	Description	Access	Reset
0	`card_reset`	This bit causes the cards to enter pre-idle state, which requires it to be re-initialized. **Value** — **Description** 0x1 — Active Mode 0x0 — Not Active Mode	RW	0x1

bmod

Details different bus operating modes.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704080

Offset: `0x80`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					pbl RO 0x0			de RW 0x0	dsl RW 0x0					fb RW 0x0	swr RW 0x0

bmod Fields

Bit	Name	Description	Access	Reset
10:8	`pbl`	These bits indicate the maximum number of beats to be performed in one IDMAC transaction. The IDMAC will always attempt to burst as specified in PBL each time it starts a Burst transfer on the host bus. This value is the mirror of MSIZE of FIFOTH register. In order to change this value, write the required value to FIFOTH register. This is an encode value as follows. **Value** **Description** 0x0 Transfer 1 0x1 Transfer 4 0x2 Transfer 8 0x3 Transfer 16 0x4 Transfer 32 0x5 Transfer 64 0x6 Transfer 128 0x7 Transfer 256	`RO`	`0x0`
7	`de`	Enables and Disables Internal DMA. **Value** **Description** 0x1 IDMAC Enable 0x0 IDMAC Disable	`RW`	`0x0`
6:2	`dsl`	Specifies the number of HWord/Word/Dword (depending on 16/32/64-bit bus) to skip between two unchained descriptors.	`RW`	`0x0`
1	`fb`	Controls whether the AHB Master interface performs fixed burst transfers or not. Will use only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. **Value** **Description** 0x1 AHB Master Fixed Burst 0x0 Non Fixed Burst - default	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	swr	This bit resets all internal registers of the DMA Controller. It is automatically cleared after 1 clock cycle. **Value** **Description** 0x1 Resets DMA Internal Registers 0x0 No reset - default	RW	0x0

pldmnd

See Field Description.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704084

Offset: 0x84

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pd WO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pd WO 0x0															

pldmnd Fields

Bit	Name	Description	Access	Reset
31:0	pd	If the OWN bit of a descriptor is not set, the FSM goes to the Suspend state. The host needs to write any value into this register for the IDMAC FSM to resume normal descriptor fetch operation.	WO	0x0

dbaddr

See Field Descriptor

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704088

Offset: 0x88

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
sdl RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sdl RW 0x0														Reserved	

dbaddr Fields

Bit	Name	Description	Access	Reset
31:2	`sdl`	Contains the base address of the First Descriptor. This is the byte address divided by 4.	`RW`	`0x0`

idsts

Sets Internal DMAC Status Fields

Module Instance	Base Address	Register Address
`sdmmc`	`0xFF704000`	`0xFF70408C`

Offset: `0x8C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															fsm RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
fsm RO 0x0			eb RO 0x0			ais RW 0x0	nis RW 0x0	Reserved		ces RW 0x0	du RW 0x0	Reserved	fbe RW 0x0	ri RW 0x0	ti RW 0x0

idsts Fields

Bit	Name	Description	Access	Reset
16:13	fsm	DMAC FSM present state. **Value** / **Description** 0x0 DMA IDLE 0x1 DMA SUSPEND 0x2 DESC_RD 0x3 DESC_CHK 0x4 DMA RD REQ WAIT 0x5 DMA WR REQ WAIT 0x6 DMA RD 0x7 DMA WR 0x8 DESC CLOSE	RO	0x0
12:10	eb	Indicates the type of error that caused a Bus Error. Valid only with Fatal Bus Error bit (IDSTS[2]) set. This field does not generate an interrupt. **Value** / **Description** 0x1 Host Abort during transmission Status Bit 0x2 Host Abort received during reception Status Bit	RO	0x0
9	ais	Logical OR of the following: IDSTS[2] - Fatal Bus Interrupt IDSTS[4] - DU bit Interrupt IDSTS[5] - Card Error Summary Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. **Value** / **Description** 0x1 Clears Abnormal Summary Interrupt Status Bit 0x0 No Clear Abnormal Summary Interrupt Status Bit	RW	0x0

Bit	Name	Description	Access	Reset
8	nis	Logical OR of the following: IDSTS[0] - Transmit Interrupt IDSTS[1] - Receive Interrupt Only unmasked bits affect this bit. This is a sticky bit and must be cleared each time a corresponding bit that causes NIS to be set is cleared. **Value** **Description** 0x1 Clears Normal Interrupt Summary Status Bit 0x0 No Clear Normal Interrupt Summary Status Bit	RW	0x0
5	ces	Indicates the status of the transaction to/from the card; also present in RINTSTS. Indicates the logical OR of the following bits: EBE - End Bit Error RTO - Response Timeout/Boot Ack Timeout RCRC - Response CRC SBE - Start Bit Error DRTO - Data Read Timeout/BDS timeout DCRC - Data CRC for Receive RE - Response Error **Value** **Description** 0x1 Clears Card Error Summary Interrupt Status Bit 0x0 No Clear Card Error Summary Interrupt Status Bit	RW	0x0
4	du	This status bit is set when the descriptor is unavailable due to OWN bit = 0 (DES0[31] =0). **Value** **Description** 0x1 Clears Descriptor Unavailable Interrupt Status Bit 0x0 No Clear of Descriptor Unavailable Interrupt Status Bit	RW	0x0
2	fbe	Indicates that a Bus Error occurred (IDSTS[12:10]). When setthe DMA disables all its bus accesses. **Value** **Description** 0x1 Clears Fatal Bus Error Interrupt Status Bit 0x0 No Clear of Fatal Bus Error Interrupt Status Bit	RW	0x0

Bit	Name	Description	Access	Reset
1	ri	Indicates the completion of data reception for a descriptor **Value** **Description** 0x1 Clears Receive Interrupt Status Bit 0x0 No Clear of Receive Interrupt Status Bit	RW	0x0
0	ti	Indicates that data transmission is finished for a descriptor. **Value** **Description** 0x1 Clears Transmit Interrupt Status Bit 0x0 No Clear of Transmit Interrupt Status Bit	RW	0x0

idinten

Various DMA Interrupt Enable Status

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ai RW 0x0	ni RW 0x0	Reserved		ces RW 0x0	du RW 0x0	Reserved	fbe RW 0x0	ri RW 0x0	ti RW 0x0

idinten Fields

Bit	Name	Description	Access	Reset
9	ai	This bit enables the following bits: IDINTEN[2] - Fatal Bus Error Interrupt IDINTEN[4] - DU Interrupt IDINTEN[5] - Card Error Summary Interrupt **Value** **Description** 0x1 Abnormal Interrupt Summary is enabled 0x0 Abnormal Interrupt Summary is disabled	RW	0x0
8	ni	Enable and Disable Normal Interrupt Summary **Value** **Description** 0x1 Normal Interrupt Summary is enabled 0x0 Normal Interrupt Summary is disabled	RW	0x0
5	ces	Enable and disable Card Error Interrupt Summary **Value** **Description** 0x1 Card Error Summary Interrupt is enabled 0x0 Card Error Summary Interrupt is disabled	RW	0x0
4	du	When set along with Abnormal Interrupt Summary Enable, the DU interrupt is enabled. **Value** **Description** 0x1 Descriptor Unavailable Interrupt is enabled 0x0 Descriptor Unavailable Interrupt is disabled	RW	0x0
2	fbe	When set with Abnormal Interrupt Summary Enable, the Fatal Bus Error Interrupt is enabled. **Value** **Description** 0x1 Fatal Bus Error Interrupt is enabled 0x0 Fatal Bus Error Interrupt is disabled	RW	0x0

Bit	Name	Description	Access	Reset
1	ri	Enables and Disables Receive Interrupt when Normal Interrupt Summary Enable is set. **Value** **Description** 0x1 Receive Interrupt is enabled 0x0 Receive Interrupt is disabled	RW	0x0
0	ti	Enables and Disables Transmit Interrupt when Normal Interrupt Summary Enable is set. **Value** **Description** 0x1 Transmit Interrupt is enabled 0x0 Transmit Interrupt is disabled	RW	0x0

dscaddr

See Field Description.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704094

Offset: 0x94

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hda RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hda RO 0x0															

dscaddr Fields

Bit	Name	Description	Access	Reset
31:0	hda	Cleared on reset. Pointer updated by IDMAC during operation. This register points to the start address of the current descriptor read by the IDMAC.	RO	0x0

bufaddr

See Field Description.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704098

Offset: 0x98

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hba RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hba RO 0x0															

bufaddr Fields

Bit	Name	Description	Access	Reset
31:0	hba	Cleared on Reset. Pointer updated by IDMAC during operation. This register points to the current Data Buffer Address being accessed by the IDMAC.	RO	0x0

cardthrctl

See Field descriptions

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704100

Offset: 0x100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				cardrdthreshold RW 0x0											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															cardrdthren RW 0x0

cardthrctl Fields

Bit	Name	Description	Access	Reset
27:16	cardrdthreshold	Card Read Threshold size	RW	0x0

Bit	Name	Description	Access	Reset
0	cardrdthren	Host Controller initiates Read Transfer only if CardRdThreshold amount of space is available in receive FIFO. **Value** **Description** 0x1 Card Read Threshold is enabled 0x0 Card Read Threshold is disabled	RW	0x0

back_end_power_r

See Field Description

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704104

Offset: 0x104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
back_end_power RW 0x0															

back_end_power_r Fields

Bit	Name	Description	Access	Reset
15:0	back_end_power	Back end power operation. **Value** **Description** 0x1 Back-end Power supplied to card only 1 card 0x0 Off Reset	RW	0x0

data

Provides read/write access to data FIFO. Addresses 0x200 and above are mapped to the data FIFO. More than one address is mapped to data FIFO so that FIFO can be accessed using bursts.

Module Instance	Base Address	Register Address
sdmmc	0xFF704000	0xFF704200

Offset: 0x200

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

data Fields

Bit	Name	Description	Access	Reset
31:0	value	Provides read/write access to data FIFO.	RW	0x0

Document Revision History

Table 14-35: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address maps and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.1	• Added programming model section. • Reorganized programming information. • Added information about ECCs. • Added pin listing. • Updated clocks section.
January 2012	1.0	Initial release

Quad SPI Flash Controller 15

2014.07.31

cv_54012 Subscribe Send Feedback

The hard processor system (HPS) provides a quad serial peripheral interface (SPI) flash controller for access to serial NOR flash devices. The quad SPI flash controller supports standard SPI flash devices as well as high-performance dual and quad SPI flash devices. The quad SPI flash controller is based on Cadence Quad SPI Flash Controller (QSPI_FLASH_CTRL).

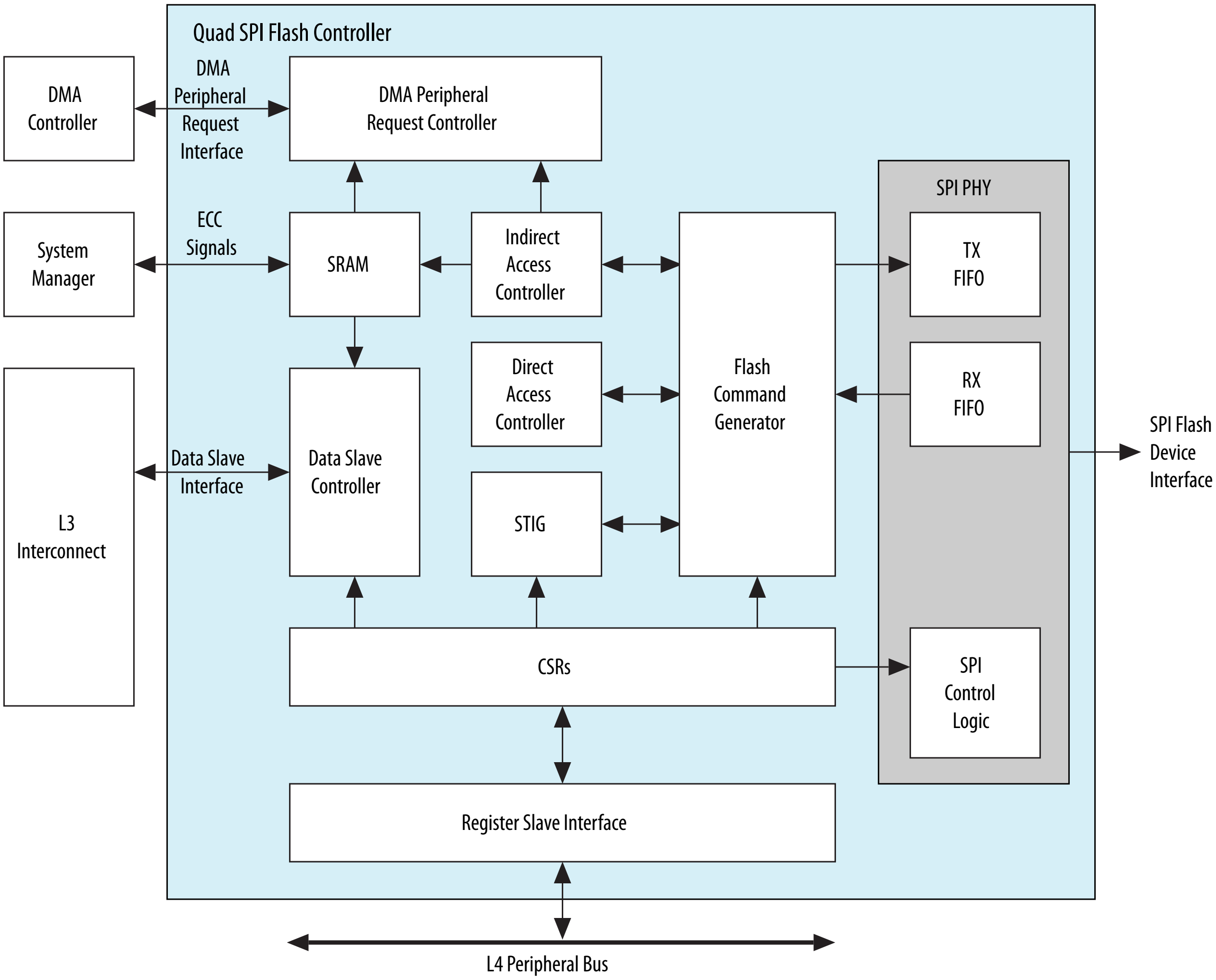
Features of the Quad SPI Flash Controller

The quad SPI flash controller supports the following features:

- Single, dual, and quad I/O commands
- Device frequencies up to 108 MHz
- Direct access and indirect access modes
- External direct memory access (DMA) controller support for indirect transfers
- Configurable clock polarity and phase
- Programmable write-protected regions
- Programmable delays between transactions
- Programmable device sizes
- Read data capture tuning
- Local buffering with error correction code (ECC) logic for indirect transfers
- Up to four devices
- eXecute-In-Place (XIP) mode

Quad SPI Flash Controller Block Diagram and System Integration

Figure 15-1: Quad SPI Flash Controller Block Diagram and System Integration



The quad SPI controller consists of the following blocks and interfaces:

- Register slave interface—Slave interface that provides access to the control and status registers (CSRs)
- Data slave controller—Slave interface and controller that provides the following functionality:
  - Performs data transfers to and from the level 3 (L3) interconnect
  - Validates incoming accesses
  - Performs byte or half-word reordering
  - Performs write protection
  - Forwards transfer requests to direct and indirect controller
- Direct access controller—provides memory-mapped slaves direct access to the flash memory
- Indirect access controller—provides higher-performance access to the flash memory through local buffering and software transfer requests

- Software triggered instruction generator (STIG)—generates flash commands through the flash command register (`flashcmd`) and provides low-level access to flash memory
- Flash command generator—generates flash command and address instructions based on instructions from the direct and indirect access controllers or the STIG
- DMA peripheral request controller—issues requests to the DMA peripheral request interface to communicate with the external DMA controller
- SPI PHY—serially transfers data and commands to the external SPI flash devices

Functional Description of the Quad SPI Flash Controller

Overview

Note: Terms used in this section are defined in detail in the sections that follow.

The quad SPI flash controller uses the register slave interface to select the operation modes and configure the data slave interface for data transfers. The quad SPI flash controller uses the data slave interface for direct and indirect accesses, and the register slave interface for software triggered instruction generator (STIG) operation and SPI legacy mode accesses.

Accesses to the data slave are forwarded to the direct or indirect access controller. If the access address is within the configured indirect address range, the access is sent to the indirect access controller.

Data Slave Interface

The quad SPI flash controller uses the data slave interface for direct, indirect, and SPI legacy mode accesses.

Note:
- The data slave is 32 bits wide.
- Byte, half-word, and word accesses are permitted.
- For write accesses:
  - only incrementing bursts are supported
  - only of sizes 1, 4, 8, and 16 transfers
- For read accesses
  - all burst types and sizes are supported

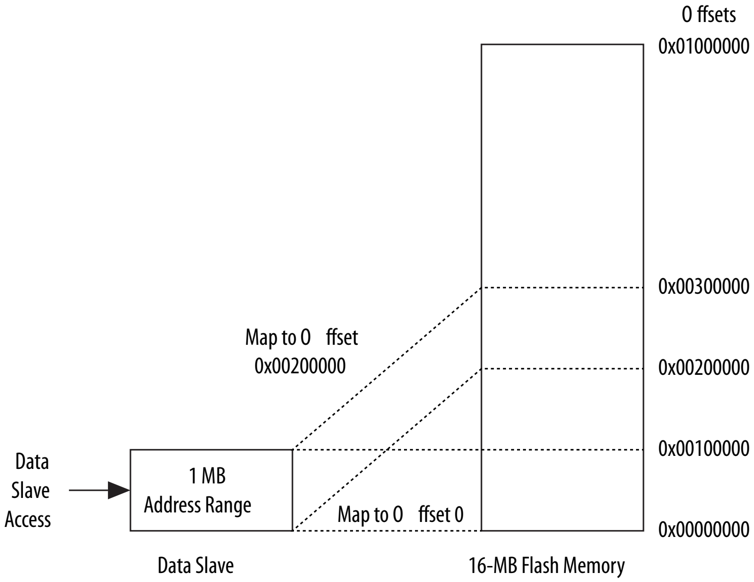
Direct Access Mode

Direct access mode is memory mapped and can be used to both access and directly execute code from external FLASH memory. Any incoming AMBA data slave interface access that is not recognized as being within the programmable indirect trigger region is assumed to be a direct access and is serviced by the direct access controller.

Note: Accesses that use the direct access controller do not use the embedded SRAM.

Data Slave Remapping Example

Figure 15-2: Data Slave Remapping Example



To remap the data slave to access other 1 MB regions of the flash device, enable address remapping in the enable ARM® AMBA® advanced high speed bus (AHB) address remapping field (`enahbremap`) of the `cfg` register. All incoming data slave accesses remap to the offset specified in the remap address register (`remapaddr`).

The 20 LSBs of incoming addresses are used for accessing the 1 MB region and the higher bits are ignored.

Note: The quad SPI controller does not issue any error status for accesses that lie outside the connected flash memory space.

AHB

The data slave interface is throttled as the read or write burst is carried out. The latency is designed to be as small as possible and is kept to a minimum when the use of XIP read instructions are enabled.

FLASH erase operations, which may be required before a page write, are triggered by software using the documented programming interface. They are not issued automatically.

Once a page program cycle has been started, the QSPI Flash Controller will automatically poll for the write cycle to complete before allowing any further data slave interface accesses to complete. This is achieved by holding any subsequent AHB direct accesses in wait state.

Indirect Access Mode

In indirect access mode, flash data is temporarily buffered in the quad SPI controller's static RAM (SRAM). Software controls and triggers indirect accesses through the register slave interface. The controller transfers data through the data slave interface.

Indirect Read Operation

An indirect read operation reads data from the flash memory, places the data into the SRAM, and transfers the data to an external master through the data slave interface. The indirect read operations are controlled by the following registers:

- Indirect read transfer register (`indrd`)
- Indirect read transfer watermark register (`indrdwater`)
- Indirect read transfer start address register (`indrdstaddr`)
- Indirect read transfer number bytes register (`indrdcnt`)
- Indirect address trigger register (`indaddrtrig`)

These registers need to be configured prior to issuing indirect read operations. The start address needs to be defined in the `indrdstaddr` register and the total number of bytes to be fetched is specified in the `indircnt` register. Writing 1 to the start indirect read bit (`start`) of the `indrd` register triggers the indirect read operation from the flash memory to populate the SRAM with the returned data.

To read data from the flash device into the SRAM, an external master issues 32-bit read transactions to the data slave interface. The address of the read access must be in the indirect address range. You can configure the indirect address through the `indaddrtrig` register. The external master can issue 32-bit reads until the last word of an indirect transfer. On the final read, the external master may issue a 32-bit, 16-bit or 8-bit read to complete the transfer. If there are less than four bytes of data to read on the last transfer, the external master can still issue a 32-bit read and the quad SPI controller will pad the upper bits of the response data with zeros.

Assuming the requested data is present in the SRAM at the time the data slave read is received by the quad SPI controller, the data is fetched from SRAM and the response to the read burst is achieved with minimum latency. If the requested data is not immediately present in the SRAM, the data slave interface enters a wait state until the data has been read from flash memory into SRAM. Once the data has been read from SRAM by the external master, the quad SPI controller frees up the associated resource in the SRAM. If the SRAM is full, reads on the SPI interface are backpressured until space is available in the SRAM. The quad SPI controller completes any current read burst, waits for SRAM to free up, and issues a new read burst at the address where the previous burst was terminated.

The processor can also use the SRAM fill level in the SRAM fill register (`sramfill`) to control when data should be fetched from the SRAM.

Another alternative is to use the fill level watermark of the SRAM, which you configure in the `indrdwater` register. When the SRAM fill level passes the watermark level, the indirect transfer watermark interrupt is generated. You can disable the watermark feature by writing zero to the `indrdwater` register.

For the final bytes of data read by the quad SPI controller and placed in the SRAM, if the watermark level is greater than zero, the indirect transfer watermark interrupt is generated even when the actual SRAM fill level has not risen above the watermark.

If the address of the read access is outside the range of the indirect trigger address, one of the following actions occurs:

- When direct access mode is enabled, the read uses direct access mode.
- When direct access mode is disabled, the slave returns an error back to the requesting master.

You can cancel an indirect operation by setting the cancel indirect read bit (`cancel`) of the `indrd` register to 1. For more information, refer to the *"Indirect Read Operation with DMA Disabled"* section.

Related Information

Indirect Read Operation with DMA Disabled on page 15-15

Indirect Write Operation

An indirect write operation programs data from the SRAM to the flash memory. The indirect write operations are controlled by the following registers:

- Indirect write transfer register (`indwr`)
- Indirect write transfer watermark register (`indwrwater`)
- Indirect write transfer start address register (`indwrstaddr`)
- Indirect write transfer number bytes register (`indwrcnt`)
- `indaddrtrig` register

These registers need to be configured prior to issuing indirect write operations. The start address needs to be defined in the `indwrstaddr` register and the total number of bytes to be written is specified in the `indwrcnt` register. The start indirect write bit (`start`) of the `indwr` register triggers the indirect write operation from the SRAM to the flash memory.

To write data from the SRAM to the flash device, an external master issues 32-bit write transactions to the data slave. The address of the write access must be in the indirect address range. You can configure the indirect address through the `indaddrtrig` register. The external master can issue 32-bit writes until the last word of an indirect transfer. On the final write, the external master may issue a 32-bit, 16-bit or 8-bit write to complete the transfer. If there are less than four bytes of data to write on the last transfer, the external master can still issue a 32-bit write and the quad SPI controller discards the extra bytes.

The SRAM size can limit the amount of data that the quad SPI controller can accept from the external master. If the SRAM is not full at the point of the write access, the data is pushed to the SRAM with minimum latency. If the external master attempts to push more data to the SRAM than the SRAM can accept, the quad SPI controller backpressures the external master with wait states. When the SRAM resource is freed up by pushing the data from SRAM to the flash memory, the SRAM is ready to receive more data from the external master. When the SRAM holds an equal or greater number of bytes than the size of a flash page, or when the SRAM holds all the remaining bytes of the current indirect transfer, the quad SPI controller initiates a write operation to the flash memory.

The processor can also use the SRAM fill level, in the `sramfill` register, to control when to write more data into the SRAM.

Alternatively, you can configure the fill level watermark of the SRAM in the `indwrwater` register. When the SRAM fill level falls below the watermark level, an indirect transfer watermark interrupt is generated to tell software to write the next page of data to SRAM. Because the quad SPI controller initiates non-end-of-data writes to the flash memory only when the SRAM contains a full flash page of data, you must set the watermark level to a value greater than one flash page to avoid the system stalling. You can disable the watermark feature by writing zero to the `indwrwater` register.

If the address of the write access is outside the range of the indirect trigger address, one of the following actions occurs:

- When direct access mode is enabled, the write uses direct access mode.
- When direct access mode is disabled, the slave returns an error back to the requesting master.

You can cancel an indirect operation by setting the cancel indirect write bit (`cancel`) of the `indwr` register to 1. For more information, refer to the *"Indirect Write Operation with DMA Disabled"* section.

Related Information

Indirect Write Operation with DMA Disabled on page 15-16

Consecutive Reads and Writes

It is possible to trigger two indirect operations at a time by triggering the `start` bit of the `indrd` or `indwr` register twice in short succession. The second operation can be triggered while the first operation is in progress. For example, software may trigger an indirect read or write operation while an indirect write operation is in progress. The corresponding start and count registers must be configured properly before software triggers each transfer operation.

This approach allows for a short turnaround time between the completion of one indirect operation and the start of a second operation. Any attempt to queue more than two operations causes the indirect read reject interrupt to be generated.

SPI Legacy Mode

SPI legacy mode allows software to access the internal TX FIFO and RX FIFO buffers directly, thus bypassing the direct, indirect and STIG controllers. Software accesses the TX FIFO and RX FIFO buffers by writing any value to any address through the data slave while legacy mode is enabled. You can enable legacy mode with the legacy IP mode enable bit (`enlegacyip`) of the `cfg` register.

Legacy mode allows the user to issue any flash instruction to the flash device, but imposes a heavy software overhead in order to manage the fill levels of the FIFO buffers effectively. The legacy SPI mode is bidirectional in nature, with data continuously being transferred both directions while the chip select is enabled. If the driver only needs to read data from the flash device, dummy data must be written to ensure the chip select stays active, and vice versa for write transactions.

For example, to perform a basic read of four bytes to a flash device that has three address bytes, software must write a total of eight bytes to the TX FIFO buffer. The first byte would be the instruction opcode, the next three bytes are the address, and the final four bytes would be dummy data to ensure the chip select stays active while the read data is returned. Similarly, because eight bytes were written to the TX FIFO buffer, software should expect eight bytes to be returned in the RX FIFO buffer. The first four bytes of this would be discarded, leaving the final four bytes holding the data read from the device.

Because the TX FIFO and RX FIFO buffers are four bytes deep each, software must maintain the FIFO buffer levels to ensure the TX FIFO buffer does not underflow and the RX FIFO buffer does not overflow. Interrupts are provided to indicate when the fill levels pass the watermark levels, which are configurable through the TX threshold register (`txtresh`) and RX threshold register (`rxtresh`).

Register Slave Interface

The quad SPI flash controller uses the register slave interface to configure the quad SPI controller through the quad SPI configuration registers, and to access flash memory under software control, through the `flashcmd` register in the STIG.

STIG Operation

The Software Triggered Instruction Generator (STIG) is used to access the volatile and non-volatile configuration registers, the legacy SPI status register, and other status and protection registers. The STIG also is used to perform ERASE functions. The direct and indirect access controllers are used only to transfer data. The `flashcmd` register uses the following parameters to define the command to be issued to the flash device:

- Instruction opcode
- Number of address bytes
- Number of dummy bytes
- Number of write data bytes
- Write data
- Number of read data bytes

The address is specified through the flash command address register (`flashcmdaddr`). Once these settings have been specified, software can trigger the command with the execute command field (`execcmd`) of the flashcmd register and wait for its completion by polling the command execution status bit (cmdexecstat) of the flashcmd register. A maximum of eight data bytes may be read from the flash command read data lower (flashcmdrddatalo) and flash command read data upper (flashcmdrddataup) registers or written to the flash command write data lower (flashcmdwrdatalo) and flash command write data upper (flashcmdwrdataup) registers per command.

Commands issued through the STIG have a higher priority than all other read accesses and therefore interrupt any read commands being requested by the direct or indirect controllers. However, the STIG does not interrupt a write sequence that may have been issued through the direct or indirect access controller. In these cases, it might take a long time for the cmdexecstat bit of the flashcmd register indicates the operation is complete.

Note: Altera recommends using the STIG instead of the SPI legacy mode to access the flash device registers and perform erase operations.

Local Memory Buffer

The SRAM local memory buffer is a 128 by 32-bit (512 total bytes) memory and includes support for error correction code (ECC). The ECC logic provides outputs to notify the system manager when single-bit correctable errors are detected (and corrected) and when double-bit uncorrectable errors are detected. The ECC logic also allows the injection of single- and double-bit errors for test purposes.

The SRAM has two partitions, with the lower partition reserved for indirect read operations and the upper partition for indirect write operations, as shown in **Figure 15-1**. The size of the partitions is specified in the SRAM partition register (srampart), based on 32-bit word sizes. For example, to specify four bytes of storage, write the value 1. The value written to the indirect read partition size field (addr) defines the number of entries reserved for indirect read operations. For example, write the value 32 (0x20) to partition the 128-entry SRAM to 32 entries (25%) for read usage and 96 entries (75%) for write usage.

For more information about ECC, refer to the *System Manager* chapter in volume 3 of the Cyclone® V Device Handbook.

Related Information

System Manager on page 5-1

DMA Peripheral Request Controller

The DMA peripheral request controller is only used for the indirect mode of operation where data is temporarily stored in the SRAM. The quad SPI flash controller uses the DMA peripheral request interface to trigger the external DMA into performing data transfers between memory and the quad SPI controller.

There are two DMA peripheral request interfaces, one for indirect reads and one for indirect writes. The DMA peripheral request controller can issue two types of DMA requests, single or burst, to the external DMA. The number of bytes for each single or burst request is specified in the number of single bytes (numsglreqbytes) and number of burst bytes (numburstreqbytes) fields of the DMA peripheral register (dmaper). The DMA peripheral request controller splits the total amount of data to be transferred into a number of DMA burst and single requests by dividing the total number of bytes by the number of bytes specified in the burst request, and then dividing the remainder by the number of bytes in a single request.

Note: When programming the DMA controller, the burst request size must match the burst request size set in the quad SPI controller to avoid quickly reaching an overflow or underflow condition.

For indirect reads, the DMA peripheral request controller only issues DMA requests after the data has been retrieved from flash memory and written to SRAM. The rate at which DMA requests are issued depends on the watermark level. The indrdwater register defines the minimum fill level in bytes at which the DMA peripheral request controller can issue the DMA request. The higher this number is, the more data that must be buffered in SRAM before the external DMA moves the data. When the SRAM fill level passes the watermark level, the transfer watermark reached interrupt is generated.

For example, consider the following conditions:

- The total amount of data to be read using indirect mode is 256 bytes
- The SRAM watermark level is set at 128 bytes
- Software configures the burst type transfer size to 64 bytes

Under these conditions, the DMA peripheral request controller issues the first DMA burst request when the SRAM fill level passes 128 bytes (the watermark level). The DMA peripheral request controller triggers consecutive DMA burst requests as long as there is sufficient data in the SRAM to perform burst type requests. In this example, DMA peripheral request controller can issue at least two consecutive DMA burst requests to transfer a total of 128 bytes. If there is sufficient data in the SRAM, the DMA peripheral request controller requests the third DMA burst immediately. Otherwise the DMA peripheral request controller waits for the SRAM fill level to pass the watermark level again to trigger the next burst request. When the watermark level is triggered, there is sufficient data in the SRAM to perform the third and fourth burst requests to complete the entire transaction.

For the indirect writes, the DMA peripheral request controller issues DMA requests immediately after the transfer is triggered and continues to do so until the entire indirect write transfer has been transferred. The rate at which DMA requests are issued depends on the watermark level. The `indwrwater` register defines the maximum fill level in bytes at which the controller can issue the first DMA burst or single request. When the SRAM fill level falls below the watermark level, the transfer watermark reached interrupt is generated. When there is one flash page of data in the SRAM, the quad SPI controller initiates the write operation from SRAM to the flash memory.

Software can disable the DMA peripheral request interface with the `endma` field of the `cfg` register. If a master other than the DMA performs the data transfer for indirect operations, the DMA peripheral request interface must be disabled. By default, the indirect watermark registers are set to zero, which means the DMA peripheral request controller can issue DMA request as soon as possible.

For more information about the HPS DMA controller, refer to the *DMA Controller* chapter in volume 3 of the Cyclone® V Device Handbook.

Related Information

DMA Controller on page 16-1

Arbitration between Direct/Indirect Access Controller and STIG

When multiple controllers are active simultaneously, a fixed-priority arbitration scheme is used to arbitrate between each interface and access the external FLASH. The fixed priority is defined as follows, highest priority first.

1. The Indirect Access Write
2. The Direct Access Write
3. The STIG
4. The Direct Access Read
5. The Indirect Access Read

Each controller is back pressured while waiting to be serviced.

Configuring the Flash Device

For read and write accesses, software must initialize the device read instruction register (`devrd`) and the device write instruction register (`devwr`). These registers include fields to initialize the instruction opcodes that should be used as well as the instruction type, and whether the instruction uses single, dual or quad pins for address and data transfer. To ensure the quad SPI controller can operate from a reset state, the opcode registers reset to opcodes compatible with single I/O flash devices.

The quad SPI flash controller uses the instruction transfer width field (`instwidth`) of the `devrd` register to set the instruction transfer width for both reads and writes. There is no `instwidth` field in the `devwr` register. If instruction type is set to dual or quad mode, the address transfer width (`addrwidth`) and data transfer width (`datawidth`) fields of both registers are redundant because the address and data type is based on the instruction type. Thus, software can support the less common flash instructions where the opcode, address, and data are sent on two or four lanes. For most instructions, the opcodes are sent serially to the flash device, even for dual and quad instructions. One of the flash devices that supports instructions that can send the opcode over two or four lanes is the Micron N25Q128. For reference, Table 15-1 and Table 15-2 show how software should configure the quad SPI controller for each specific read and write instruction, respectively, supported by the Micron N25Q128 device.

Table 15-1: Quad SPI Configuration for Micron N25Q128 Device (Read Instructions)

Instruction	Lanes Used By Opcode	Lanes Used to Send Address	Lanes Used to Send Data	instwidth Value	addrwidth Value	datawidth Value
Read	1	1	1	0	0	0
Fast read	1	1	1	0	0	0
Dual output fast read (DOFR)	1	1	2	0	0	1
Dual I/O fast read (DIOFR)	1	2	2	0	1	1
Quad output fast read (QOFR)	1	1	4	0	0	2
Quad I/O fast read (QIOFR)	1	4	4	0	2	2
Dual command fast read (DCFR)	2	2	2	1	Don't care	Don't care
Quad command fast read (QCFR)	4	4	4	2	Don't care	Don't care

Table 15-2: Quad SPI Configuration for Micron N25Q128 Device (Write Instructions)

Instruction	Lanes Used By Opcode	Lanes Used to Send Address	Lanes Used to Send Data	instwidth Value	addrwidth Value	datawidth Value
Page program	1	1	1	0	0	0
Dual input fast program (DIFP)	1	1	2	0	0	1
Dual input extended fast program (DIEFP)	1	2	2	0	1	1
Quad input fast program (QIFP)	1	1	4	0	0	2
Quad input extended fast program (QIEFP)	1	4	4	0	2	2
Dual command fast program (DCFP)	2	2	2	1	Don’t care	Don’t care
Quad command fast program (QCFP)	4	4	4	2	Don’t care	Don’t care

XIP Mode

The quad SPI controller supports XIP mode, if the flash devices support XIP mode. Depending on the flash device, XIP mode puts the flash device in read-only mode, reducing command overhead.

The quad SPI controller must instruct the flash device to enter XIP mode by sending the mode bits. When the enter XIP mode on next read bit (`enterxipnextrd`) of the `cfg` register is set to 1, the quad SPI controller and the flash device are ready to enter XIP mode on the next read instruction. When the enter XIP mode immediately bit (`enterxipimm`) of the `cfg` register is set to 1, the quad SPI controller and flash device enter XIP mode immediately.

When the `enterxipnextrd` or `enterxipimm` bit of the `cfg` register is set to 0, the quad SPI controller and flash device exit XIP mode on the next read instruction. For more information, refer to the *“XIP Mode Operations”* section.

Related Information

XIP Mode Operations on page 15-17

Write Protection

You can program the controller to write protect a specific region of the flash device. The protected region is defined as a set of blocks, specified by a starting and ending block. Writing to an area of protected flash region memory generates an error and triggers an interrupt.

You define the block size by specifying the number of bytes per block through the number of bytes per block field (`bytespersubsector`) of the device size register (`devsz`). The lower write protection register (lowwrprot) specifies the first flash block in the protected region. The upper write protection register (uppwrprot) specifies the last flash block in the protected region.

The write protection enable bit (en) of the write protection register (wrprot) enables and disables write protection. The write protection inversion bit (inv) of the wrprot register flips the definition of protection so that the region specified by lowwrprt and uppwrprt is unprotected and all flash memory outside that region is protected.

Data Slave Sequential Access Detection

The quad SPI flash controller detects sequential accesses to the data slave interface by comparing the current access with the previous access. An access is sequential when it meets the following conditions:

- The address of the current access sequentially follows the address of the previous access.
- The direction of the current access (read or write) is the same as previous access.
- The size of the current access (byte, half-word, or word) is the same as previous access.

When the access is detected as nonsequential, the sequential access to the flash device is terminated and a new sequential access begins. Altera recommends accessing the data slave sequentially. Sequential access has less command overhead, and therefore, increases data throughput.

Clocks

There are two clock inputs to the quad SPI controller (l4_mp_clk and qspi_clk) and one clock output (sclk_out). The quad SPI flash controller uses the l4_mp_clk clock to clock the data slave transfers and register slave accesses. The qspi_clk clock is the reference clock for the quad SPI controller and is used to serialize the data and drive the external SPI interface. The sclk_out clock is the clock source for the connected flash devices.

The qspi_clk clock must be greater than two times the l4_mp_clk. The sclk_out clock is derived by dividing down the qspi_clk clock by the baud rate divisor field (bauddiv) of the cfg register.

Related Information

Clock Manager on page 2-1

Resets

A single reset signal (qspi_flash_rst_n) is provided as an input to the quad SPI controller. The reset manager drives the signal on a cold or warm reset.

Related Information

Reset Manager on page 3-1

Interrupts

All interrupt sources are combined to create a single level-sensitive, active-high interrupt (qspi_intr_in). Software can determine the source of the interrupt by reading the interrupt status register (irqstat). By default, the interrupt source is cleared when software writes a one (1) to the interrupt status register. The interrupts are individually maskable through the interrupt mask register (irqmask). Table 12–2 lists the interrupt sources in the irqstat register.

Table 15-3: Interrupt Sources in the irqstat Register

Interrupt Source	Description
Underflow detected	When 0, no underflow has been detected. When 1, the data slave write data is being supplied too slowly. This situation can occur when data slave write data is being supplied too slowly to keep up with the requested write operation This bit is reset only by a system reset and cleared only when a 1 is written to it.
Indirect operation complete	The controller has completed a triggered indirect operation.
Indirect read reject	An indirect operation was requested but could not be accepted because two indirect operations are already in the queue.
Protected area write attempt	A write to a protected area was attempted and rejected.
Illegal data slave access detected	An illegal data slave access has been detected. Data slave wrapping bursts and the use of split and retry accesses can cause this interrupt. It is usually an indicator that soft masters in the FPGA fabric are attempting to access the HPS in an unsupported way.
Transfer watermark reached	The indirect transfer watermark level has been reached.
Receive overflow	This condition occurs only in legacy SPI mode. When 0, no overflow has been detected. When 1, an over flow to the RX FIFO buffer has occurred. This bit is reset only by a system reset and cleared to zero only when this register is written to. If a new write to the RX FIFO buffer occurs at the same time as a register is read, this flag remains set to 1.
TX FIFO not full	This condition occurs only in legacy SPI mode. When 0, the TX FIFO buffer is full. When 1, the TX FIFO buffer is not full.
TX FIFO full	This condition occurs only in legacy SPI mode. When 0, the TX FIFO buffer is not full. When 1, the TX FIFO buffer is full.
RX FIFO not empty	This condition occurs only in legacy SPI mode. When 0, the RX FIFO buffer is empty. When 1, the RX FIFO buffer is not empty.

Interrupt Source	Description
RX FIFO full	This condition occurs only in legacy SPI mode. When 0, the RX FIFO buffer is not full. When 1, the RX FIFO buffer is full.
Indirect read partition overflow	Indirect Read Partition of SRAM is full and unable to immediately complete indirect operation

Interface Signals

The quad SPI controller provides four chip select outputs to allow control of up to four external quad SPI flash devices. The outputs serve different purposes depending on whether the device is used in single, dual, or quad operation mode. Table 12–3 lists the I/O pin use of the quad SPI controller interface signals for each operation mode.

Table 15-4: Interface Signals

Signal	Mode	Direction	Function
`data[0]`	Single	Output	Data output 0
	Dual or quad	Bidirectional	Data I/O 0
`data[1]`	Single	Input	Data input 0
	Dual or quad	Bidirectional	Data I/O 1
`data[2]`	Single or dual	Output	Active low write protect
	Quad	Bidirectional	Data I/O 2
`data[3]`	Single, dual, or quad	Bidirectional	Data I/O 3
`ss_n[0]`	Single, dual, or quad	Output	Active low slave select 0
`ss_n[1]`			Active low slave select 1
`ss_n[2]`			Active low slave select 2
`ss_n[3]`			Active low slave select 3
`sclk`			Serial clock

Quad SPI Flash Controller Programming Model

Setting Up the Quad SPI Flash Controller

The following steps describe how to set up the quad SPI controller:

1. Wait until any pending operation has completed.
2. Disable the quad SPI controller with the quad SPI enable field (`en`) of the `cfg` register.
3. Update the `instwidth` field of the `devrd` register with the instruction type you wish to use for indirect and direct writes and reads.
4. If mode bit enable bit (`enmodebits`) of the `devrd` register is enabled, update the mode bit register (`modebit`).
5. Update the `devsz` register as needed. Parts or all of this register might have been updated after initialization. The number of address bytes is a key configuration setting required for performing reads and writes. The number of bytes per page is required for performing any write. The number of bytes per device block is only required if the write protect feature is used.
6. Update the device delay register (`delay`). This register allows the user to adjust how the chip select is driven after each flash access. Each device may have different timing requirements. If the serial clock frequency is increased, these timing requirements become more critical. The numbers specified in this register are based on the period of the `qspi_clk` clock. For example, an some devices need 50 ns minimum time before the slave select can be reasserted after it has been deasserted. When the device is operating at 100 MHz, the clock period is 10 ns, so 40 ns extra is required. If the `qspi_clk` clock is running at 400 MHz (2.5 ns period), specify a value of at least 16 to the clock delay for chip select deassert field (`nss`) of the `delay` register.
7. Update the `remapaddr` register as needed. This register only affects direct access mode.
8. Set up and enable the write protection registers (`wrprot`, `lowwrprot`, and `uppwrprot`), when write protection is required.
9. Enable required interrupts though the `irqmask` register.
10. Set up the `bauddiv` field of the `cfg` register to define the required clock frequency of the target device.
11. Update the read data capture register (`rddatacap`) as needed. This register delays when the read data is captured and can help when the read data path from the device to the quad SPI controller is long and the device clock frequency is high.
12. Enable the quad SPI controller with the `en` field of the `cfg` register.

Indirect Read Operation with DMA Disabled

The following steps describe the general software flow to set up the quad SPI controller for indirect read operation with the DMA disabled:

1. Perform the steps described in the *"Setting Up the Quad SPI Flash Controller"* section.
2. Set the flash memory start address in the `indrdstaddr` register.
3. Set the number of bytes to be transferred in the `indrdcnt` register.
4. Set the indirect transfer trigger address in the `indaddrtrig` register.
5. Set up the required interrupts through the `irqmask` register.
6. If the watermark level is used, set the SRAM watermark level through the `indrdwater` register.
7. Start the indirect read operation by setting the `start` field of the `indrd` register to 1.

8. Either use the watermark level interrupt or poll the SRAM fill level in the `sramfill` register to determine when there is sufficient data in the SRAM.
9. Issue a read transaction to the indirect address to access the SRAM. Repeat **step 8** if more read transactions are needed to complete the indirect read transfer.
10. Either use the indirect complete interrupt to determine when the indirect read operation has completed or poll the completion status of the indirect read operation through the indirect completion status bit (`ind_ops_done_status`) of the `indrd` register.

Related Information

Setting Up the Quad SPI Flash Controller on page 15-15

Indirect Read Operation with DMA Enabled

The following steps describe the general software flow to set up the quad SPI controller for indirect read operation with the DMA enabled:

1. Perform the steps described in the *"Setting Up the Quad SPI Flash Controller"* section.
2. Set the flash memory start address in the `indrdstaddr` register.
3. Set the number of bytes to be transferred in the `indrdcnt` register.
4. Set the indirect transfer trigger address in the `indaddrtrig` register.
5. Set the number of bytes for single and burst type DMA transfers in the `dmaper` register.
6. Optionally set the SRAM watermark level in the `indrdwater` register to control the rate DMA requests are issued.
7. Start an indirect read access by setting the `start` field of the `indrd` register to 1.
8. Either use the indirect complete interrupt to determine when the indirect read operation has completed or poll the completion status of the indirect read operation through the `ind_ops_done_status` field of the `indrd` register.

Related Information

Setting Up the Quad SPI Flash Controller on page 15-15

Indirect Write Operation with DMA Disabled

The following steps describe the general software flow to set up the quad SPI controller for indirect write operation with the DMA disabled:

1. Perform the steps described in the *"Setting Up the Quad SPI Flash Controller"* section.
2. Set the flash memory start address in the `indwrstaddr` register.
3. Set up the number of bytes to be transferred in the `indwrcnt` register.
4. Set the indirect transfer trigger address in the `indaddrtrig` register.
5. Set up the required interrupts through the interrupt mask register (`irqmask`).
6. Optionally set the SRAM watermark level in the `indwrwater` register to control the rate DMA requests are issued. The value set must be greater than one flash page. For more information, refer to the *"Indirect Write Operation"* section.

7. Start the indirect write operation by setting the `start` field of the `indwr` register to 1.
8. Either use the watermark level interrupt or poll the SRAM fill level in the `sramfill` register to determine when there is sufficient space in the SRAM.
9. Issue a write transaction to the indirect address to write one flash page of data to the SRAM. Repeat **step 8** if more write transactions are needed to complete the indirect write transfer. The final write may be less than one page of data.

Related Information

- **Indirect Write Operation** on page 15-6
- **Setting Up the Quad SPI Flash Controller** on page 15-15

Indirect Write Operation with DMA Enabled

The following steps describe the general software flow to set up the quad SPI controller for indirect write operation with the DMA enabled:

1. Perform the steps described in the *"Setting Up the Quad SPI Flash Controller"* section.
2. Set the flash memory start address in the `indwrstaddr` register.
3. Set the number of bytes to be transferred in the `indcnt` field of the `indwr` register.
4. Set the indirect transfer trigger address in the `indaddrtrig` register.
5. Set the number of bytes for single and burst type DMA transfers in the `dmaper` register.
6. Optionally set the SRAM watermark level in the `indwrwater` register to control the rate DMA requests are issued. The value set must be greater than one flash page. For more information, refer to the *"Indirect Write Operation"* section.
7. Start the indirect write access by setting the `start` field of the `indirwr` register to 1.
8. Either use the indirect complete interrupt to determine when the indirect write operation has completed or poll the completion status of the indirect write operation through the `ind_ops_done_status` field of the `indwr` register.

Related Information

- **Indirect Write Operation** on page 15-6
- **Setting Up the Quad SPI Flash Controller** on page 15-15

XIP Mode Operations

XIP mode is supported in most SPI flash devices. However, flash device manufacturers do not use a consistent standard approach. Most use signature bits that are sent to the device immediately following the address bytes. Some devices use signature bits and also require a flash device configuration register write to enable XIP mode.

Entering XIP Mode

Micron Quad SPI Flash Devices with Support for Basic-XIP

To enter XIP mode in a Micron quad SPI flash device with support for Basic-XIP, perform the following steps:

1. Save the values in the mode bits, if you intend to restore them upon exit.
2. Disable the direct access controller and indirect access controller to ensure no new read or write accesses are sent to the flash device.
3. Set the XIP mode bits in the `modebit` register to 0x80.
4. Enable the quad SPI controller's XIP mode by setting the `enterxipnextrd` bit of the `cfg` register to 1.
5. Re-enable the direct access controller and, if required, the indirect access controller.

Micron Quad SPI Flash Devices without Support for Basic-XIP

To enter XIP mode in a Micron quad SPI flash device without support for Basic-XIP, perform the following steps:

1. Save the values in the mode bits, if you intend to restore them upon exit.
2. Disable the direct access controller and indirect access controller to ensure no new read or write accesses will be sent to the flash device.
3. Ensure XIP mode is enabled in the flash device by setting the volatile configuration register (VCR) bit 3 to 1. Use the `flashcmd` register to issue the VCR write command.
4. Set the XIP mode bits in the `modebit` register to 0x00.
5. Enable the quad SPI controller’s XIP mode by setting the `enterxipnextrd` bit of the `cfg` register to 1.
6. Re-enable the direct access controller and, if required, the indirect access controller.

Winbond Quad SPI Flash Devices

To enter XIP mode in a Winbond quad SPI flash device, perform the following steps:

1. Save the values in the mode bits, if you intend to restore them upon exit.
2. Disable the direct access controller and indirect access controller to ensure no new read or write accesses are sent to the flash device.
3. Set the XIP mode bits in the `modebit` register to 0x20.
4. Enable the quad SPI controller’s XIP mode by setting the `enterxipnextrd` bit of the `cfg` register to 1.
5. Re-enable the direct access controller and, if required, the indirect access controller.

Spansion Quad SPI Flash Devices

To enter XIP mode a Spansion quad SPI flash device, perform the following steps:

1. Save the values in the mode bits, if you intend to restore them upon exit.
2. Disable the direct access controller and indirect access controller to ensure no new read or write accesses are sent to the flash device.
3. Set the XIP mode bits in the `modebit` register to 0xA0.
4. Enable the quad SPI controller’s XIP mode by setting the `enterxipnextrd` bit of the `cfg` register to 1.
5. Re-enable the direct access controller and, if required, the indirect access controller.

Exiting XIP Mode

To exit XIP mode, perform the following steps:

1. Disable the direct access controller and indirect access controller to ensure no new read or write accesses are sent to the flash device.
2. Restore the mode bits to the values before entering XIP mode, depending on the flash device and manufacturer.
3. Set the `enterxipnextrd` bit of the `cfg` register to 0.

The flash device must receive a read instruction before it can disable its internal XIP mode state. Thus, XIP mode internally stays active until the next read instruction is serviced. Ensure that XIP mode is disabled before the end of any read sequence.

XIP Mode at Power on Reset

Some flash devices can be XIP-enabled as a nonvolatile configuration setting, allowing the flash device to enter XIP mode at power-on reset (POR) without software intervention. Software cannot discover the XIP state at POR through flash status register reads because an XIP-enabled flash device can only be accessed through the XIP read operation. If you known the device will enter XIP mode at POR, have your initial boot software configure the `modebit` register and set the `enterxipimm` bit of the `cfg` register to 1.

If you do not know in advance whether or not the device will enter XIP mode at POR, have your initial boot software issue an XIP mode exit command through the `flashcmd` register, then follow the steps in the *"Entering XIP Mode"* section. Software must be aware of the mode bit requirements of the device, because XIP mode entry and exit varies by device.

Related Information

Entering XIP Mode on page 15-18

FIFO RAM Initialization Requirement with ECC Enabled

Due to the Quad SPI controller FIFO implementation, a spurious ECC interrupt always happens when ECC is enabled for the first time. This happens because:

- The FIFO always reads from RAM when the write operation is not present
- The ECC syndrome bits generated when ECC is enabled only; and then remain uninitialized when ECC is disabled

Therefore, it reads the uninitialized content or syndrome bits when ECC is enabled for first time and generates the interrupt.

Note: If ECC is disabled, initialization is not required. However, if ECC is enabled after FIFO has been used for read or write operation, initialization must be done to initialize the ECC syndrome bits.

Clearing Spurious Interrupt

To clean this spurious interrupt, it is recommended for the driver or the system software to initialize the FIFO RAM and clear the interrupt for ECC usage.

1. Configure the SRAM Partition Register to maximize the FIFO RAM for read transfer.

The FIFO RAM is 512 bytes in size, which is 128 words for QSPI RAM. The recommended maximum read partition size is 126 words. This is the limitation support by SRAM Fill Level Register, which will leave 2 words for write partition.

2. Enable the ECC. This will trigger the spurious ECC interrupt.
3. Trigger an indirect read transfer with transfer size that will fill up the read partition, which is 126 words.
   Do not start the AHB read, this will ensure all read partition in FIFO RAM will be initialized. Monitor the SRAM Fill Level Register for the status.
4. Now start AHB read and start indirect write operation to write back to the same device location, this will fill up and initialized the write partition RAM.
5. To clear the spurious interrupt, the QSPI controller has to be reset then the QSPI ECC interrupt bit in system manager has to be cleared.

Quad SPI Flash Controller Address Map and Register Definitions

The address map and register definitions for the Quad SPI Flash Controller consist of the following regions:

- QSPI Flash Controller Module Registers
- QSPI Flash Module Data

Related Information

Introduction to Cyclone V Hard Processor System on page 1-1

QSPI Flash Module Data (AHB Slave) Address Map

This address space is allocated for QSPI direct, indirect, and SPI legacy mode accesses. For more information, please refer to the *Quad SPI Flash Controller* chapter in the *Hard Processor System Technical Reference Manual*.

Table 15-5: QSPI Flash Module Data Space Address Range

Module Instance	Start Address	End Address
QSPI_DATA	0xFFA00000	0xFFAFFFFF

QSPI Flash Controller Module Registers Address Map

Registers in the QSPI Flash Controller module accessible via its APB slave

Base Address: 0xFF705000

QSPI Flash Controller Module Registers

Register	Offset	Width	Acces s	Reset Value	Description
cfg on page 15-22	0x0	32	RW	0x780000	QSPI Configuration Register
devrd on page 15-27	0x4	32	RW	0x3	Device Read Instruction Register
devwr on page 15-29	0x8	32	RW	0x2	Device Write Instruction Register

Register	Offset	Width	Acces s	Reset Value	Description
delay on page 15-30	0xC	32	RW	0x0	QSPI Device Delay Register
rddatacap on page 15-31	0x10	32	RW	0x1	Read Data Capture Register
devsz on page 15-32	0x14	32	RW	0x101002	Device Size Register
srampart on page 15-33	0x18	32	RW	0x40	SRAM Partition Register
indaddrtrig on page 15-33	0x1C	32	RW	0x0	Indirect AHB Address Trigger Register
dmaper on page 15-34	0x20	32	RW	0x0	DMA Peripheral Register
remapaddr on page 15-35	0x24	32	RW	0x0	Remap Address Register
modebit on page 15-35	0x28	32	RW	0x0	Mode Bit Register
sramfill on page 15-36	0x2C	32	RO	0x0	SRAM Fill Register
txthresh on page 15-36	0x30	32	RW	0x1	TX Threshold Register
rxthresh on page 15-37	0x34	32	RW	0x1	RX Threshold Register
irqstat on page 15-37	0x40	32	RW	0x100	Interrupt Status Register
irqmask on page 15-40	0x44	32	RW	0x0	Interrupt Mask
lowwrprot on page 15-42	0x50	32	RW	0x0	Lower Write Protection Register
uppwrprot on page 15-43	0x54	32	RW	0x0	Upper Write Protection Register
wrprot on page 15-44	0x58	32	RW	0x0	Write Protection Register
indrd on page 15-45	0x60	32	RW	0x0	Indirect Read Transfer Register
indrdwater on page 15-47	0x64	32	RW	0x0	Indirect Read Transfer Watermark Register
indrdstaddr on page 15-47	0x68	32	RW	0x0	Indirect Read Transfer Start Address Register
indrdcnt on page 15-48	0x6C	32	RW	0x0	Indirect Read Transfer Number Bytes Register
indwr on page 15-48	0x70	32	RW	0x0	Indirect Write Transfer Register
indwrwater on page 15-50	0x74	32	RW	0xFFFFFFFF	Indirect Write Transfer Watermark Register
indwrstaddr on page 15-50	0x78	32	RW	0x0	Indirect Write Transfer Start Address Register
indwrcnt on page 15-51	0x7C	32	RW	0x0	Indirect Write Transfer Count Register
flashcmd on page 15-51	0x90	32	RW	0x0	Flash Command Register

Register	Offset	Width	Access	Reset Value	Description
flashcmdaddr on page 15-55	0x94	32	RW	0x0	Flash Command Address Registers
flashcmdrddatalo on page 15-55	0xA0	32	RW	0x0	Flash Command Read Data Register (Lower)
flashcmdrddataup on page 15-56	0xA4	32	RW	0x0	Flash Command Read Data Register (Upper)
flashcmdwrdatalo on page 15-57	0xA8	32	RW	0x0	Flash Command Write Data Register (Lower)
flashcmdwrdataup on page 15-57	0xAC	32	RW	0x0	Flash Command Write Data Register (Upper)
moduleid on page 15-58	0xFC	32	RO	0x1001	Module ID Register

cfg

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
idle RO 0x0	Reserved								bauddiv RW 0xF				enterxipimm RW 0x0	enterxipnextrd RW 0x0	enahbremap RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
endma RW 0x0	wp RW 0x0	percslines RW 0x0				perseldec RW 0x0	enlegacyip RW 0x0	endiracc RW 0x0	Reserved				selclkphase RW 0x0	selclkpol RW 0x0	en RW 0x0

cfg Fields

Bit	Name	Description	Access	Reset
31	idle	This is a STATUS read-only bit. Note this is a retimed signal, so there will be some inherent delay on the generation of this status signal. **Value** **Description** 0x1 Idle Mode 0x0 Non-Idle Mode	RO	0x0
22:19	bauddiv	SPI baud rate = ref_clk / (2 * baud_rate_divisor) **Value** **Description** 0x0 Baud Rate Div/2 0x1 Baud Rate Div/4 0x2 Baud Rate Div/6 0x3 Baud Rate Div/8 0x4 Baud Rate Div/10 0x5 Baud Rate Div/12 0x6 Baud Rate Div/14 0x7 Baud Rate Div/16 0x8 Baud Rate Div/18 0x9 Baud Rate Div/20 0xa Baud Rate Div/22 0xb Baud Rate Div/24 0xc Baud Rate Div/26 0xd Baud Rate Div/28 0xe Baud Rate Div/30 0xf Baud Rate Div/32	RW	0xF

Bit	Name	Description	Access	Reset
18	`enterxipimm`	If XIP is enabled, then setting to disabled will cause the controller to exit XIP mode on the next READ instruction. If XIP is disabled, then setting enable will operate the device in XIP mode immediately. Use this register when the external device wakes up in XIP mode (as per the contents of its non- volatile configuration register). The controller will assume the next READ instruction will be passed to the device as an XIP instruction, and therefore will not require the READ opcode to be transferred. Note: To exit XIP mode, this bit should be set to 0. This will take effect in the attached device only after the next READ instruction is executed. Software therefore should ensure that at least one READ instruction is requested after resetting this bit in order to be sure that XIP mode is exited. **Value** **Description** 0x1 Enter XIP Mode immediately 0x0 Exit XIP Mode on next READ instruction	`RW`	`0x0`
17	`enterxipnextrd`	If XIP is enabled, then setting to disabled will cause the controller to exit XIP mode on the next READ instruction. If XIP is disabled, then setting to enabled will inform the controller that the device is ready to enter XIP on the next READ instruction. The controller will therefore send the appropriate command sequence, including mode bits to cause the device to enter XIP mode. Use this register after the controller has ensured the FLASH device has been configured to be ready to enter XIP mode. Note : To exit XIP mode, this bit should be set to 0. This will take effect in the attached device only AFTER the next READ instruction is executed. Software should therefore ensure that at least one READ instruction is requested after resetting this bit before it can be sure XIP mode in the device is exited. **Value** **Description** 0x1 Enter XIP Mode on next READ instruction 0x0 Exit XIP Mode on next READ instruction	`RW`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>16</td><td>enahbremap</td><td>(Direct Access Mode Only) When enabled, the incoming AHB address will be adapted and sent to the FLASH device as (address + N), where N is the value stored in the remap address register.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x1</td><td>Enable AHB Re-mapping</td></tr><tr><td>0x0</td><td>Disable AHB Re-mapping</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>15</td><td>endma</td><td>Allows DMA handshaking mode. When enabled the QSPI will trigger DMA transfer requests via the DMA peripheral interface.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x1</td><td>Enable DMA Mode</td></tr><tr><td>0x0</td><td>Disable DMA Mode</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>14</td><td>wp</td><td>This bit controls the write protect pin of the flash devices. The signal qspi_mo2_wpn needs to be resynchronized to the generated memory clock as necessary.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x1</td><td>Enable Write Protect</td></tr><tr><td>0x0</td><td>Disable Write Protect</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>13:10</td><td>percslines</td><td>Peripheral chip select line output decode type. As per perseldec, if perseldec = 0, the decode is select 1 of 4 decoding on signals, qspi_n_ss_out[3:0], The asserted decode line goes to 0. If perseldec = 1, the signals qspi_n_ss_out[3:0] require an external 4 to 16 decoder.</td><td>RW</td><td>0x0</td></tr>
<tr><td>9</td><td>perseldec</td><td>Select between '1 of 4 selects' or 'external 4-to-16 decode'. The qspi_n_ss_out[3:0] output signals are controlled.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x1</td><td>Select external 4-to-16 decode</td></tr><tr><td>0x0</td><td>Selects 1 of 4 qspi_n_ss_out[3:0]</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
8	`enlegacyip`	This bit can select the Direct Access Controller/ Indirect Access Controller or legacy mode.If legacy mode is selected, any write to the controller via the AHB interface is serialized and sent to the FLASH device. Any valid AHB read will pop the internal RX-FIFO, retrieving data that was forwarded by the external FLASH device on the SPI lines, byte transfers of 4, 2 or 1 are permitted and controlled via the HSIZE input. **Value** — **Description** 0x1 — Legacy Mode 0x0 — Use Direct/Indirect Access Controller	RW	0x0
7	`endiracc`	If disabled, the Direct Access Controller becomes inactive once the current transfer of the data word (FF_W) is complete. When the Direct Access Controller and Indirect Access Controller are both disabled, all AHB requests are completed with an error response. **Value** — **Description** 0x0 — Disable Direct Access Ctrl 0x1 — Enable Direct Access Ctrl	RW	0x0
2	`selclkphase`	Selects whether the clock is in an active or inactive phase outside the SPI word. **Value** — **Description** 0x0 — SPI clock is quiescent low 0x1 — Clock Inactive	RW	0x0
1	`selclkpol`	Controls spiclk modes of operation. **Value** — **Description** 0x1 — SPI clock is quiescent low 0x0 — SPI clock is quiescent high	RW	0x0

Bit	Name	Description	Access	Reset
0	en	If this bit is disabled, the QSPI will finish the current transfer of the data word (FF_W) and stop sending. When Enabled, and qspi_n_mo_en = 0, all output enables are inactive and all pins are set to input mode. **Value** **Description** 0x0 Disable the QSPI 0x1 Enable the QSPI	RW	0x0

devrd

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705004

Offset: 0x4

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="5">dummyrdclks
RW 0x0</td><td colspan="3">Reserved</td><td>enmodebits
RW 0x0</td><td colspan="2">Reserved</td><td colspan="2">datawidth
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">addrwidth
RW 0x0</td><td colspan="2">Reserved</td><td colspan="2">instwidth
RW 0x0</td><td colspan="8">rdopcode
RW 0x3</td></tr>
</table>

devrd Fields

Bit	Name	Description	Access	Reset
28:24	dummyrdclks	Number of dummy clock cycles required by device for read instruction.	RW	0x0

Bit	Name	Description	Access	Reset
20	`enmodebits`	If this bit is set, the mode bits as defined in the Mode Bit Configuration register are sent following the address bytes. **Value** — **Description** 0x0 — No Order 0x1 — Mode Bits follow address bytes	RW	0x0
17:16	`datawidth`	Sets read data transfer width (1, 2, or 4 bits). **Value** — **Description** 0x0 — Read data transferred on DQ0. Supported by all SPI flash devices 0x1 — Read data transferred on DQ0 and DQ1. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Dual SP (DIO-SPI) Protocol. 0x2 — Read data transferred on DQ0, DQ1, DQ2, and DQ3. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Quad SP (QIO-SPI) Protocol.	RW	0x0
13:12	`addrwidth`	Sets read address transfer width (1, 2, or 4 bits). **Value** — **Description** 0x0 — Read address transferred on DQ0. Supported by all SPI flash devices 0x1 — Read address transferred on DQ0 and DQ1. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Dual SP (DIO-SPI) Protocol. 0x2 — Read address transferred on DQ0, DQ1, DQ2, and DQ3. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Quad SP (QIO-SPI) Protocol.	RW	0x0

Bit	Name	Description	Access	Reset
9:8	`instwidth`	Sets instruction transfer width (1, 2, or 4 bits). Applies to all instructions sent to SPI flash device (not just read instructions). **Value** **Description** 0x0 Instruction transferred on DQ0. Supported by all SPI flash devices. 0x1 Instruction transferred on DQ0 and DQ1. Supported by all SPI flash devices that support the Dual SP (DIO-SPI) Protocol. 0x2 Instruction transferred on DQ0, DQ1, DQ2, and DQ3. Supported by all SPI flash devices that support the Quad SP (QIO-SPI) Protocol.	RW	0x0
7:0	`rdopcode`	Read Opcode to use when not in XIP mode **Value** **Description** 0x3 Read Opcode in Non-XIP mode 0xb Fast Read in Non-XIP mode	RW	0x3

devwr

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF705008`

Offset: `0x8`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="5">dummywrclks
RW 0x0</td><td colspan="6">Reserved</td><td colspan="2">datawidth
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">addrwidth
RW 0x0</td><td colspan="4">Reserved</td><td colspan="8">wropcode
RW 0x2</td></tr>
</table>

devwr Fields

Bit	Name	Description	Access	Reset
28:24	dummywrclks	Number of dummy clock cycles required by device for write instruction.	RW	0x0
17:16	datawidth	Sets write data transfer width (1, 2, or 4 bits). **Value** **Description** 0x0 Write data transferred on DQ0. Supported by all SPI flash devices 0x1 Read data transferred on DQ0 and DQ1. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Dual SP (DIO-SPI) Protocol. 0x2 Read data transferred on DQ0, DQ1, DQ2, and DQ3. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Quad SP (QIO-SPI) Protocol.	RW	0x0
13:12	addrwidth	Sets write address transfer width (1, 2, or 4 bits). **Value** **Description** 0x0 Write address transferred on DQ0. Supported by all SPI flash devices 0x1 Read address transferred on DQ0 and DQ1. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Dual SP (DIO-SPI) Protocol. 0x2 Read address transferred on DQ0, DQ1, DQ2, and DQ3. Supported by some SPI flash devices that support the Extended SPI Protocol and by all SPI flash devices that support the Quad SP (QIO-SPI) Protocol.	RW	0x0
7:0	wropcode	Write Opcode	RW	0x2

delay

This register is used to introduce relative delays into the generation of the master output signals. All timings are defined in cycles of the qspi_clk.

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF70500C

Offset: 0xC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
nss RW 0x0								btwn RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
after RW 0x0								init RW 0x0							

delay Fields

Bit	Name	Description	Access	Reset
31:24	nss	Delay in master reference clocks for the length that the master mode chip select outputs are de-asserted between transactions. The minimum delay is always qspi_sck_out period to ensure the chip select is never re-asserted within an qspi_sck_out period.	RW	0x0
23:16	btwn	Delay in master reference clocks between one chip select being de-activated and the activation of another. This is used to ensure a quiet period between the selection of two different slaves and requires the transmit FIFO to be empty.	RW	0x0
15:8	after	Delay in master reference clocks between last bit of current transaction and deasserting the device chip select (qspi_n_ss_out). By default, the chip select will be deasserted on the cycle following the completion of the current transaction.	RW	0x0
7:0	init	Delay in master reference clocks between setting qspi_n_ss_out low and first bit transfer.	RW	0x0

rddatacap

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705010

Offset: 0x10

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved											delay RW 0x0				byp RW 0x1

rddatacap Fields

Bit	Name	Description	Access	Reset
4:1	delay	Delay the read data capturing logic by the programmed number of qspi_clk cycles	RW	0x0
0	byp	Controls bypass of the adapted loopback clock circuit **Value** **Description** 0x0 No Bypass 0x1 Bypass loopback clock circuit	RW	0x1

devsz

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705014

Offset: 0x14

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											bytespersubsector RW 0x10				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
bytesperdevicepage RW 0x100												numaddrbytes RW 0x2			

devsz Fields

Bit	Name	Description	Access	Reset
20:16	bytespersubsector	Number of bytes per Block. This is required by the controller for performing the write protection logic. The number of bytes per block must be a power of 2 number.	RW	0x10

Bit	Name	Description	Access	Reset
15:4	`bytesperdevicepage`	Number of bytes per device page. This is required by the controller for performing FLASH writes up to and across page boundaries.	`RW`	`0x100`
3:0	`numaddrbytes`	Number of address bytes. A value of 0 indicates 1 byte.	`RW`	`0x2`

srampart

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF705018`

Offset: `0x18`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									addr RW 0x40						

srampart Fields

Bit	Name	Description	Access	Reset
6:0	`addr`	Defines the size of the indirect read partition in the SRAM, in units of SRAM locations. By default, half of the SRAM is reserved for indirect read operations and half for indirect write operations.	`RW`	`0x40`

indaddrtrig

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF70501C`

Offset: `0x1C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr RW 0x0															

indaddrtrig Fields

Bit	Name	Description	Access	Reset
31:0	addr	This is the base address that will be used by the AHB controller. When the incoming AHB read access address matches a range of addresses from this trigger address to the trigger address + 15, then the AHB request will be completed by fetching data from the Indirect Controllers SRAM.	RW	0x0

dmaper

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705020

Offset: 0x20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				numburstreqbytes RW 0x0				Reserved				numsglreqbytes RW 0x0			

dmaper Fields

Bit	Name	Description	Access	Reset
11:8	numburstreqbytes	Number of bytes in a burst type request on the DMA peripheral request. A programmed value of 0 represents a single byte. This should be setup before starting the indirect read or write operation. The actual number of bytes used is 2**(value in this register) which will simplify implementation.	RW	0x0

Bit	Name	Description	Access	Reset
3:0	numsglreqbytes	Number of bytes in a single type request on the DMA peripheral request. A programmed value of 0 represents a single byte. This should be setup before starting the indirect read or write operation. The actual number of bytes used is 2**(value in this register) which will simplify implementation.	RW	0x0

remapaddr

This register is used to remap an incoming AHB address to a different address used by the FLASH device.

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705024

Offset: 0x24

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">value
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">value
RW 0x0</td></tr>
</table>

remapaddr Fields

Bit	Name	Description	Access	Reset
31:0	value	This offset is added to the incoming AHB address to determine the address used by the FLASH device.	RW	0x0

modebit

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705028

Offset: 0x28

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">mode
RW 0x0</td></tr>
</table>

modebit Fields

Bit	Name	Description	Access	Reset
7:0	mode	These are the 8 mode bits that are sent to the device following the address bytes if mode bit transmission has been enabled.	RW	0x0

sramfill

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF70502C

Offset: 0x2C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
indwrpart RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
indrdpart RO 0x0															

sramfill Fields

Bit	Name	Description	Access	Reset
31:16	indwrpart		RO	0x0
15:0	indrdpart		RO	0x0

txthresh

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705030

Offset: 0x30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x1			

txthresh Fields

Bit	Name	Description	Access	Reset
3:0	level	Defines the level at which the transmit FIFO not full interrupt is generated	RW	0x1

rxthresh

Device Instruction Register

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705034

Offset: 0x34

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												level RW 0x1			

rxthresh Fields

Bit	Name	Description	Access	Reset
3:0	level	Defines the level at which the receive FIFO not empty interrupt is generated	RW	0x1

irqstat

The status fields in this register are set when the described event occurs and the interrupt is enabled in the mask register. When any of these bit fields are set, the interrupt output is asserted high. The fields are each cleared by writing a 1 to the field. Note that bit fields 7 thru 11 are only valid when legacy SPI mode is active.

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705040

Offset: 0x40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			indsramfull RW 0x0	rxfull RW 0x0	rxthreshcmp RW 0x0	txfull RW 0x0	txthreshcmp RW 0x1	rxover RW 0x0	indxfrlvl RW 0x0	illegalacc RW 0x0	protwrattempt RW 0x0	indrdreject RW 0x0	indopdone RW 0x0	underflowdet RW 0x0	Reserved

irqstat Fields

Bit	Name	Description	Access	Reset
12	indsramfull	Indirect Read Partition of SRAM is full and unable to immediately complete indirect operation **Value** **Description** 0x1 SRAM is full 0x0 SRAM is not full	RW	0x0
11	rxfull	Indicates that the receive FIFO is full or not. Only relevant in SPI legacy mode. **Value** **Description** 0x0 Receive FIFO Not Full 0x1 Receive FIFO Full	RW	0x0
10	rxthreshcmp	Indicates the number of entries in the receive FIFO with respect to the threshold specified in the RXTHRESH register. Only relevant in SPI legacy mode. **Value** **Description** 0x0 FIFO has <= RXTHRESH entries 0x1 FIFO has > RXTHRESH entries	RW	0x0
9	txfull	Indicates that the transmit FIFO is full or not. Only relevant in SPI legacy mode. **Value** **Description** 0x0 Transmit FIFO Not Full 0x1 Transmit FIFO Full	RW	0x0

Bit	Name	Description	Access	Reset
8	`txthreshcmp`	Indicates the number of entries in the transmit FIFO with respect to the threshold specified in the TXTHRESH register. Only relevant in SPI legacy mode. **Value** — **Description** 0x0 — FIFO has > TXTHRESH entries 0x1 — FIFO has <= TXTHRESH entries	RW	0x1
7	`rxover`	This should only occur in Legacy SPI mode. Set if an attempt is made to push the RX FIFO when it is full. This bit is reset only by a system reset and cleared only when this register is read. If a new push to the RX FIFO occurs coincident with a register read this flag will remain set. 0 : no overflow has been detected. 1 : an overflow has occurred. **Value** — **Description** 0x1 — Receive Overflow 0x0 — No Receive Overflow	RW	0x0
6	`indxfrlvl`	Indirect Transfer Watermark Level Reached **Value** — **Description** 0x1 — Water level reached 0x0 — No water level reached	RW	0x0
5	`illegalacc`	Illegal AHB access has been detected. AHB wrapping bursts and the use of SPLIT/RETRY accesses will cause this error interrupt to trigger. **Value** — **Description** 0x1 — Illegal AHB attempt 0x0 — No Illegal AHB attempt	RW	0x0
4	`protwrattempt`	Write to protected area was attempted and rejected. **Value** — **Description** 0x1 — Write Attempt to protected area 0x0 — No Write Attempt	RW	0x0

Bit	Name	Description	Access	Reset
3	`indrdreject`	Indirect operation was requested but could not be accepted. Two indirect operations already in storage. **Value** / **Description** 0x1 Indirect Operation Requested 0x0 No Indirect Operation	RW	0x0
2	`indopdone`	Controller has completed last triggered indirect operation **Value** / **Description** 0x1 Completed Indirect Operation 0x0 No Indirect Operation	RW	0x0
1	`underflowdet`	An underflow is detected when an attempt to transfer data is made when the transmit FIFO is empty. This may occur when the AHB write data is being supplied too slowly to keep up with the requested write operation. This bit is reset only by a system reset and cleared only when the register is read. **Value** / **Description** 0x1 Underflow 0x0 No Underflow	RW	0x0

irqmask

If disabled, the interrupt for the corresponding interrupt status register bit is disabled. If enabled, the interrupt for the corresponding interrupt status register bit is enabled.

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF705044`

Offset: `0x44`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			indsramfull RW 0x0	rxfull RW 0x0	rxthreshcmp RW 0x0	txfull RW 0x0	txthreshcmp RW 0x0	rxover RW 0x0	indxfrlvl RW 0x0	illegalacc RW 0x0	protwrattempt RW 0x0	indrdreject RW 0x0	indopdone RW 0x0	underflowdet RW 0x0	Reserved

irqmask Fields

Bit	Name	Description	Access	Reset
12	indsramfull	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
11	rxfull	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
10	rxthreshcmp	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
9	txfull	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
8	txthreshcmp	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
7	rxover	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
6	indxfrlvl	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
5	illegalacc	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
4	protwrattempt	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
3	indrdreject	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
2	indopdone	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0
1	underflowdet	**Value** **Description** 0x0 Disable Interrupt by Masking 0x1 Enable Interrupt	RW	0x0

lowwrprot

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
subsector RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
subsector RW 0x0															

lowwrprot Fields

Bit	Name	Description	Access	Reset
31:0	subsector	The block number that defines the lower block in the range of blocks that is to be locked from writing. The definition of a block in terms of number of bytes is programmable via the Device Size Configuration register.	RW	0x0

uppwrprot

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705054

Offset: 0x54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
subsector RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
subsector RW 0x0															

uppwrprot Fields

Bit	Name	Description	Access	Reset
31:0	subsector	The block number that defines the upper block in the range of blocks that is to be locked from writing. The definition of a block in terms of number of bytes is programmable via the Device Size Configuration register.	RW	0x0

wrprot

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705058

Offset: 0x58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														en RW 0x0	inv RW 0x0

wrprot Fields

Bit	Name	Description	Access	Reset
1	en	When enabled, any AHB write access with an address within the protection region defined in the lower and upper write protection registers is rejected. An AHB error response is generated and an interrupt source triggered. When disabled, the protection region is disabled. **Value** / **Description** 0x1 AHB Write Access rejected 0x0 Protection Region Disabled	RW	0x0

Bit	Name	Description	Access	Reset
0	inv	When enabled, the protection region defined in the lower and upper write protection registers is inverted meaning it is the region that the system is permitted to write to. When disabled, the protection region defined in the lower and upper write protection registers is the region that the system is not permitted to write to. **Value** — **Description** 0x1 — Write Region allowed 0x0 — Write Region not allowed	RW	0x0

indrd

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705060

Offset: 0x60

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								num_ind_ops_done RO 0x0		ind_ops_done_status RW 0x0	rd_queued RO 0x0	sram_full RW 0x0	rd_status RO 0x0	cancel RW 0x0	start RW 0x0

indrd Fields

Bit	Name	Description	Access	Reset
7:6	num_ind_ops_done	This field contains the number of indirect operations which have been completed. This is used in conjunction with the indirect completion status field (bit 5).	RO	0x0

Bit	Name	Description	Access	Reset
5	`ind_ops_done_status`	This field is set to 1 when an indirect operation has completed. Write a 1 to this field to clear it. **Value** **Description** 0x1 Indirect Op Complete operation 0x0 Indirect Op Not Complete	RW	0x0
4	`rd_queued`	Two indirect read operations have been queued **Value** **Description** 0x1 Queued Indirect Read 0x0 No Queued Read	RO	0x0
3	`sram_full`	SRAM full and unable to immediately complete an indirect operation. Write a 1 to this field to clear it. ; indirect operation (status) **Value** **Description** 0x1 Sram Full- Cant complete operation 0x0 SRram Not Full	RW	0x0
2	`rd_status`	Indirect read operation in progress (status) **Value** **Description** 0x1 Read Operation in progress 0x0 No read operation in progress	RO	0x0
1	`cancel`	This bit will cancel all ongoing indirect read operations. **Value** **Description** 0x1 Cancel Indirect Read 0x0 Do Not Cancel Indirect Read	RW	0x0

Bit	Name	Description	Access	Reset
0	start	When this bit is enabled, it will trigger an indirect read operation. The assumption is that the indirect start address and the indirect number of bytes register is setup before triggering the indirect read operation. **Value** **Description** 0x1 Trigger Indirect Read 0x0 No Indirect Read	RW	0x0

indrdwater

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705064

Offset: 0x64

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">level
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">level
RW 0x0</td></tr>
</table>

indrdwater Fields

Bit	Name	Description	Access	Reset
31:0	level	This represents the minimum fill level of the SRAM before a DMA peripheral access is permitted. When the SRAM fill level passes the watermark, an interrupt is also generated. This field can be disabled by writing a value of all zeroes. The units of this register are BYTES	RW	0x0

indrdstaddr

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705068

Offset: 0x68

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr RW 0x0															

indrdstaddr Fields

Bit	Name	Description	Access	Reset
31:0	addr	This is the start address from which the indirect access will commence its READ operation.	RW	0x0

indrdcnt

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF70506C

Offset: 0x6C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

indrdcnt Fields

Bit	Name	Description	Access	Reset
31:0	value	This is the number of bytes that the indirect access will consume. This can be bigger than the configured size of SRAM.	RW	0x0

indwr

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705070

Offset: 0x70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								indcnt RO 0x0		inddone RW 0x0	rdqueued RO 0x0	sramfull RO 0x0	rdstat RO 0x0	cancel RW 0x0	start RW 0x0

indwr Fields

Bit	Name	Description	Access	Reset
7:6	`indcnt`	This field contains the count of indirect operations which have been completed. This is used in conjunction with the indirect completion status field (bit 5).	RO	0x0
5	`inddone`	This field is set to 1 when an indirect operation has completed. Write a 1 to this field to clear it. **Value** **Description** 0x1 Indirect operation completed 0x0 No Action	RW	0x0
4	`rdqueued`	Two indirect write operations have been queued **Value** **Description** 0x1 Two Indirect write operation 0x0 No Action	RO	0x0
3	`sramfull`		RO	0x0
2	`rdstat`	Indirect write operation in progress (status) **Value** **Description** 0x1 Indirect write operation 0x0 No Action	RO	0x0
1	`cancel`	Writing a 1 to this bit will cancel all ongoing indirect write operations. **Value** **Description** 0x1 Cancel Indirect write operation 0x0 No Action	RW	0x0

Bit	Name	Description	Access	Reset
0	`start`	Writing a 1 to this bit will trigger an indirect write operation. The assumption is that the indirect start address and the indirect number of bytes register is setup before triggering the indirect write operation. **Value** **Description** 0x1 Trigger indirect write operation 0x0 No Action	`RW`	`0x0`

indwrwater

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF705074`

Offset: `0x74`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`level` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`level` `RW 0xFFFFFFFF`															

indwrwater Fields

Bit	Name	Description	Access	Reset
31:0	`level`	This represents the maximum fill level of the SRAM before a DMA peripheral access is permitted. When the SRAM fill level falls below the watermark, an interrupt is also generated. This field can be disabled by writing a value of all ones. The units of this register are bytes.	`RW`	`0xFFFFFFFF`

indwrstaddr

Module Instance	Base Address	Register Address
`qspiregs`	`0xFF705000`	`0xFF705078`

Offset: `0x78`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr RW 0x0															

indwrstaddr Fields

Bit	Name	Description	Access	Reset
31:0	addr	This is the start address from which the indirect access will commence its write operation.	RW	0x0

indwrcnt

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF70507C

Offset: 0x7C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
value RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RW 0x0															

indwrcnt Fields

Bit	Name	Description	Access	Reset
31:0	value	This is the number of bytes that the indirect access will consume. This can be bigger than the configured size of SRAM.	RW	0x0

flashcmd

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cmdopcode RW 0x0								enrddata RW 0x0	numrdatabytes RW 0x0			encmdaddr RW 0x0	enmodebit RW 0x0	numaddrbytes RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
enwrdata RW 0x0	numwrdatabytes RW 0x0			numdummybytes RW 0x0					Reserved					cmdexecstat RO 0x0	execcmd RW 0x0

flashcmd Fields

Bit	Name	Description	Access	Reset
31:24	cmdopcode	The command opcode field should be setup before triggering the command. For example, 0x20 maps to SubSector Erase. Writeing to the execute field (bit 0) of this register launches the command. NOTE : Using this approach to issue commands to the device will make use of the instruction type of the device instruction configuration register. If this field is set to 2'b00, then the command opcode, command address, command dummy bytes and command data will all be transferred in a serial fashion. If this field is set to 2'b01, then the command opcode, command address, command dummy bytes and command data will all be transferred in parallel using DQ0 and DQ1 pins. If this field is set to 2'b10, then the command opcode, command address, command dummy bytes and command data will all be transferred in parallel using DQ0, DQ1, DQ2 and DQ3 pins.	RW	0x0
23	enrddata	If enabled, the command specified in the command opcode field (bits 31:24) requires read data bytes to be received from the device. **Value** **Description** 0x1 Command Requires read data 0x0 No Action	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>22:20</td><td>numrddatabytes</td><td>Up to 8 data bytes may be read using this command. Set to 0 for 1 byte and 7 for 8 bytes.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Read 1 Byte</td></tr>
<tr><td>0x1</td><td>Read 2 Byte</td></tr>
<tr><td>0x2</td><td>Read 3 Byte</td></tr>
<tr><td>0x3</td><td>Read 4 Byte</td></tr>
<tr><td>0x4</td><td>Read 5 Byte</td></tr>
<tr><td>0x5</td><td>Read 6 Byte</td></tr>
<tr><td>0x6</td><td>Read 7 Byte</td></tr>
<tr><td>0x7</td><td>Read 8 Byte</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>19</td><td>encmdaddr</td><td>If enabled, the command specified in bits 31:24 requires an address. This should be setup before triggering the command via writing a 1 to the execute field.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>Command in bits 31:24 requires address</td></tr>
<tr><td>0x0</td><td>No Action</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>18</td><td>enmodebit</td><td>Set to 1 to ensure the mode bits as defined in the Mode Bit Configuration register are sent following the address bytes.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>Mode Bit follows address bytes</td></tr>
<tr><td>0x0</td><td>No Action</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17:16</td><td>numaddrbytes</td><td>Set to the number of address bytes required (the address itself is programmed in the FLASH COMMAND ADDRESS REGISTERS). This should be setup before triggering the command via bit 0 of this register. 2'b00 : 1 address byte 2'b01 : 2 address bytes 2'b10 : 3 address bytes 2'b11 : 4 address bytes
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Write 1 Address Byte</td></tr>
<tr><td>0x1</td><td>Write 2 Address Bytes</td></tr>
<tr><td>0x2</td><td>Write 3 Address Bytes</td></tr>
<tr><td>0x3</td><td>Write 4 Address Bytes</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>15</td><td>enwrdata</td><td>Set to 1 if the command specified in the command opcode field requires write data bytes to be sent to the device.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>Command requires write data bytes</td></tr>
<tr><td>0x0</td><td>No Action</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>14:12</td><td>numwrdatabytes</td><td>Up to 8 Data bytes may be written using this command.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Write 1 Byte</td></tr>
<tr><td>0x1</td><td>Write 2 Byte</td></tr>
<tr><td>0x2</td><td>Write 3 Byte</td></tr>
<tr><td>0x3</td><td>Write 4 Byte</td></tr>
<tr><td>0x4</td><td>Write 5 Byte</td></tr>
<tr><td>0x5</td><td>Write 6 Byte</td></tr>
<tr><td>0x6</td><td>Write 7 Byte</td></tr>
<tr><td>0x7</td><td>Write 8 Byte</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>11:7</td><td>numdummybytes</td><td>Set to the number of dummy bytes required This should be setup before triggering the command via the execute field of this register.</td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
1	cmdexecstat	Command execution in progress. **Value** — **Description** 0x1 — Command Execution Status 0x0 — No Action	RO	0x0
0	execcmd	Execute the command. **Value** — **Description** 0x1 — Execute Command 0x0 — No Action	RW	0x0

flashcmdaddr

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF705094

Offset: 0x94

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addr RW 0x0															

flashcmdaddr Fields

Bit	Name	Description	Access	Reset
31:0	addr	This should be setup before triggering the command with execute field (bit 0) of the Flash Command Control register. It is the address used by the command specified in the opcode field (bits 31:24) of the Flash Command Control register.	RW	0x0

flashcmdrddatalo

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF7050A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data RW 0x0															

flashcmdrddatalo Fields

Bit	Name	Description	Access	Reset
31:0	data	This is the data that is returned by the flash device for any status or configuration read operation carried out by triggering the event in the control register. The register will be valid when the polling bit in the control register is low.	RW	0x0

flashcmdrddataup

Device Instruction Register

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF7050A4

Offset: 0xA4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data RW 0x0															

flashcmdrddataup Fields

Bit	Name	Description	Access	Reset
31:0	data	This is the data that is returned by the FLASH device for any status or configuration read operation carried out by triggering the event in the control register. The register will be valid when the polling bit in the control register is low.	RW	0x0

flashcmdwrdatalo

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF7050A8

Offset: 0xA8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data RW 0x0															

flashcmdwrdatalo Fields

Bit	Name	Description	Access	Reset
31:0	data	This is the command write data lower byte. This should be setup before triggering the command with execute field (bit 0) of the Flash Command Control register. It is the data that is to be written to the flash for any status or configuration write operation carried out by triggering the event in the Flash Command Control register.	RW	0x0

flashcmdwrdataup

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF7050AC

Offset: 0xAC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
data RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data RW 0x0															

flashcmdwrdataup Fields

Bit	Name	Description	Access	Reset
31:0	data	This is the command write data upper byte. This should be setup before triggering the command with execute field (bit 0) of the Flash Command Control register. It is the data that is to be written to the flash for any status or configuration write operation carried out by triggering the event in the Flash Command Control register.	RW	0x0

moduleid

Module Instance	Base Address	Register Address
qspiregs	0xFF705000	0xFF7050FC

Offset: 0xFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							value RO 0x1001								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
value RO 0x1001															

moduleid Fields

Bit	Name	Description	Access	Reset
24:0	value		RO	0x1001

Document Revision History

Table 15-6: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address maps and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
November 2012	1.2	Minor updates.

Date	Version	Changes
May 2012	1.1	Added block diagram and system integration, functional description, programming model, and address map and register definitions sections.
January 2012	1.0	Initial release.

DMA Controller 16

2014.07.31

cv_54016 Subscribe Send Feedback

This chapter describes the direct memory access controller (DMAC) contained in the hard processor system (HPS). The DMAC is used to transfer data between memory and peripherals and other memory locations in the system. The DMA controller is an instance of the ARM Corelink DMA Controller (DMA-330).

Related Information

http://infocenter.arm.com/

For more information about ARM's DMA-330 controller, refer to the *CoreLink DMA Controller DMA-330 Revision: r1p1* Technical Reference Manual on the ARM info center website

Features of the DMA Controller

The HPS provides one DMAC to handle the data transfer between memory-mapped peripherals and memories, off-loading this work from the microprocessor unit (MPU) subsystem. The DMAC supports memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers. The DMAC supports up to eight logical channels for different levels of service requirements. It provides up to 31 peripheral handshake interfaces for peripheral hardware flow control.

The DMA controller contains an instruction processing block that enables it to process program code that controls a DMA transfer. The DMAC also contains an ARM Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) master interface unit to fetch the program code from system memory into its instruction cache. The AXI master interface also performs DMA data transfer as well. The DMA instruction execution engine executes the program code from its instruction cache and schedules read or write AXI instructions through the respective instruction queues. The DMAC also contains a multi-FIFO (MFIFO) data buffer that stores data that it reads, or writes, during a DMA transfer.

The DMAC provides 11 interrupt outputs to enable efficient communication of events to the MPU subsystem. The peripheral request interfaces support the connection of DMA-capable peripherals to enable memory-to-peripheral and peripheral-to-memory transfers to occur, without intervention from the processor. Because, the HPS supports some peripherals that do not comply with ARM DMA peripheral interface protocol, adapters are added to allow these peripherals to work with the DMAC.

The DMAC supports the following interface protocols:

- Synopsys protocol
    - Serial peripheral interface (SPI)
    - Universal asynchronous receiver/transmitter (UART)
    - Inter-integrated circuit (I^{2}C)
    - FPGA
- ARM protocol
    - Quad SPI flash controller
    - System trace macrocell (STM™)

Dual slave interfaces enable the operation of the DMA controller to be partitioned into the Secure state and Non-secure state. The network interconnect must be configured to ensure that only secure transactions can access the secure interface. The slave interfaces can access status registers and also directly execute instructions in the DMA controller.

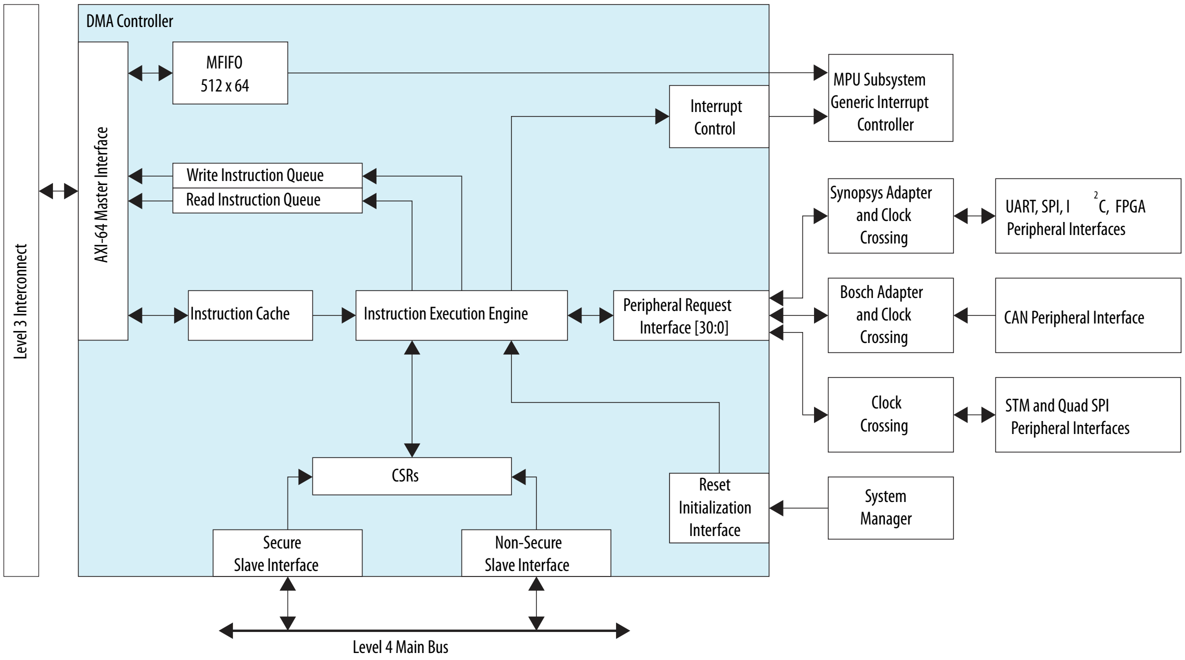
The DMAC has the following features:

- A small instruction set that provides a flexible method of specifying the DMA operations. This architecture provides greater flexibility than the fixed capabilities of a Linked-List Item (LLI) based DMA controller.
- Supports multiple transfer types:
    - Memory-to-memory
    - Memory-to-peripheral
    - Peripheral-to-memory
    - Scatter-gather
- Supports up to eight DMA channels.
- Supports up to eight outstanding AXI read and eight outstanding AXI write transactions.
- Enables software to schedule up to 16 outstanding read and (16) outstanding write instructions.
- Supports (11) interrupt lines into the MPU subsystem:
    - One (1) for DMA thread abort
    - Eight (8) for events
    - Two (2) for MFIFO buffer ECC
- Single and double bit ECC support
- Supports 31 peripheral request interfaces:
    - Four (4) for FPGA
    - Four (4) shared for FPGA or Controller area network (CAN)
    - Eight (8) for I^{2}C
    - Eight (8) for SPI
    - Two (2) for quad SPI
    - One (1) for System Trace Macrocell (STM)
    - Four (4) for UART

DMA Controller Block Diagram and System Integration

The following figure shows a block diagram of the DMAC and how it integrates into the rest of the HPS system.

Figure 16-1: DMA Controller Connectivity


Level 3 Interconnect
DMA Controller
AXI-64 Master Interface
MFIFO 512 x 64
Write Instruction Queue
Read Instruction Queue
Instruction Cache
Instruction Execution Engine
Interrupt Control
Peripheral Request Interface [30:0]
CSRs
Secure Slave Interface
Non-Secure Slave Interface
Reset Initialization Interface
MPU Subsystem Generic Interrupt Controller
Synopsys Adapter and Clock Crossing
UART, SPI, I²C, FPGA Peripheral Interfaces
Bosch Adapter and Clock Crossing
CAN Peripheral Interface
Clock Crossing
STM and Quad SPI Peripheral Interfaces
System Manager
Level 4 Main Bus

The `l4_main_clk` clock drives the DMA controller, controller logic, and all the interfaces. The DMA controller accesses the level 3 (L3) main switch with its 64-bit AXI master interface.

The DMA controller provides the following slave interfaces:

- Non-secure slave interface
- Secure slave interface

You can use these slave interfaces to access the registers that control the functionality of the DMA controller. The DMA controller implements TrustZone secure technology with one interface operating in the Secure state and the other operating in the Non-secure state.

Functional Description of the DMA Controller

This section describes the major interfaces and components of the DMAC, and its operation.

The DMAC contains an instruction processing block that processes program code that controls a DMA transfer. The program code is stored in a region of system memory that the DMAC accesses using its AXI master interface. The DMAC stores instructions temporarily in an internal cache.

The DMAC has eight DMA channels. Each channel supports a single concurrent thread of DMA operation. In addition, a single DMA manager thread exists, and you can use it to initialize the DMA channel threads.

The DMAC executes up to one instruction per clock cycle. To ensure that it regularly executes each active thread, the DMAC alternates by processing the DMA manager thread and then a DMA channel thread. It performs a round-robin process when selecting the next active DMA channel thread to execute.

The DMAC uses variable-length instructions that consist of one to six bytes. It provides a separate program counter (PC) register for each DMA channel.

A 16-line instruction cache to improve the instruction fetch performance is implemented. Each line contains eight words (4 bytes each) making the cache line size 32 bytes. If you take the 32 bytes per cache line times the 16 cache lines, the instruction cache is equal to 512 bytes. When a thread requests an instruction from an address, the cache performs a lookup. If a cache hit occurs, then the cache immediately provides the instruction. Otherwise, the thread is stalled while the DMAC performs a cache line fill with the AXI master interface. If an instruction spans the end of a cache line, the DMAC performs multiple cache accesses to fetch the instruction.

Note: When a cache line fill is in progress, the DMAC enables other threads to access the cache. But if another cache miss occurs, the pipeline stalls until the first line fill is complete.

When a DMA channel thread executes a load or store instruction, the DMAC adds the instruction to the relevant read or write queue. The DMAC uses these queues as an instruction storage buffer prior to it issuing the instructions on the AXI. The DMAC also contains an MFIFO data buffer in which it stores data that it reads, or writes, during a DMA transfer.

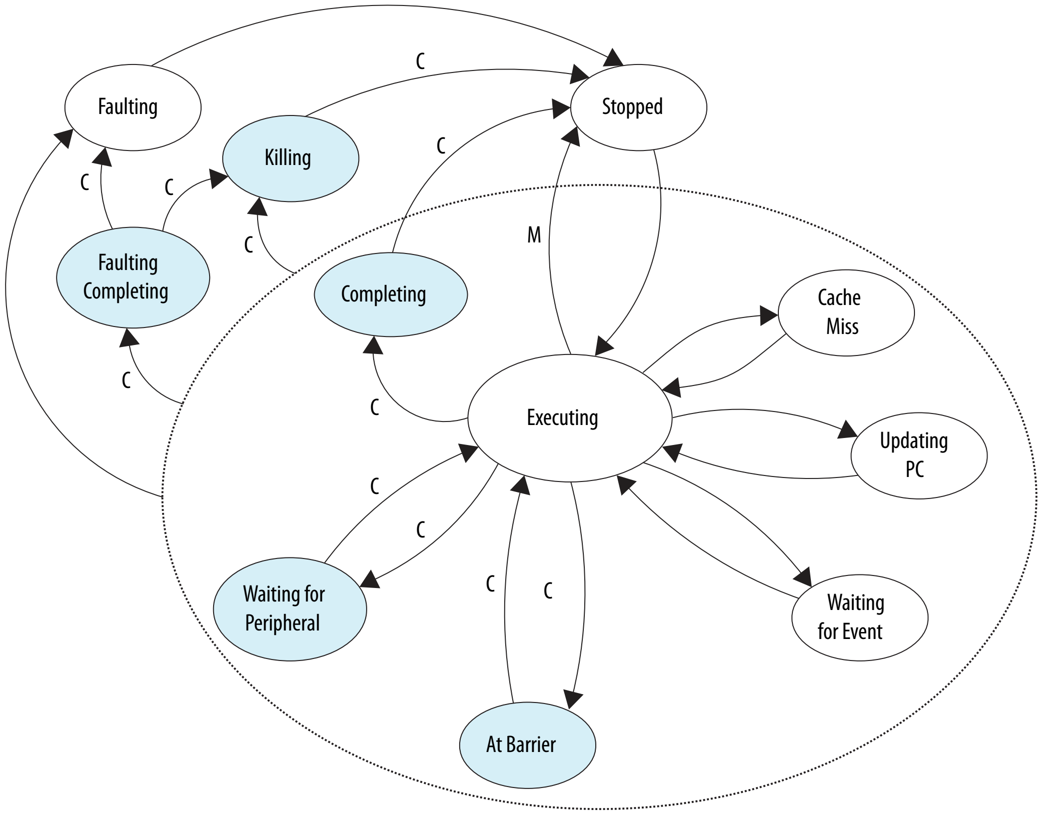
The DMAC provides multiple interrupt outputs to enable efficient communication of events to external microprocessors. The peripheral request interfaces support the connection of DMA-capable peripherals to enable memory-to-peripheral and peripheral-to-memory DMA transfers to occur, without intervention from a microprocessor.

Dual slave interfaces enable the operation of the DMAC to be partitioned into the Secure state and Non-secure states. You can access status registers and also directly execute instructions in the DMAC with the slave interfaces.

Operating States

The following figure shows the transitions among operating states for the DMA manager thread and DMA channel threads. The DMAC provides a separate state machine for each thread.

Figure 16-2: Thread Operating States



In the above figure, the DMAC permits all of the following:

- Only DMA channel threads can use states shown in blue
- Arcs with no letter designator indicate state transitions for the DMA manager and DMA channel threads, otherwise use is restricted as follows:
  - C—DMA channel threads only
  - M—DMA manager thread only
- States within the dotted line can transition to the Faulting completing, Faulting, or Killing

After the DMAC exits from reset, it sets all DMA channel threads to the Stopped state, and DMA manager thread moves to the Stopped state.

The following sections describe the *Thread Operating* states:

Stopped

The thread has an invalid PC (Program Counter) and it is not fetching instructions. Depending on the thread type, you can cause the thread to move to the Executing state by all of the following:

- DMA manager thread—Issuing the DMAGO instruction through the slave interface
- DMA channel thread—Programming the DMA manager thread to execute `DMAGO` for a DMA channel thread in the Stopped state

Executing

The thread has a valid PC and therefore the DMAC includes the thread when it arbitrates. The thread can then change to one of the following states under the following conditions:

- Stopped—When the DMA manager thread executes `DMAEND`
- Cache miss—When the instruction cache does not contain the next instruction for either the DMA manager thread or the DMA channel thread
- Updating PC—When the DMAC calculates the address of the next access in the cache
- Waiting for event—When a thread executes `DMAWFE`
- At barrier—When a DMA channel thread either:
  - Executes `DMARMB`, `DMAWMB`, or `DMAFLUSHP`
  - Updates control registers that affect alignment during a DMA Cycle
- Waiting for peripheral—When a DMA channel thread executes `DMAWFP`
- Killing—When a DMA channel thread executes `DMAKILL`
- Faulting completing—For a DMA channel thread when either:
  - The thread executes an undefined or invalid instruction
  - An AXI error occurs during an instruction fetch or data transfer
- Faulting—For the DMA manager thread when either:
  - The thread executes an undefined or invalid instruction
  - An AXI error occurs during an instruction fetch

  For a DMA channel thread when a watchdog timeout abort occurs.
- Completing—When a DMA channel thread executes `DMAEND`

Cache Miss

The thread is stalled and the DMAC is performing a cache line fill. After it completes the cache fill, the thread returns to the Executing state.

Updating PC

The DMAC is calculating the address of the next access in the cache. After it calculates the PC, the thread returns to the Executing state.

Waiting For Event

The thread is stalled and is waiting for the DMAC to execute `DMASEV` using the corresponding event number. After the corresponding event occurs, the thread returns to the Executing state.

At Barrier

A DMA channel thread is stalled and the DMAC is waiting for transactions on the AXI to complete. After the AXI transactions complete, the thread returns to the Executing state.

Waiting For Peripheral

A DMA channel thread is stalled and the DMAC is waiting for the peripheral to provide the requested data. After the peripheral provides the data, the thread returns to the Executing state.

Faulting Completing

A DMA channel thread is waiting for the AXI master interface to signal that the outstanding load or store transactions are complete. After the transactions complete, the thread moves to the Faulting state.

Faulting

The thread is stalled indefinitely. The thread moves to the Stopped state when you use the DBGCMD register to instruct the DMAC to execute DMAKILL for that thread.

Killing

A DMA channel thread is waiting for the AXI master interface to signal that the outstanding load or store transactions are complete. After the transactions complete, the thread moves to the Stopped state.

Completing

A DMA channel thread is waiting for the AXI master interface to signal that the outstanding load or store transactions are complete. After the transactions complete, the thread moves to the Stopped state.

Related Information

Updating DMA Channel Control Registers During a DMA Cycle on page 16-24

Initializing the DMAC

The DMAC provides several memory-mapped control signals that initialize its operating state when it exits from reset. The DMAC does not automatically begin executing code when it exits from reset. The system manager controls these memory-mapped control signals.

Related Information

System Manager on page 5-1

How to Set the Security State of the DMA Manager

The boot_manager_ns signal is the only method to set the security state of the DMA manager.

When the DMAC exits from reset, it reads the status of the boot_manager_ns signal and sets the security of the DMA manager.

Note: When set, the security state remains constant until a state transition on the dma_rst_n signal resets the DMAC.

Related Information

- DMA Manager Thread in Secure State on page 16-20
  Describes how the security state of the DMA manager affects how the DMAC operates.
- DMA Manager Thread in Non-Secure State on page 16-20

How to Set the Security State for the Interrupt Outputs

The DMAC provides the boot_irq_ns[7:0] signals to enable you to assign each irq[x] signal to a security state.

The boot_irq_ns[7:0] signals connect to the system manager. Before taking the DMA out of reset, you should program boot_irq_ns[7:0] through the system manager to control which interrupt bits are secure.

Note: The DMAC samples the `boot_irq_ns[7:0]` bits immediately after it comes out of reset, and then ignores them until the next reset.

When set, the security state of each `irq[x]` signal remains constant until a state transition on the `dma_rst_n` signal resets the DMAC.

Related Information

Security Usage on page 16-19
Describes how the security state of the irq[x] signals affects how the DMAC executes the DMAWFE and DMASEV instructions.

How to Set the Security State for a Peripheral Request Interface

The DMAC provides the signals to enable you to assign each peripheral request interface to a security state.

The `boot_periph_ns[31:0]` signals connect to the system manager. Before taking the DMA out of reset, you should program the `boot_periph_ns[31:0]` signals through the system manager to control which peripheral interfaces are secure.

Note: The DMAC samples the `boot_periph_ns[31:0]` bits immediately after it comes out of reset, and then ignores them until the next reset. When set, the security state of each peripheral request interface remains constant, until a state transition on the `dma_rst_n` signal resets the DMAC.

Related Information

Security Usage on page 16-19
Describes the effect of the security state of the peripheral request interfaces on the execution of the DMAWFP, DMALDP, DMASTP, or DMAFLUSHP instructions by a DMA channel thread.

Using the Slave Interfaces

The slave interfaces connect the DMAC to the level 4 (L4) main bus and enable a microprocessor to access the registers. Using these registers, a microprocessor can perform the following functions:

- Access the status of the DMA manager thread
- Access the status of the DMA channel threads
- Enable or clear interrupts
- Enable events
- Execute an instruction for the DMAC by programming the following debug registers:
    - `DBGCMD` register
    - `DBGINST0` register
    - `DBGINST1` register

Issuing Instructions to the DMAC using a Slave Interface

When the DMAC is operating, you can only issue the following instructions:

- `DMAGO`—Starts a DMA transaction using a DMA channel that you specify
- `DMASEV`—Signals the occurrence of an event, or interrupt, using an event number that you specify
- `DMAKILL`—Terminates a thread

You must ensure that you use the appropriate slave interface, depending on the security state in which the `boot_manager_ns` signal initializes the DMAC. For example, if the DMAC is in the Secure state, you must issue the instruction using the secure slave interface, otherwise the DMAC ignores the instruction. You can use the secure or non-secure slave interface to start or restart a DMA channel when the DMAC is in the Non-secure state.

Note: Before you can issue instructions using the debug instruction registers or the DBGCMD register, you must read the DBGSTATUS register to ensure that debug is idle, otherwise the DMAC ignores the instructions.

The DMAC immediately processes any instructions received from a slave interface, unless the pipeline is busy processing another instruction.

Note: Prior to issuing DMAGO, you must ensure that the system memory contains a suitable program for the DMAC to execute, starting at the address that the DMAGO specifies.

Using DMAGO with the Debug Instruction Registers

The following example shows the necessary steps to start a DMA channel thread using the debug instruction registers:

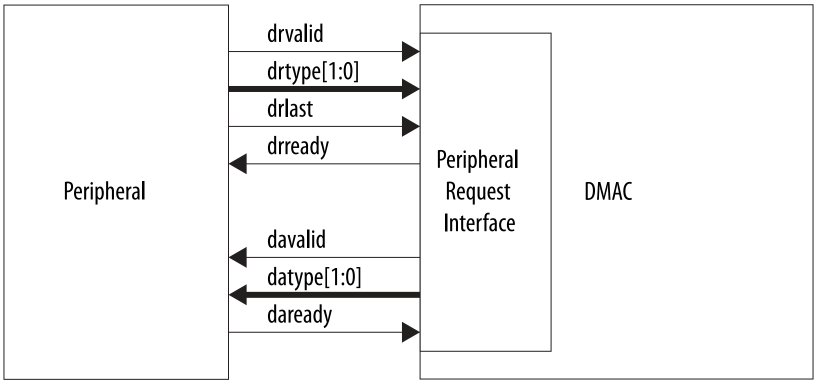
1. Create a program for the DMA channel.
2. Store the program in a region of system memory.
3. Program one of the slave interfaces on the DMAC to a DMAGO instruction as follows:
   a. Poll the DBGSTATUS register to ensure that debug is idle, and the dbgstatus bit is 0.
   b. Write to the DBGINST0 register and enter all of the following:
      - Instruction byte 0 encoding for DMAGO
      - Instruction byte 1 encoding for DMAGO
      - Debug thread bit to 0 to select the DMA manager thread
   c. Write to the DBGINST1 register with the DMAGO instruction byte [5:2] data. You must set these four bytes to the address of the first instruction in the program, that is written to system memory in step 2 above.
4. Instruct the DMAC to execute the instruction that the debug instruction registers contain by writing zero to the DBGCMD register. The DMAC starts the DMA channel thread and sets the dbgstatus bit to 1. After the DMAC completes execution of the instruction, it clears the dbgstatus bit to 0.

Peripheral Request Interface

The following figure shows that the peripheral request interface consists of a peripheral request bus and a DMAC acknowledge bus that use the prefixes:

- dr—The peripheral request bus
- da—The DMAC acknowledge bus

Figure 16-3: Request and Acknowledge Buses on the Peripheral Request Interface



The peripheral indicates the following on the request bus:

- Request a single transfer
- Request a burst transfer
- Acknowledge a flush request

The peripheral indicates the DMAC when it issues the last request of the DMA transfer sequence.

The DMAC can indicate the following on the acknowledge bus:

- When it completes the requested single transfer
- When it completes the requested burst transfer
- When it issues a flush request

You can assign a peripheral request interface to any of the DMA channels. When a DMA channel thread executes `DMAWFP`, the value programmed in the peripheral [4:0] field specifies the peripheral associated with that DMA channel.

The DMAC supports 31 peripheral request handshakes. Each request handshake can receive up to four outstanding requests, and is assigned a specific peripheral device ID. The following table lists the peripheral device ID assignments.

Table 16-1: Peripheral Request Interface Mapping

Peripheral	Request Interface ID	Protocol	Peripheral	Request Interface ID	Protocol
FPGA 0	0	Synopsys	SPI Master 0 TX	16	Synopsys
FPGA 1	1	Synopsys	SPI Master 0 RX	17	Synopsys
FPGA 2	2	Synopsys	SPI Slave 0 TX	18	Synopsys
FPGA 3	3	Synopsys	SPI Slave 0 RX	19	Synopsys

FPGA 4 /CAN 0 interface 1	4	Synopsys / Bosch	SPI Master 1 TX	20	Synopsys
FPGA 5 /CAN 0 interface 2	5	Synopsys / Bosch	SPI Master 1 RX	21	Synopsys
FPGA 6 /CAN 1 interface 1	6	Synopsys / Bosch	SPI Slave 1 TX	22	Synopsys
FPGA 7 /CAN 1 interface 2	7	Synopsys / Bosch	SPI Slave 1 RX	23	Synopsys
I^{2}C 0 TX	8	Synopsys	Quad SPI Flash TX	24	ARM
I^{2}C 0 RX	9	Synopsys	Quad SPI Flash RX	25	ARM
I^{2}C 1 TX	10	Synopsys	STM	26	ARM
I^{2}C 1 RX	11	Synopsys	Reserved	27	Synopsys
I^{2}C 2 TX (EMAC)	12	Synopsys	UART 0 TX	28	Synopsys
I^{2}C 2 RX (EMAC)	13	Synopsys	UART 0 RX	29	Synopsys
I^{2}C 3 TX (EMAC)	14	Synopsys	UART 1 TX	30	Synopsys
I^{2}C 3 RX (EMAC)	15	Synopsys	UART 1 RX	31	Synopsys

Note: Request interface numbers 4 through 7 are multiplexed between the CAN controllers and soft logic implemented in the FPGA fabric. The switching between the CAN controller and FPGA interfaces is controlled by the system manager.

Request Acceptance Capability

The DMAC can accept one active request for each peripheral request interface. An active request exists when the DMAC has not started the requested AXI data transfers.

Peripheral Length Management

A peripheral can control the quantity of data that a DMA cycle contains, without the DMAC being aware of how many data transfers it contains. The peripheral controls the DMA cycle in one of the following ways:

- Selects a single transfer
- Selects a burst transfer
- Notifies the DMAC when it commences the final request in the current series

When the DMAC executes a DMAWFP peripheral instruction, it halts execution of the thread and waits for the peripheral to send a request. When the peripheral sends the request, the DMAC sets the request flags depending on the state of the following signals:

- drtype_<x>[1:0] —The DMAC sets the state of the request_type flag:
  - drtype_<x>[1:0]=b00—request_type <x> = Single
  - drtype_<x>[1:0]=b01—request_type <x> = Burst
- drlast_<x>—The DMAC sets the state of the request_last flag:
  - drlast_<x>=0—request_last <x> = 0
  - drlast_<x>=1—request_last <x> = 1

Note: If the DMAC executes a DMAWFP single or DMAWFP burst instruction then the DMAC sets:

- The request_type<x> flag to Single or Burst, respectively
- The request_last<x> flag to 0

DMALPFE is an assembler directive which forces the associated DMALPEND instruction to have its nf bit set to 0. This creates a program loop that does not use a loop counter to terminate the loop.

The DMAC exits the loop when the request_last flag is set to 1.

The DMAC conditionally executes the following instructions, depending on the state of the request_type and request_last flags:

- DMALD , DMAST , DMALPEND

When these instructions use the optional B|S suffix then the DMAC executes a DMANOP if the request_type flag does not match.

- DMALDP<B|S> , DMASTP<B|S>

The DMAC executes a DMANOP if the request_type <x> flag does not match the B|S

- DMALPEND

When the nf bit is 0, the DMAC executes a DMANOP if the request_last flag is set.

Use the DMALDB, DMALDPB, DMASTB and DMASTPB instructions if you require the DMAC to issue a burst transfer when the DMAC receives a burst request. The values in the CCR *n* register control the amount of data that the DMAC transfers.

Use the DMALDS, DMALDPS, DMASTS and DMASTPS instructions if you require the DMAC to issue a single transfer when the DMAC receives a single request. The DMAC ignores the value of the src_burst_len and dst_burst_len fields in the CCR*n* register and sets the arlen[3:0] or awlen[3:0] buses to 0x0.

DMAC Length Management

The DMAC controls the total amount of data. The peripheral uses the peripheral request interface to notify the DMAC when it requires the DMAC to transfer data to or from the peripheral. The DMA channel thread controls how the DMAC responds to the peripheral requests.

The following constraints apply to DMAC length management:

- The total quantity of data for all the single requests from a peripheral must be less than the quantity of data for a burst request for that peripheral.

  Note: The `CCRn` register controls how much data is transferred for a burst request and a single request. Altera recommends that you do not update a `CCRn` register when a transfer is in progress for channel `n`.

- After the peripheral sends a burst request, the peripheral must not send a single request until the DMAC acknowledges that the burst request is complete.

Program the `DMAWFP` single instruction when you require the program thread to halt execution until the peripheral request interface receives any request type.

- Single— If the head entry in the request FIFO buffer is a single request type, the DMAC pops the entry from the FIFO buffer and continues program execution.
- Burst—If the head entry in the request FIFO buffer is a burst request type, the DMAC leaves the entry in the FIFO buffer and continues program execution.

  Note: The burst request entry remains in the request FIFO buffer until the DMAC executes a `DMAWFP` burst instruction or a `DMAFLUSHP` instruction.

Program the `DMAWFP` burst instruction when you require the program thread to halt execution until the peripheral request interface receives a burst request.

- Single—If the head entry in the request FIFO buffer is a single request type, the DMAC removes the entry from the FIFO buffer and continues program execution.
- Burst—If the head entry in the request FIFO buffer is a burst request type, the DMAC pops the entry from the FIFO buffer and continues program execution.

Program the `DMALDP` instruction when you require the DMAC to send an acknowledgment to the peripheral when it completes the AXI read transfers. Similarly, use the `DMASTP` instruction when you require the DMAC to send an acknowledgment to the peripheral when it completes the AXI write transfers. The DMAC uses the acknowledge bus to signal a transfer acknowledgment to peripheral <x>.

Note: The DMAC sends an acknowledgment for a read transaction when the `rvalid` and `rlast` signals are high and for a write transaction when `bvalid` signal is high. The DMAC might send an acknowledgment to the peripheral while the transfer of write data to the end destination is still in progress.

Use the `DMAFLUSHP` instruction to reset the request FIFO buffer for the peripheral request interface. After the DMAC executes `DMAFLUSHP`, it ignores peripheral requests until the peripheral acknowledges the flush request. This enables the DMAC and peripheral to synchronize with each other.

ARM Protocol

The DMA peripheral request interface communicates with peripherals by either the ARM protocol or the Synopsys protocol.

For peripherals using the ARM protocol, clock-crossing logic is the only logic between the DMA and the peripheral.

Synopsys Protocol Overview

The FIFO is Avalon®-MM based with hardware flow control that is compatible with the DMA-330 core in the HPS. The Synopsys protocol is used for the hardware flow control which is not the same as the ARM® protocol used by the DMA-330 core.

For peripherals using the Synopsys protocol, clock-crossing logic and protocol adaptation logic exist between the DMA and the peripheral.

The Synopsys protocol is described, below:

- A single transfer line is asserted whenever the peripheral is able to accept a single word transfer.
- A burst transfer line is asserted when the peripheral is able to accept a burst transfer. The burst transfer size needs to be identically set in both the DMA and peripheral to avoid potential data loss or corruption.
- The single and burst transfer lines must remain asserted until the DMA issues an acknowledge back to the peripheral.

  Note: If the peripheral needs to be reset, then the DMA should be instructed to flush the request which will be translated into an acknowledgement by the Synopsys protocol adapter.
- When a transfer is acknowledged the peripheral must deassert the burst and single request lines for the duration of the acknowledgment.

Note: A transmit or receive request is not complete until the DMA issues an acknowledge back to the peripheral.

Using Events and Interrupts

The DMAC can support eight events and interrupts. The `INTEN` register determines whether each event-interrupt resource is an event or an interrupt.

When the DMAC executes a `DMASEV` instruction it modifies the event-interrupt resource that you specify. The `INTEN` register sets the event-interrupt resource to function as an:

- **Event**—The DMAC generates an event for the specified event-interrupt resource. When the DMAC executes a `DMAWFE` instruction for the same event-interrupt resource then it clears the event.
- **Interrupt**—The DMAC sets the `irq <event_num>` signal high, where <event_num> is the number of the specified event resource. To clear the interrupt you must write to the `INTCLR` register.

Using an Event to Restart DMA Channels

When you program the `INTEN` register to generate an event, you can use the `DMASEV` and `DMAWFE` instructions to restart one or more DMA channels.

DMAC executes DMAWFE before DMASEV

To restart a single DMA channel:

1. The first DMA channel executes `DMAWFE` and then stalls while it waits for the event to occur.
2. The other DMA channel executes `DMASEV` using the same event number. This generates an event, and the first DMA channel restarts. The DMAC clears the event, one clock cycle after it executes `DMASEV`.

You can program multiple channels to wait for the same event. For example, if four DMA channels have all executed DMAWFE for event 2, and another DMA channel executes DMASEV for event 2, the four DMA channels all restart at the same time. The DMAC clears the event, one clock cycle after it executes DMASEV.

DMAC executes DMASEV before DMAWFE

If the DMAC executes DMASEV before another channel executes DMAWFE, the event remains pending until the DMAC executes DMAWFE. When the DMAC executes DMAWFE, it halts execution for one aclk clock cycle, clears the event and then continues execution of the channel thread.

For example, if the DMAC executes DMASEV 6 and none of the other threads have executed DMAWFE 6, then the event remains pending. If the DMAC executes the DMAWFE 6 instruction for channel 4 and then executes the DMAWFE 6 instruction for channel 3, the following actions occur:

1. The DMAC halts execution of the channel 4 thread for one clock cycle.
2. The DMAC clears event 6.
3. The DMAC resumes execution of the channel 4 thread.
4. The DMAC halts execution of the channel 3 thread and the thread stalls while it waits for the next occurrence of event 6.

Interrupting the MPU Subsystem

The DMAC provides the irq[x] signals for use as active-high level-sensitive interrupts to the MPU subsystem. When you program the INTEN register to generate an interrupt, after the DMAC executes DMASEV, it sets the corresponding irq[x] signal high.

The MPU subsystem can clear the interrupt by writing to the INTCLR register.

Note: Executing DMAWFE does not clear an interrupt.

If you use the DMASEV instruction to notify a microprocessor when the DMAC completes a DMALD or DMAST instruction then you should insert a memory barrier instruction before the DMASEV. Otherwise the DMAC might signal an interrupt before the AXI transfers complete.

The following program shows the example of Memory Barrier Instruction.

```
DMALD
DMAST
# Issue a write memory barrier
# Wait for the AXI write transfer to complete before the DMAC
# can send an interrupt
DMAWMB
# The DMAC sends the interrupt
DMASEV
```

Aborts

Abort Types

An abort can be classified as either precise or imprecise, depending on whether the DMAC provides an abort handler with the precise state of the DMAC when the abort occurs.

- **Precise Abort**: The DMAC updates the PC register with the address of the instruction that created the abort.
- **Imprecise Abort**: The PC register might contain the address of an instruction which did not cause the abort to occur.

Abort Sources

The DMAC indicates a precise abort under any of the following conditions:

- A DMA channel thread in the Non-secure state attempts to program its `CCRn` register and generate a secure AXI transaction.
- A DMA channel thread in the Non-secure state executes `DMAWFE` or `DMASEV` for an event that is set as secure. The `boot_irq_ns` memory-mapped control signals initialize the security state for an event.

Note: For each event, the INTEN register controls if the DMAC generates an event or signals an interrupt.

- A DMA channel thread attempts to execute `DMAST` but the DMAC calculates that when it eventually performs the store, the MFIFO buffer contains insufficient data to enable it to complete the store.
- A DMA channel thread in the Non-secure state executes `DMAWFP`, `DMALDP`, `DMASTP`, or `DMAFLUSHP` for a peripheral request interface that is set as secure. The `boot_periph_ns` memory-mapped control signals initialize the security state for a peripheral request interface.
- A DMA manager thread in the Non-secure state executes `DMAGO` to attempt to start a secure DMA channel thread.
- The DMAC receives an ERROR response on the AXI master interface when it performs an instruction fetch.
- A thread executes an undefined instruction.
- A thread executes an instruction with an operand that is invalid for the configuration of the DMAC.

Note: When the DMAC signals a precise abort, the instruction that triggers the abort is not executed. Instead, the DMAC executes a `DMANOP`.

The DMAC signals an imprecise abort under the following conditions:

- The DMAC receives an ERROR response on the AXI master interface when it performs a data load.
- The DMAC receives an ERROR response on the AXI master interface when it performs a data store.
- A DMA channel thread executes `DMALD` or `DMAST`, and the MFIFO buffer is too small to hold the required amount of data.
- A DMA channel thread executes `DMAST` but the thread has not executed sufficient `DMALD` instructions.
- A DMA channel thread locks up because of resource starvation, and this causes the internal watchdog timer to time out.

Watchdog Abort

The DMAC can lock up if one or more DMA channel programs are running and the MFIFO buffer is too small to satisfy the storage requirements of the DMA programs.

The DMAC contains logic to prevent it from remaining in a state where it is unable to complete a DMA transfer.

The DMAC detects a lock up when all of the following conditions occur:

- Load queue is empty.
- Store queue is empty.
- All of the running channels are prevented from executing a `DMALD` instruction either because the MFIFO buffer does not have sufficient free space or another channel owns the load-lock.

When the DMAC detects a lockup, it signals an interrupt and can also abort the contributing channels. The DMAC behavior depends on the state of the wd_irq_only bit in the WD register, if:

- wd_irq_only=0—The DMAC aborts all of the contributing DMA channels and sets the irq_abort signal high.
- wd_irq_only=1—The DMAC sets the irq_abort signal high.

Related Information

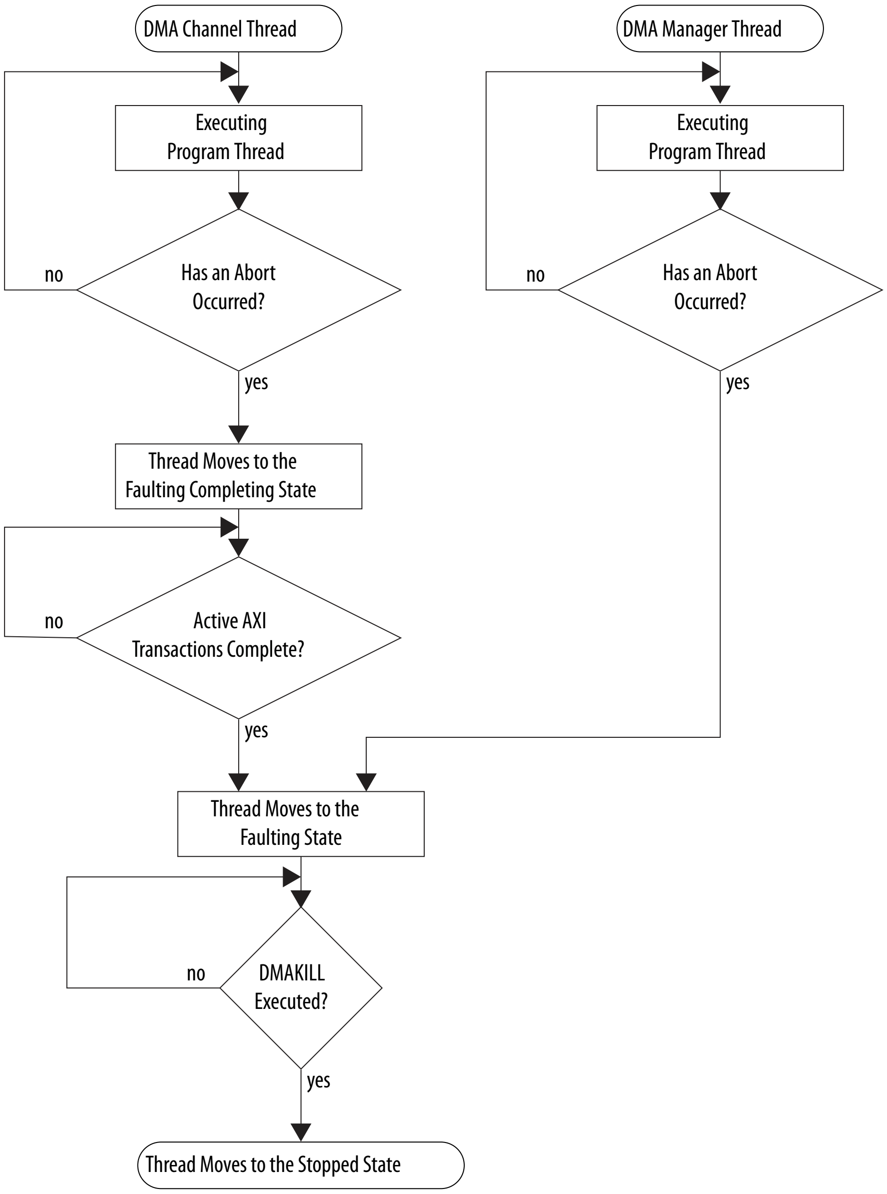
Resource Sharing Between DMA Channels on page 16-24

Abort Handling

The architecture of the DMAC is not designed to recover from an abort. You must use an external agent, such as a microprocessor, to terminate a thread when an abort occurs.

The following figure shows the operating states for the DMA channel and DMA manager threads after an abort occurs.

Figure 16-4: Abort Process



After an abort occurs, the action of the DMAC depends on the thread type:

- DMA channel thread—The thread immediately moves to the Faulting completing state. In this state, the DMAC performs the following operations:
  - Sets the `irq_abort` signal high.
  - Stops executing instructions for the DMA channel.
  - Invalidates all cache entries for the DMA channel updates the `CPC` *n* register to contain the address of the aborted instruction provided that the abort is precise.
  - Does not generate AXI accesses for any instructions remaining in the read queue and write queue.
  - Permits currently active AXI transactions to complete.

  Note: After the transactions for the DMA channel finish, the thread moves to the Faulting state.
- DMA manager thread—The thread immediately moves to the Faulting state and the DMAC sets the `irq_abort` signal high.

  The external agent can respond to the assertion of the `irq_abort` signal by all of the following:
  - Reading the status of the `FSRD` register to determine if the DMA manager is Faulting. In the Faulting state, the `FSRD` register provides the cause of the abort.
  - Reading the status of the `FSRC` register to determine if a DMA channel is Faulting. In the Faulting state, the `FSRC` register provides the cause of the abort.

  To enable a thread in the Faulting state to move to the Stopped state, the external agent must:
  - Program the `DBGINST0` register with the encoding for the `DMAKILL` instruction.
  - Write to the `DBGCMD` register.

  Note: If the aborted thread is secure, you must use the secure slave interface to update these registers.

  After a thread in the Faulting state executes `DMAKILL`, it moves to the Stopped state.

Security Usage

When the DMAC exits from reset, the status of the configuration signals configures the security for:

- DMA manager thread—The `DNS` bit in the DSR register returns the security state of the DMA manager thread.
- Events and interrupts—The `INS` bit in the `CR3` register returns the security state of the event-interrupt resources.
- Peripheral request interfaces—The `PNS` bit in the `CR4` register returns the security state of these interfaces.

Additionally, each DMA channel thread contains a dynamic non-secure bit, `CNS`, that is valid when the channel is not in the Stopped state.

DMA Manager Thread in Secure State

If the DNS bit is 0, the DMA manager thread operates in the Secure state and performs only secure instruction fetches. When a DMA manager thread in the Secure state processes:

- DMAGO—The DMAC uses the status of the ns bit, to set the security state of the DMA channel thread by writing to the CNS bit for that channel.
- DMAWFE—The DMAC halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit.
- DMASEV—The DMAC sets the corresponding bit in the INT_EVENT_RIS register, irrespective of the security state of the corresponding INS bit.

DMA Manager Thread in Non-Secure State

If the DNS bit is 1, the DMA manager thread operates in the Non-secure state, and it only performs non-secure instruction fetches. When a DMA manager thread in the Non-secure state processes:

- DMAGO—The DMAC uses the status of the ns bit, to control if it starts a DMA channel thread.
  - If ns = 0—The DMAC does not start a DMA channel thread and instead it:
    - Executes an NOP.
    - Sets the FSRD register.
    - Sets the dmago_err bit in the FTRD register.
    - Moves the DMA manager to the Faulting state.
  - If ns = 1—The DMAC starts a DMA channel thread in the Non-secure state and programs the CNS bit to be non-secure.
- DMAWFE—The DMAC uses the status of the corresponding INS bit, in the CR3 register, to control whether it waits for the event.
  - If INS = 0—The event is in the Secure state. The DMAC:
    - Executes an NOP.
    - Sets the FSRD register.
    - Sets the mgr_evnt_err bit in the FTRD register.
    - Moves the DMA manager to the Faulting state.
  - If INS = 1—The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.
- DMASEV—The DMAC uses the status of the corresponding INS bit, in the CR3 register, to control if it creates the event interrupt.
  - If INS = 0—The event-interrupt resource is in the Secure state. The DMAC:
    - Executes a NOP.
    - Sets the FSRD register.
    - Sets the mgr_evnt_err bit in the FTRD register.
    - Moves the DMA manager to the Faulting state.
  - If INS = 1—The event-interrupt resource is in the Non-secure state. The DMAC creates the event interrupt.

DMA Channel Thread in Secure State

When the CNS bit is 0, the DMA channel thread is programmed to operate in the Secure state and it only performs secure instruction fetches.

When a DMA channel thread in the Secure state processes the following instructions:

- DMAWFE—The DMAC halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit, in the CR3 register.
- DMASEV—The DMAC creates the event interrupt, irrespective of the security state of the corresponding INS bit, in the CR3 register.
- DMAWFP—The DMAC halts execution of the thread until the peripheral signals a DMA request. When this occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding PNS bit, in the CR4 register.
- DMALDP and DMASTP—The DMAC sends a message to the peripheral to communicate that data transfer is complete, irrespective of the security state of the corresponding PNS bit, in the CR4 register.
- DMAFLUSHP—The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status, irrespective of the security state of the corresponding PNS bit, in the CR4 register.

When a DMA channel thread is in the Secure state, it enables the DMAC to perform secure and non-secure AXI accesses.

DMA Channel Thread in Non-Secure State

When the CNS bit is 1, the DMA channel thread is programmed to operate in the Non-secure state and it only performs non-secure instruction fetches.

When a DMA channel thread in the Non-secure state processes the following instructions:

- DMAWFE—The DMAC uses the status of the corresponding INS bit, in the CR3 register, to control if it waits for the event.
    - If INS = 0—The event is in the Secure state. The DMAC:
        - Executes an NOP.
        - Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
        - Sets the ch_evnt_err bit in the FTRn register.
        - Moves the DMA channel to the Faulting completing state.
    - If INS = 1—The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.
- DMASEV—The DMAC uses the status of the corresponding INS bit, in the CR3 register, to control if it creates the event.
    - If INS = 0—The event-interrupt resource is in the Secure state. The DMAC:
        - Executes an NOP.
        - Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
        - Sets the ch_evnt_err bit in the FTRn register.
        - Moves the DMA channel to the Faulting completing state.
    - If INS = 1—The event-interrupt resource is in the Non-secure state. The DMAC creates the event interrupt.
- DMAWFP—The DMAC uses the status of the corresponding PNS bit, in the CR4 register, to control if it waits for the peripheral to signal a request.
    - If PNS = 0—The peripheral is in the Secure state. The DMAC:
        - Executes an NOP.
        - Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
        - Sets the ch_periph_err bit in the FTRn register.
        - Moves the DMA channel to the Faulting completing state.
    - If PNS = 1—The peripheral is in the Non-secure state. The DMAC halts execution of the thread and waits for the peripheral to signal a request.
- DMALDP and DMASTP—The DMAC uses the status of the corresponding PNS bit, in the CR4 register, to control if it sends an acknowledgement to the peripheral.
    - If PNS = 0—The peripheral is in the Secure state. The DMAC:
        - Executes an NOP.
        - Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
        - Sets the ch_periph_err bit in the FTRn register.
        - Moves the DMA channel to the Faulting completing state.
    - If PNS = 1—The peripheral is in the Non-secure state. The DMAC sends a message to the peripheral to communicate when the data transfer is complete.
- DMAFLUSHP—The DMAC uses the status of the corresponding PNS bit, in the CR4 register, to control if it sends a flush request to the peripheral.
    - If PNS = 0—The peripheral is in the Secure state. The DMAC:
        - Executes an NOP.
        - Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
        - Sets the ch_periph_err bit in the FTRn register.
        - Moves the DMA channel to the Faulting completing state.
    - If PNS = 1—The peripheral is in the Non-secure state. The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status.

When a DMA channel thread is in the Non-secure state, and a DMAMOV CCR instruction attempts to program the channel to perform a secure AXI transaction, the DMAC:

1. Executes a DMANOP.
2. Sets the appropriate bit in the FSRC register that corresponds to the DMA channel number.
3. Sets the ch_rdwr_err bit in the FTR*n* register.
4. Moves the DMA channel thread to the Faulting completing state.

Programming Restrictions

Certain restrictions apply when programming the DMAC.

Fixed Unaligned Bursts

The DMAC does not support fixed unaligned bursts. If you program the following conditions, the DMAC treats this as a programming error:

- **Unaligned read**
  - src_inc field is 0 in the CCRn register.
  - The SARn register contains an address that is not aligned to the size of data that the src_burst_size field contains.
- **Unaligned write**
  - dst_inc field is 0 in the CCRn register.
  - The DARn register contains an address that is not aligned to the size of data that the dst_burst_size field contains.

Endian Swap Size Restrictions

If you program the endian_swap_size field in the CCRn register, to enable a DMA channel to perform an endian swap, then you must set the corresponding SAR register and the corresponding DARn register to contain an address that is aligned to the size that the endian_swap_size field specifies before executing any DMALD or DMAST instructions.

Note: If you update any of endian_swap_size, SARn , or DARn , for example, using a DMAADDH SAR instruction, then you must ensure that the SARn and DARn registers contain an address aligned to the size that the endian_swap_size field specifies before executing any additional DMALD or DMAST instructions.

If you program the src_inc field in the CCR register to use a fixed address, you must program the src_burst_size field to select a burst size that is greater than or equal to the value that the endian_swap_size field specifies. Similarly, if you program the dst_inc field to select a fixed destination address, you must program the dst_burst_size field to select a burst size that is greater than or equal to the value that the endian_swap_size field specifies.

If you program the dst_inc field in the CCRn register to use an incrementing address, you must program the CCRn register so that dst_burst_len times dst_burst_size is a multiple of endian_swap_size. For example, if endian_swap_size = 2, 32-bit, and dst_burst_size = 1, 2 bytes per beat, then you can program dst_burst_len = 1, 3, 5, ..., 15, that is 2, 4, 6, ..., 16 transfers.

Updating DMA Channel Control Registers During a DMA Cycle

Prior to the DMAC executing a sequence of DMALD and DMAST instructions, the values you program in to the CCRn register, SARn register, and DARn register control the data byte lane manipulation that the DMAC performs when it transfers the data from the source address to the destination address.

You can update these registers during a DMA cycle but if you change certain register fields then it can cause the DMAC to discard data. The following sections describe the register fields that might have a detrimental impact on a data transfer.

Updates that affect the destination address

If you use a DMAMOV instruction to update the DARn register or CCRn register part way through a DMA cycle then this might cause a discontinuity in the destination data stream.

A discontinuity occurs if you change any of the following:

- endian_swap_size field.
- dst_inc bit.
- dst_burst_size field when dst_inc = 0, that is, fixed-address burst.
- DARn register so that it modifies the destination byte lane alignment. Because the bus width is 64 bits, you change bits [2:0] in the DARn register.

When a discontinuity in the destination data stream occurs, the DMAC:

1. Halts execution of the DMA channel thread.
2. Completes all outstanding read and write operations for the channel. That is, as if the DMAC were executing DMARMB and DMAWMB.
3. Discards any residual MFIFO buffer data for the channel.
4. Resumes execution of the DMA channel thread.

Updates that affect the source address

If you use a DMAMOV instruction to update the SARn register or CCRn register part way through a DMA cycle then this might cause a discontinuity in the source data stream.

A discontinuity occurs if you change any of the following:

- src_inc bit.
- src_burst_size field.
- SAR register so that it modifies the source byte lane alignment. Because the bus width is 64 bits, you change bits [2:0] in the SAR register.

When a discontinuity in the source data stream occurs, the DMAC:

1. Halts execution of the DMA channel thread.
2. Completes all outstanding read operations for the channel. That is, as if the DMAC were executing DMARMB
3. Resumes execution of the DMA channel thread. No data is discarded from the MFIFO buffer.

Resource Sharing Between DMA Channels

DMA channel programs share the MFIFO buffer data storage resource. You must not start a set of concurrently running DMA channel programs with a resource requirement that exceeds 512, the size of the MFIFO buffer. If you exceed this limit then the DMAC might lock up and generate a Watchdog abort, refer to “Watchdog Abort”.

The DMAC includes a mechanism called the *load lock* to ensure that the shared MFIFO buffer resource is used correctly. The load-lock is either owned by one channel, or it is free. The channel that owns the load-lock can execute `DMALD` instructions successfully. A channel that does not own the load-lock pauses at a `DMALD` instruction until it takes ownership of the load-lock.

A channel claims ownership of the load lock when:

- It executes a `DMALD` or `DMALDP` instruction
- No other channel currently owns the load-lock.

A channel releases ownership of the load-lock when any of the following occur:

- It executes a `DMAST`, `DMASTP`, or `DMASTZ`
- It reaches a barrier, that is, it executes `DMARMB` or `DMAWMB`
- It waits, that is, it executes `DMAWFP` or `DMAWFE`
- It terminates normally, that is, it executes `DMAEND`
- It aborts for any reason, including `DMAKILL`.

The MFIFO buffer resource usage of a DMA channel program is measured in MFIFO buffer entries, and rises and falls as the program proceeds. The MFIFO buffer resource requirement of a DMA channel program is described using a *static requirement* and a *dynamic requirement* which are affected by the load-lock mechanism.

The static requirement is defined to be the maximum number of MFIFO buffer entries that a channel is currently using before that channel does one of the following:

- Executes a WFP or WFE instruction
- Claims ownership of the load-lock.

The dynamic requirement is defined to be the difference between the static requirement and the maximum number of MFIFO buffer entries that a channel program uses at any time during its

To calculate the total MFIFO buffer requirement, add the largest dynamic requirement to the sum of all the static requirements.

To avoid DMAC lockup, the total MFIFO buffer requirement of the set of channel programs must be equal to or less than 512, the MFIFO buffer depth.

Related Information

- **Watchdog Abort** on page 16-16
- **MFIFO Buffer Usage Overview** on page 16-45

Clocks and Resets

Clock

The DMA controller operates on the `l4_main_clk` input.

Related Information

Clock Manager on page 2-1

Resets

The DMA controller has nine reset signals. The reset manager drives `dma_rst_n` signal to the DMA controller on a cold or warm reset. The `dma_periph_if_rst_n[7:0]` signal resets the eight FPGA peripheral request interfaces.

Table 16-2: Reset inputs to the DMA controller

Reset Signal	Description
`dma_rst_n`	Resets DMA controller
`dma_periph_if_rst_n[7:0]`	Resets the eight FPGA peripheral request interfaces

Related Information

Reset Manager on page 3-1

Constraints and Limitations of Use

DMA Channel Arbitration

The DMAC uses a round-robin scheme to serve the active DMA channels. To ensure that the DMAC continues to serve the DMA manager, it always serves the DMA manager prior to serving the next DMA channel.

You cannot alter the arbitration process of the DMAC.

DMA Channel Prioritization

The DMAC responds to all active DMA channels with equal priority. You cannot increase the priority of a DMA channel over any other DMA channels.

Instruction Cache Latency

When a cache miss occurs, most of the delay is introduced by the memory containing the DMA code; the DMAC adds minimal delay.

AXI Data Transfer Size

The DMAC can only perform data accesses up to 64 bits in width. If you program the `src_burst_size` or `dst_burst_size` fields to be larger, the DMAC indicates a precise abort.

Related Information

Abort Sources on page 16-16

AXI Bursts Crossing 4 KB Boundaries

The AXI specification does not permit AXI bursts to cross 4 KB address boundaries. If you program the DMAC with a combination of burst start address, size, and length that would cause a single burst to cross a 4 KB address boundary, then the DMAC generates a pair of bursts with a combined length equal to that specified. This operation is transparent to the DMAC channel thread program so that, for example, the DMAC responds to a single `DMALD` instruction by generating the appropriate pair of AXI read bursts.

AXI Burst Types

You can program the DMAC to generate only fixed-address or incrementing-address burst types for data accesses. It does not generate wrapping-address bursts for data accesses or for instruction fetches.

AXI Write Addresses

The DMAC can issue up to eight outstanding write addresses. The DMAC does not issue a write address until it has read in all of the data bytes required to fulfill that write transaction.

AXI Write Data Interleaving

The DMAC does not generate interleaved write data. All write data beats for one write transaction are output before any write data beat for the next write transaction.

DMA Controller Programming Model

Instruction Syntax Conventions

The following conventions are used in assembler syntax prototype lines and their subfields:

- < >

Any item bracketed by < and > is mandatory. A description of the item and of how it is encoded in the instruction is supplied by subsequent text.

- [ ]

Any item bracketed by [and] is optional. A description of the item and of how its presence or absence is encoded in the instruction is supplied by subsequent text.

- **Spaces**

To separate items, single spaces are used for clarity. When a space is obligatory in the assembler syntax, two or more consecutive spaces are used.

Instruction Set Summary

The DMAC instructions:

- Indicate a DMA prefix, to provide a unique name-space
- Have 8-bit opcodes that might use a variable data payload of 0, 8, 16, or 32 bits
- Indicate suffixes that are consistent.

Table 16-3: Instruction Syntax Summary

Mnemonic	Instruction	DMA Manager Usage	DMA Channel Usage	Description
DMAADDH	Add Halfword	No	Yes	DMAADDH on page 16-29
DMAADNH	Add Negative Halfword	No	Yes	DMAADNH on page 16-29
DMAEND	End	Yes	Yes	DMAEND on page 16-30
DMAFLUSHP	Flush and Notify Peripheral	No	Yes	DMAFLUSHP on page 16-30

Mnemonic	Instruction	DMA Manager Usage	DMA Channel Usage	Description
DMAGO	Go	Yes	No	DMAGO on page 16-31
DMAKILL	Kill	Yes	Yes	DMAKILL on page 16-32
DMALD	Load	No	Yes	DMALD[S | B] on page 16-32
DMALDP	Load and Notify Peripheral	No	Yes	DMALDP<S | B> on page 16-33
DMALP	Loop	No	Yes	DMALP on page 16-34
DMALPEND	Loop End	No	Yes	DMALPEND[S | B] on page 16-35
DMALPFE	Loop Forever	No	Yes	DMALPFE on page 16-37
DMAMOV	Move	No	Yes	DMAMOV on page 16-37
DMANOP	No Operation	Yes	Yes	DMANOP on page 16-38
DMARMB	Read Memory Barrier	No	Yes	DMARMB on page 16-38
DMASEV	Send Event	Yes	Yes	DMASEV on page 16-39
DMAST	Store	No	Yes	DMAST[S | B] on page 16-39
DMASTP	Store and Notify Peripheral	No	Yes	DMASTP<S | B> on page 16-40
DMASTZ	Store Zero	No	Yes	DMASTZ on page 16-41
DMAWFE	Wait For Event	Yes	Yes	DMAWFE on page 16-41
DMAWFP	Wait For Peripheral	No	Yes	DMAWFP on page 16-42
DMAWMB	Write Memory Barrier	No	Yes	DMAWMB on page 16-43

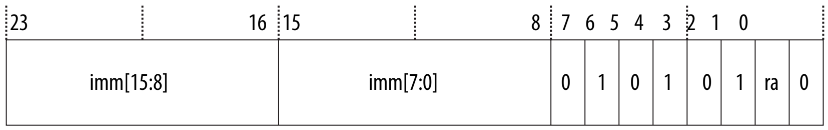
Instructions

DMAADDH

Add Halfword adds an immediate 16-bit value to the `SARn` register or `DARn` register, for the DMA channel thread. This enables the DMAC to support two-dimensional DMA operations.

Note: The immediate unsigned 16-bit value is zero-extended before the DMAC adds it to the address, using 32-bit addition. The DMAC discards the carry bit so that addresses wrap from 0xFFFFFFFF to 0x00000000.

Figure 16-5: DMAADDH Instruction Encoding



Assembler syntax

```
DMAADDH <address_register>, <16-bit bit immediate>
```

where:

`<address_register>` Selects the address register to use. It must be either:

- `SAR` `SAR`*n* register and sets `ra` to 0
- `DAR` `DAR`*n* register and sets `ra` to 1

`<16-bit immediate>` The immediate value to be added to the `<address_register>`.

Operation

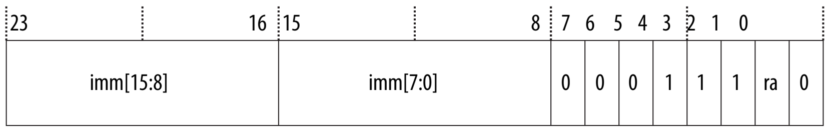
You can only use this instruction in a DMA channel thread.

DMAADNH

Add Negative Halfword adds an immediate negative 16-bit value to the `SARn` register or `DARn` register, for the DMA channel thread. This enables the DMAC to support two-dimensional DMA operations, or reading or writing an area of memory in a different order to naturally incrementing addresses.

Note: The immediate unsigned 16-bit value is one-extended to 32 bits, to create a value that is the two's complement representation of a negative number between -65536 and -1, before the DMAC adds it to the address using 32-bit addition. The DMAC discards the carry bit so that addresses wrap from 0xFFFFFFFF to 0x00000000. The net effect is to subtract between 65536 and 1 from the current value in the source or destination address register.

Figure 16-6: DMAADNH Encoding



Assembler syntax

```
DMAADNH <address_register>, <16 -bit immediate>
```

where:

`<address_register>` Selects the address register to use. It must be either:

`SAR` `SAR`*n* register and sets `ra` to 0

`DAR` `DAR`*n* register and sets `ra` to 1

`<16- bit immediate>` The immediate value to be added to the `<address_register>`.

Note: You should specify the 16-bit immediate as the number that is to be represented in the instruction encoding. For example, `DMAADNH DAR`, 0xFFF0 causes the value 0xFFFFFFF0 to be added to the current value of the Destination Address register, effectively subtracting 16 from the `DAR`.

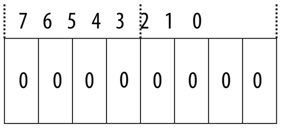
Operation

You can only use this instruction in a DMA channel thread.

DMAEND

End signals to the DMAC that the DMA sequence is complete. After all DMA transfers are complete for the DMA channel, the DMAC moves the channel to the Stopped state. It also flushes data from the MFIFO buffer and invalidates all cache entries for the thread.

Figure 16-7: DMAEND Instruction Encoding



Assembler syntax

```
DMAEND
```

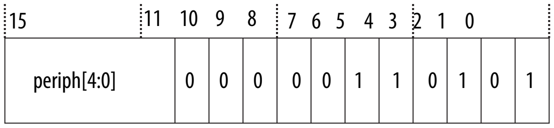
Operation

You can use the instruction with the DMA manager thread and the DMA channel thread.

DMAFLUSHP

Flush Peripheral clears the state in the DMAC that describes the contents of the peripheral and sends a message to the peripheral to resend its level status.

Figure 16-8: DMAFLUSHP Instruction Encoding



Assembler syntax

```
DMAFLUSHP <peripheral>
```

where:

`<peripheral>` 5-bit immediate, value 0-31

Operation

You can only use this instruction in a DMA channel thread.

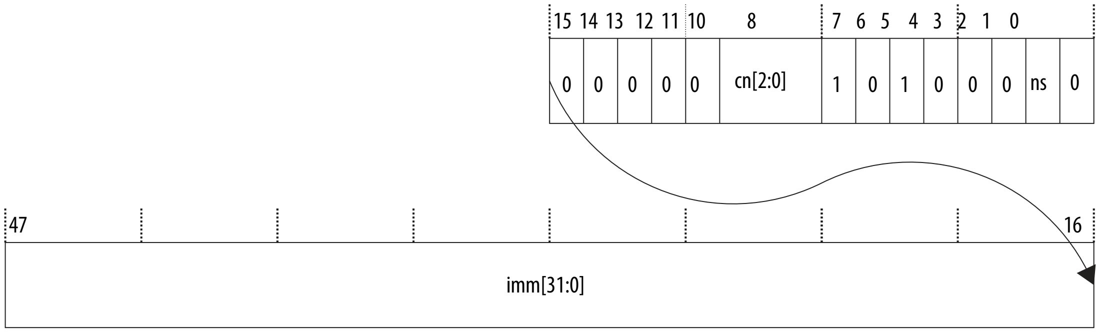
DMAGO

When the DMA manager executes Go for a DMA channel that is in the Stopped state, it performs the following steps on the DMA channel:

1. Moves a 32-bit immediate into the program counter
2. Sets its security state
3. Updates it to the Executing state.

Note: If a DMA channel is not in the Stopped state when the DMA manager executes `DMAGO` then the DMAC does not execute `DMAGO` but instead it executes `DMANOP`.

Figure 16-9: DMAGO Instruction Encoding



Assembler syntax

`DMAGO <channel_number>, <32- bit_immediate>` [, `ns`]

where:

`<channel_number>` Selects a DMA channel. It must be one of:

`C0` DMA channel 0

`C1` DMA channel 1

`C2` DMA channel 2

`C3` DMA channel 3

`C4` DMA channel 4

`C5` DMA channel 5

`C6` DMA channel 6

`C7` DMA channel 7

Note: If you provide a channel number that is not available for your configuration of the DMAC, the DMA manager thread aborts.

`<32- bit_immediate>` The immediate value that is written to the `CPC` *n* register for the selected `<channel_number>`.

`[ns]`

- If `ns` is present, the DMA channel operates in the Non-secure state.
- Otherwise, the execution of the instruction depends on the security state of the DMA manager:

DMA manager is in the Secure state—DMA channel operates in the Secure state.

DMA manager is in the Non-secure state—The DMAC aborts.

Operation

You can only use this instruction with the DMA manager thread.

DMAKILL

Kill instructs the DMAC to immediately terminate execution of a thread. Depending on the thread type, the DMAC performs the following steps:

DMA Manager Thread

1. Invalidates all cache entries for the DMA manager.
2. Moves the DMA manager to the Stopped state.

DMA Channel Thread

1. Moves the DMA channel to the Killing state.
2. Waits for AXI transactions, with an ID equal to the DMA channel number, to complete.
3. Invalidates all cache entries for the DMA channel.
4. Remove all entries in the MFIFO buffer for the DMA channel.
5. Remove all entries in the read buffer queue and write buffer queue for the DMA channel.
6. Moves the DMA channel to the Stopped state.

Figure 16-10: DMAKILL Instruction Encoding

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

Assembler syntax

`DMAKILL`

Operation

You can use the instruction with the DMA manager thread and the DMA channel thread.

Note: You must not use the `DMAKILL` instruction in DMA channel programs. To issue a `DMAKILL` instruction, use the `DBGINST0` register.

DMALD[S | B]

Load instructs the DMAC to perform a DMA load, using AXI transactions that the source address registers and channel control registers specify. It places the read data into the MFIFO buffer and tags it with the corresponding channel number. `DMALD` is an unconditional instruction but `DMALDS` and `DMALDB` are conditional on the state of the `request_type` flag. If the `src_inc` bit in the channel control registers is set to incrementing, the DMAC updates the source address registers after it executes `DMALD[S|B]`.

Note: The DMAC sets the value of `request_type` when it executes a `DMAWFP` instruction.

Figure 16-11: DMALD[S|B] Instruction Encoding

7	6	5	4	3	2	1	0
0	0	0	0	0	1	bs	x

Assembler syntax

`DMALD[S|B]`

where:

`[S]` If S is present, the assembler sets `bs` to 0 and `x` to 1. The instruction is conditional on the state of the `request_type` flag:

- `request_type` = Single

The DMAC performs a `DMALD` instruction and it sets `arlen[3:0]`=0x0 so that the AXI read transaction length is one. The DMAC ignores the value of the src_burst_len field in the channel control registers.

- `request_type` = Burst

The DMAC performs a `DMANOP` instruction. The DMAC increments the channel PC to the next instruction. No state change occurs.

`[B]` If B is present, the assembler sets `bs` to 1 and `x` to 1. The instruction is conditional on the state of the `request_type` flag:

- `request_type` = Single

The DMAC performs a `DMANOP` instruction. The DMAC increments the channel PC to the next instruction. No state change occurs.

- `request_type` = Burst

The DMAC performs a `DMALD`.

If you do not specify the S or B operand, the assembler sets `bs` to 0 and `x` to 0, and the DMAC always executes a DMA load.

Operation

You can only use this instruction in a DMA channel thread. If you specify the S or B operand, execution of the instruction is conditional on the state of `request_type` matching that of the instruction.

DMALDP<S | B>

Load and notify Peripheral instructs the DMAC to perform a DMA load, using AXI transactions that source address registers and channel control registers specify. It places the read data into a FIFO buffer that is tagged with the corresponding channel number and after it receives the last data item, it sends an acknowledgement to the peripheral that the data transfer is complete. If the `src_inc` bit in the channel control registers is set to incrementing, the DMAC updates source address registers after it executes `DMALDP<S|B>`.

Figure 16-12: DMALDP<S|B> Instruction Encoding

15 11	10	9	8	7	6	5	4	3	2	1	0
periph[4:0]	0	0	0	0	0	1	0	0	1	bs	1

Assembler syntax

`DMALDP<S|B> <peripheral>`

where:

`<S>` When S is present, the assembler sets `bs` to 0. The instruction is conditional on the state of the `request_type` flag:

- `request_type` = Single

The DMAC performs a `DMALDP` instruction and it sets `arlen[3:0]`=0x0 so that the AXI read transaction length is one. The DMAC ignores the value of the src_burst_len field in the channel control registers.

- `request_type` = Burst

The DMAC performs a `DMANOP`.

`<B>` When B is present, the assembler sets `bs` to 1. The instruction is conditional on the state of the `request_type` flag:

- `request_type` = Single

The DMAC performs a `DMANOP`.

- `request_type` = Burst

The DMAC performs a load using a burst DMA transfer.

`<peripheral>` 5-bit immediate, value 0-31.

Note: The DMAC sets the value of the `request_type` flag when it executes a `DMAWFP` instruction.

Operation

You can only use this instruction in a DMA channel thread. Execution of the instruction is conditional on the state of the `request_type` flag matching that of the instruction.

DMALP

Loop instructs the DMAC to load an 8-bit value into the loop counter register you specify.

This instruction indicates the start of a section of instructions, and you set the end of the section using the `DMALPEND` instruction. The DMAC repeats the set of instructions that you insert between `DMALP` and `DMALPEND` until the value in the loop counter register reaches zero.

Note: The DMAC saves the value of the PC for the instruction that follows `DMALP`. After the DMAC executes `DMALPEND`, and the loop counter register is not zero, this enables it to execute the first instruction in the loop.

Figure 16-13: DMALP Instruction Encoding

15 … 8	7	6	5	4	3	2	1	0
iter[7:0]	0	0	1	0	0	0	1c	0

Assembler syntax

`DMALP <loop_iterations>`

where:

`<loop_iterations>`

Specifies the number of loops to perform, range 1-256.

- The assembler determines the loop counter register to use and either:
- Sets `lc` to 0, and the DMAC writes the value loop_iterations minus 1 to the loop counter 0 registers
- Sets `lc` to 1, and the DMAC writes the value loop_iterations minus 1 to the loop counter 1 registers.

Operation

You can only use this instruction in a DMA channel thread.

DMALPEND[S | B]

Loop End indicates the last instruction in the program loop but the behavior of the DMAC depends on whether `DMALP` or `DMALPFE` starts the loop. If a loop starts with:

- `DMALP` The loop has a defined loop count and `DMALPEND[S|B]` instructs the DMAC to read the value of the loop counter register. If a loop counter register returns:
  - Zero—The DMAC executes a `DMANOP` and therefore exits the loop.
  - Nonzero—The DMAC decrements the value in the loop counter register and updates the thread PC to contain the address of the first instruction in the program loop, that is, the instruction that follows the `DMALP`.
- `DMALPFE` The loop has an undefined loop count and the DMAC uses the state of the `request_last` flag to control when it exits the loop. If the `request_last` flag is:
  - 0—The DMAC updates the thread PC to contain the address of the first instruction in the program loop, that is, the instruction that follows the `DMALP`.
  - 1—The DMAC executes a `DMANOP` and therefore exits the loop.

Figure 16-14: DMALPEND[S|B] Instruction Encoding

15 … 8	7	6	5	4	3	2	1	0
backwards_jump[7:0]	0	0	1	nf	1	lc	bs	x

Assembler syntax

`DMALPEND[S|B]`

where:

[S] If S is present and the loop starts with DMALP, then the assembler sets bs to 0 and x to 1. The instruction is conditional on the state of the request_type flag:

- request_type = Single
  - The DMAC executes the DMALPEND
- request_type = Burst
  - The DMAC performs a DMANOP and therefore exits the loop.

[B] If B is present and the loop starts with DMALP, then the assembler sets bs to 1 and x to 1. The instruction is conditional on the state of the request_type flag:

- request_type = Single
  - The DMAC performs a DMANOP and therefore exits the loop.
- request_type = Burst
  - The DMAC executes the DMALPEND

If you do not specify the S or B operand, the assembler sets bs to 0 and x to 0, and the DMAC always executes the DMALPEND.

Note: You must not specify the S or B operand when a loop starts with DMALPFE. If you do, the assembler issues a warning message and sets bs to 0, x to 0, and nf to 1. In the same way as for DMALPFE, the DMAC uses the state of the request_last flag to control when it exits the loop.

Note: The DMAC sets the value of the:

- request_type flag when it executes a DMAWFP instruction.
- request_last flag to 1 when the corresponding peripheral issues the last request command through the peripheral request interface.

To correctly assign the additional bits in the DMALPEND instruction, the assembler determines the values for:

backwards_jump[7:0] Sets the relative location of the first instruction in the program loop. The assembler calculates the value for backwards_jump[7:0] by subtracting the address of the first instruction in the loop from the address of the DMALPEND.

- nf sets it to:
  - 0 if DMALPFE started the program loop
  - 1 if DMALP started the program loop.
- lc sets it to:
  - 0 if the loop counter 0 registers contains the loop counter value
  - 1 if the loop counter 1 registers contains the loop counter value
  - 1 if DMALPFE started the program loop.

Operation

You can only use this instruction in a DMA channel thread. If you specify the S or B operand, execution of the instruction is conditional on the state of the request_type flag matching that of the instruction.

Related Information

- Peripheral Length Management on page 16-11

- **DMAWFP** on page 16-42

DMALPFE

The assembler uses Loop Forever to configure certain bits in DMALPEND.

Note: When the assembler encounters DMALPFE, it does not create an instruction for the DMAC, but instead, it modifies the behavior of DMALPEND. The insertion of DMALPFE in program code identifies the start of the loop.

Assembler syntax

```
DMALPFE
```

Related Information

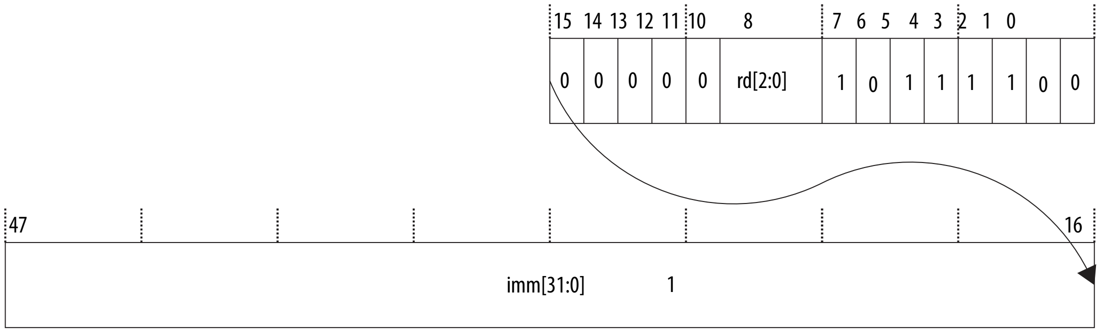
DMALPEND[S | B] on page 16-35

DMAMOV

Move instructs the DMAC to move a 32-bit immediate into the following registers:

- source address registers
- destination address registers
- channel control registers

Figure 16-15: DMAMOV Instruction Encoding



Assembler syntax

```
DMAMOV <destination_register>, <32- bit_immediate>
```

where:

```
<destination_register>
```

The valid registers are:

- SAR—selects the source address registers and sets rd to b000
- CCR—selects the channel control registers and sets rd to b001
- DAR—selects the destination address registers and sets rd to b010

```
<32 -bit_immediate>
```

A 32-bit value that is written to the specified destination register.

Note: For information about using the assembler to program the various fields that the channel control registers, refer to DMAMOV CCR section

Operation

You can only use this instruction in a DMA channel thread.

Related Information

DMAMOV CCR on page 16-44
Information about using the assembler to program the various fields that the channel control registers

DMANOP

No Operation does nothing. You can use this instruction for code alignment purposes.

Figure 16-16: DMANOP Instruction Encoding

7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0

Assembler syntax

```
DMANOP
```

Operation

You can use the instruction with the DMA manager thread and the DMA channel thread.

DMARMB

Read Memory Barrier forces the DMA channel to wait until all of the executed `DMALD` instructions for that channel have been issued on the AXI master interface and have completed.

This enables write-after-read sequences to the same address location with no hazards.

Figure 16-17: DMARMB Instruction Encoding

7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	0

Assembler syntax

```
DMARMB
```

Operation

You can only use this instruction in a DMA channel thread.

DMASEV

Send Event instructs the DMAC to modify an event-interrupt resource. Depending on how you program the interrupt enable register, this either:

- Generates event <event_num>

  Note: Typically, you use `DMAWFE` to stall a thread and then another thread executes `DMASEV`, using the appropriate event number, to unstall the waiting thread.
- Signals an interrupt using `irq <event_num>`.

Figure 16-18: DMASEV Instruction Encoding

15 … 11	10	9	8	7	6	5	4	3	2	1	0
event_num[4:0]	0	0	0	0	0	1	1	0	1	0	0

Assembler syntax

```
DMASEV <event_num>
```

where:

`<event_num>` 5-bit immediate, value 0-31

Note: The DMAC aborts the thread if you select an event number that is not available.

Operation

You can use the instruction with the DMA manager thread and the DMA channel thread.

Related Information

- **Using Events and Interrupts** on page 16-14
- **Using an Event to Restart DMA Channels** on page 16-14

DMAST[S | B]

Store instructs the DMAC to transfer data from the FIFO buffer to the location that the destination address registers specifies, using AXI transactions that the DA register and channel control registers specify. If the `dst_inc` bit in the channel control registers is set to incrementing, the DMAC updates the destination address registers after it executes `DMAST[S|B]`.

Figure 16-19: DMAST[S|B] Instruction Encoding

7	6	5	4	3	2	1	0
0	0	0	0	1	0	bs	x

Assembler syntax

```
DMAST[S|B]
```

where:

[S] If S is present, the assembler sets bs to 0 and x to 1. The instruction is conditional on the state of the request_type flag:

- request_type = Single
- The DMAC performs a DMAST instruction and it sets awlen[3:0]=0x0 so that the AXI write transaction length is one. The DMAC ignores the v value of the dst_burst_len field in the channel control registers.
- request_type = Burst
- The DMAC performs a DMANOP instruction. The DMAC increments the channel PC to the next instruction. No state change occurs.

[B] If B is present, the assembler sets bs to 1 and x to 1. The instruction is conditional on the state of the request_type flag:

- request_type = Single
- The DMAC performs a DMANOP instruction. The DMAC increments the channel PC to the next instruction. No state change occurs.
- request_type = Burst
- The DMAC performs a DMAST.
- If you do not specify the S or B operand, the assembler sets bs to 0 and x to 0, and the DMAC always executes a DMA store.

Note: The DMAC sets the value of the request_type flag when it executes a DMAWFP instruction.

Operation

You can only use this instruction in a DMA channel thread. If you specify the S or B operand, execution of the instruction is conditional on the state of the request_type flag matching that of the instruction.

The DMAC only commences the burst when the MFIFO buffer contains all of the data necessary to complete the burst transfer.

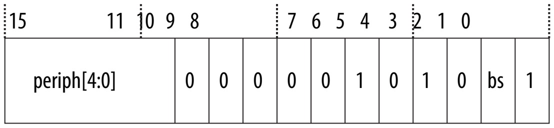
Related Information

DMAWFP on page 16-42

DMASTP<S | B>

Store and notify Peripheral instructs the DMAC to transfer data from the FIFO buffer to the location that the destination address registers specifies, using AXI transactions that the DA register and channel control registers specify. It uses the DMA channel number to access the appropriate location in the FIFO buffer. After the DMA store is complete, and the DMAC has received a buffered write response, it issues an acknowledgement to the peripheral that the data transfer is complete. If the dst_inc bit in the channel control registers is set to incrementing, the DMAC updates the destination address registers after it executes DMASTP<S|B>.

Figure 16-20: DMASTP<S|B> Instruction Encoding



Assembler syntax

```
DMASTP<S|B> <peripheral>
```

where:

`<S>` Sets `bs` to 0. This instructs the DMAC to perform:

- A single DMA store operation if `request_type` is programmed to Single
- The DMAC ignores the state of the dst_burst_len field in the channel control registers and always performs an AXI transfer with a burst length of one.
- A `DMANOP` if `request_type` is programmed to Burst.

`<B>` Sets `bs` to 1. This instructs the DMAC to perform:

- The DMA store if `request_type` is programmed to Burst
- A `DMANOP` if `request_type` is programmed to Single.

`<peripheral>` 5-bit immediate, value 0-31.

Note: The DMAC sets the value of the `request_type` flag when it executes a `DMAWFP` instruction.

Operation

You can only use this instruction in a DMA channel thread.

The DMAC only commences the burst when the MFIFO buffer contains all of the data necessary to complete the burst transfer.

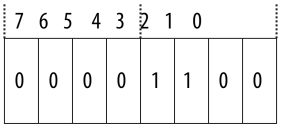
Related Information

DMAWFP on page 16-42

DMASTZ

Store Zero instructs the DMAC to store zeros, using AXI transactions that the destination address registers and channel control registers specify. If the `dst_inc` bit in the channel control registers is set to incrementing, the DMAC updates the destination address registers after it executes `DMASTZ`.

Figure 16-21: DMASTZ Instruction Encoding



Assembler syntax

```
DMASTZ
```

Operation

You can only use this instruction in a DMA channel thread.

DMAWFE

Wait For Event instructs the DMAC to halt execution of the thread until the event, that <event_num> specifies, occurs. When the event occurs, the thread moves to the Executing state and the DMAC clears the event.

Figure 16-22: DMAWFE Instruction Encoding

15	11	10	9	8	7	6	5	4	3	2	1	0
event_num[4:0]		0	i	0	0	0	1	1	0	1	1	0

Assembler syntax

`DMAWFE <event_num>[, invalid]`

where:

`<event_num>` 5-bit immediate, value 0-31

`[invalid]` Sets `i` to 1. If `invalid` is present, the DMAC invalidates the instruction cache for the current DMA thread. If `invalid` is not present, then the assembler sets `i` to 0 and the DMAC does not invalidate the instruction cache for the current DMA.

- The DMAC aborts the thread if you select an event number that is not available for your configuration of the DMAC. To ensure cache coherency, you must use `invalid` when a processor writes the instruction stream for a DMA channel.

Operation

You can use the instruction with the DMA manager thread and the DMA channel thread.

Related Information

Using Events and Interrupts on page 16-14

DMAWFP

Wait For Peripheral instructs the DMAC to halt execution of the thread until the specified peripheral signals a DMA request for that DMA channel.

Figure 16-23: DMAWFP Instruction Encoding

15	11	10	9	8	7	6	5	4	3	2	1	0
peripheral[4:0]		0	0	0	0	0	1	1	0	0	bs	p

Assembler syntax

`DMAWFP <peripheral>, <single|burst|periph>`

where:

`<peripheral>` 5-bit immediate, value 0-31

Note: The DMAC aborts the thread if you select a peripheral number that is not available.

`<single>` Sets `bs` to 0 and `p` to 0. This instructs the DMAC to continue executing the DMA channel thread after it receives a single or burst DMA request. The DMAC sets the `request_type` to Single, for that DMA channel.

`<burst>` Sets `bs` to 1 and `p` to 0. This instructs the DMAC to continue executing the DMA channel thread after it receives a burst DMA request. The DMAC sets the `request_type` to Burst.

Note: The DMAC ignores single burst DMA requests.

`<periph>` Sets `bs` to 0 and `p` to 1. This instructs the DMAC to continue executing the DMA channel thread after it receives a single or burst DMA request. The DMAC sets the `request_type` to:

- **Single:** When it receives a single DMA request.
- **Burst:** When it receives a burst DMA request.

Operation

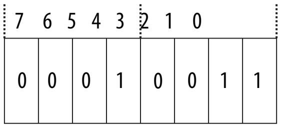
You can only use this instruction in a DMA channel thread.

DMAWMB

Write Memory Barrier forces the DMA channel to wait until all of the executed `DMAST` instructions for that channel have been issued on the AXI master interface and have completed.

This permits read-after-write sequences to the same address location with no hazards.

Figure 16-24: DMAWMB Instruction Encoding



7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	1

Assembler syntax

`DMAWMB`

Operation

You can only use this instruction in a DMA channel thread.

Assembler Directives

The assembler provides several additional commands.

DCD

Assembler directive to place a 32-bit immediate in the instruction stream.

Syntax

DCD imm32

DCB

Assembler directive to place an 8-bit immediate in the instruction stream.

Syntax

DCB imm8

DMALP

Assembler directive to insert an iterative loop.

Syntax

DMALP [<LC0>|<LC1>] `<loop_iterations>`

where:

`<loop_iterations>`

An 8-bit value that specifies the number of loops to perform.

Note: For clarity in writing assembler instructions, the 8-bit value is the actual number of iterations of the loop to be executed. The assembler decrements this by one to create the actual value, 0-255, that the DMAC uses.

`[LC0]` If `LC0` is present, the DMAC stores `<loop_iterations>` in the loop counter 0 registers.

`[LC1]` If `LC1` is present, the DMAC stores `<loop_iterations>` in the loop counter 1 registers.

Note: If `LC0` or `LC1` is not present, the assembler determines the loop counter register to use.

DMALPFE

Assembler directive to insert a repetitive loop.

Syntax

DMALPFE

Enables the assembler to clear the `nf` bit that is present in `DMALPEND[S | B]`.

DMAMOV CCR

Assembler directive that enables you to program the channel control registers using the specified format.

Syntax

DMAMOV CCR,

```
[SB<1-16>] [SS<8|16|32|64|128>] [SA<I|F>]

[SP<imm3>] [SC<imm4>]

[DB<1-16>] [DS<8|16|32|64|128>] [DA<I|F>]

[DP<imm3>] [DC<imm4>]

[ES<8|16|32|64|128>]
```

Table 16-4: DMAMOV CCR argument description and the default values

Syntax	Description	Options	Default

SA	Source address increment. Sets the value of `arburst[0]`	I = Increment
		F = Fixed
			I

SS	Source burst size in bits. Sets the value of `arsize[2:0]`	8, 16, 32, or 64	8

Syntax	Description	Options	Default
SB	Source burst length. Sets the value of `arlen[3:0]`	1 to 16	1
SP	Source protection	0 to 7[1]	0
SC	Source cache	0 to 15[1,2]	0

DA	Destination address increment. Sets the value of `awburst[0]`	I = Increment
		F = Fixed
			I

DS	Destination burst size in bits. Sets the value of `awsize[2:0]`	8, 16, 32, or 64	8
DB	Destination burst length. Sets the value of `awlen[3:0]`	1 to 16	1
DP	Destination protection	0 to 7[1]	0
DC	Destination cache	0 to 15[1,3]	0
ES	Endian swap size, in bits	8, 16, 32, or 64	8

Note: **1.** You must use decimal values when programming this immediate value.
2. Because the DMAC ties ARCACHE[3] LOW, the assembler always sets bit 3 to 0 and uses bits [2:0] of your chosen value for SC.
3. Because the DMAC ties AWCACHE[2] LOW, the assembler always sets bit 2 to 0 and uses bit [3] and bits [1:0] of your chosen value for DC.

MFIFO Buffer Usage Overview

About MFIFO Buffer Usage Overview

The MFIFO buffer is a shared resource that is utilized on a first-come, first-served basis by all currently active channels. To a program, it appears as a set of variable-depth parallel FIFO buffers, one per channel, with the restriction that the total depth of all the Fifes cannot exceed the buffer depth, 512. The width of the AXI master interface is the same as the MFIFO buffer width.

The DMAC is capable of realigning data from the source to the destination. For example, the DMAC shifts the data by two byte lanes when it reads a word from address 0x103 and writes to address 0x205. All byte manipulations occur when data enters the MFIFO buffer, as a result of an AXI read due to a `DMALD` instruction, so that the DMAC does not need to manipulate the data when it removes it from the MFIFO buffer, as a result of an AXI write due to a `DMAST` instruction. Therefore the storage and packing of the data in the MFIFO buffer is determined by the destination address and transfer characteristics.

When a program specifies that incrementing transactions are to be performed to the destination, the DMAC packs data into the MFIFO buffer to minimize the usage of the MFIFO buffer entries. For example, the DMAC packs two 32-bit words into a single entry in the MFIFO buffer when the DMAC has a 64-bit AXI data bus and the program uses a source address of 0x100, and destination address of 0x200.

In certain situations, the number of entries required to store the data loaded from a source is not a simple calculation of amount of source data divided by MFIFO buffer width. The calculation of the number of entries required is not simple when any of the following occur:

- The source address is not aligned to the AXI bus width.
- The destination address is not aligned to the AXI bus width.
- The transactions are to a fixed destination, that is, a non-incrementing address.

The DMALD and DMAST instructions each specify that an AXI transaction is to be performed. The amount of data transferred by an AXI transaction depends on the values programmed in to the CCR *n* register and the address of the transaction.

The following sections provide several example DMAC programs together with illustrations of the MFIFO buffer usage.

Note: These sections show MFIFO buffer usage in the following ways:

- A graph of the number of MFIFO buffer entries versus time
- A diagram of the byte-lane manipulation that the DMAC performs when data enters the MFIFO buffer.

Note: The numbers 0 and 7 in the MFIFO buffer diagrams indicate the byte lanes in the MFIFO buffer.

Related Information

http://infocenter.arm.com/

Information about unaligned transfers available from the ARM info center website.

Aligned Transfers

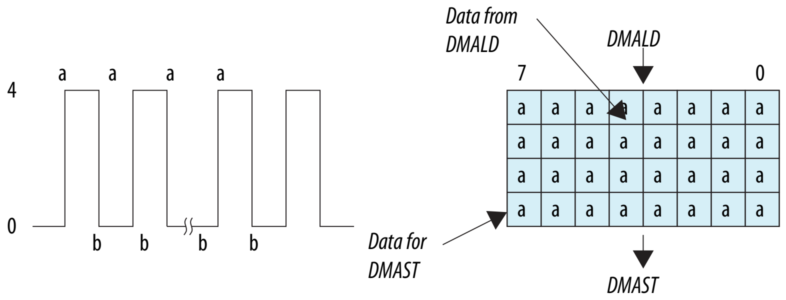
Simple Aligned Program

The following program shows the MFIFO buffer usage. In this program, the source address and destination address are aligned with the AXI data bus width.

```
DMAMOV CCR, SB4 SS64 DB4 DS64
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4000
DMALP 16
    DMALD shown as a in figure below
DMALPEND
DMAEND
```

Figure 16-25: Simple Aligned Program

Each DMALD requires four entries and each DMAST removes four entries.



This example has a static requirement of zero MFIFO buffer entries and a dynamic requirement of four MFIFO buffer entries.

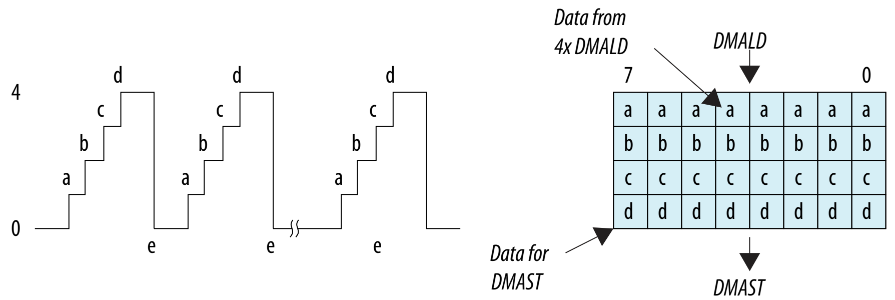
Aligned Asymmetric Program with Multiple Loads

The following program performs four loads for each store and the source address and destination address are aligned with the AXI data bus width.

```
DMAMOV CCR, SB1 SS64 DB4 DS64
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4000
DMALP 16
    DMALD ; shown as a in the figure below
    DMALD ; shown as b in the figure below
    DMALD ; shown as c in the figure below
    DMALD ; shown as d in the figure below
    DMAST ; shown as e in the figure below
DMALPEND
DMAEND
```

Figure 16-26: Aligned Asymmetric Program with Multiple Loads

Each `DMALD` requires one entry and each `DMAST` removes four entries.



This example has a static requirement of zero MFIFO buffer entries and a dynamic requirement of four MFIFO buffer entries.

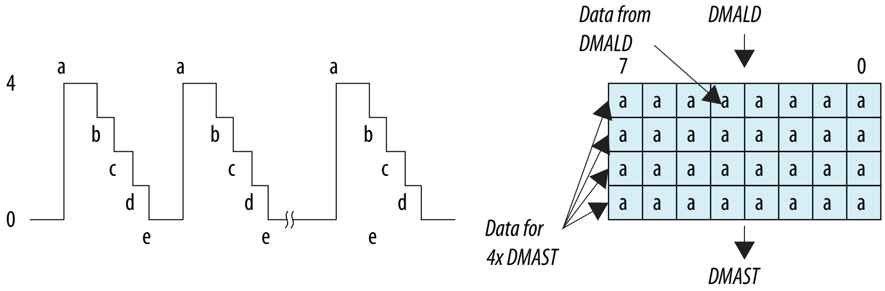
Aligned Asymmetric Program with Multiple Stores

The following program performs four stores for each load and the source address and destination address are aligned with the AXI data bus width.

```
DMAMOV CCR, SB4 SS64 DB1 DS64
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4000
DMALP 16
    DMALD ; shown as a in the figure below
    DMAST ; shown as b in the figure below
    DMAST ; shown as c in the figure below
    DMAST ; shown as d in the figure below
    DMAST ; shown as e in the figure below
DMALPEND
DMAEND
```

Figure 16-27: Aligned Asymmetric Program with Multiple Stores

Each DMALD requires four entries and each DMAST removes one entry.



This example has a static requirement of zero MFIFO buffer entries and a dynamic requirement of four MFIFO buffer entries.

Unaligned Transfers

Aligned Source Address to Unaligned Destination Address

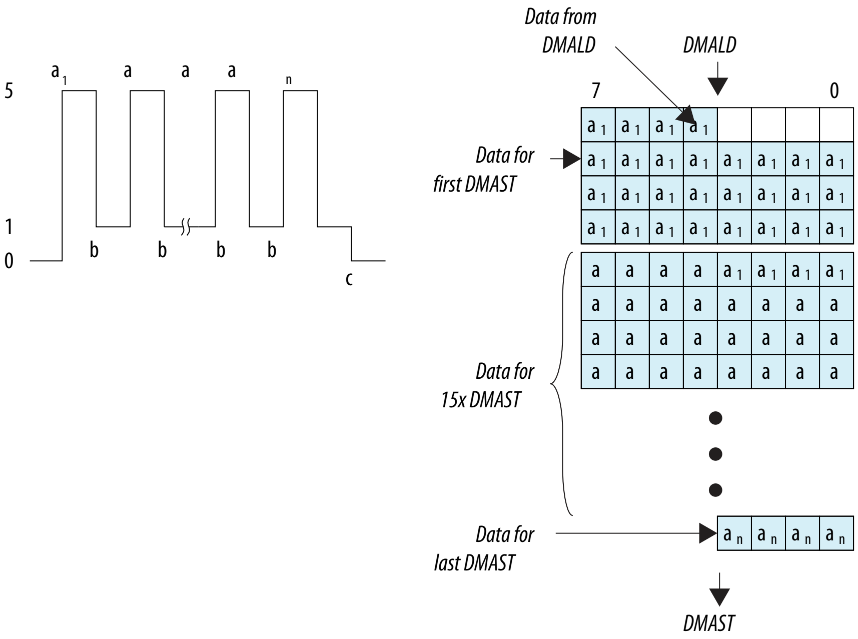
In the following program, the source address is aligned with the AXI data bus width but the destination address is unaligned. The destination address is not aligned to the destination burst size so the first DMAST instruction removes less data than the first DMALD instruction reads. Therefore, a final DMAST of a single word is required to clear the data from the MFIFO buffer.

```
DMAMOV CCR, SB4 SS64 DB4 DS64
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4004
DMALP 16
    DMALD ; shown as a1, ... a, an in the figure below
    DMAST ; shown as b in the figure below
DMALPEND
DMAMOV CCR, SB4 SS64 DB1 DS32
DMAST ; shown as c in the figure below
DMAEND
```

Figure 16-28: Aligned to Unaligned Program

The first DMALD instruction loads four doublewords but because the destination address is unaligned, the DMAC shifts them by four bytes and therefore it uses five entries in the MFIFO buffer.

Each DMAST requires only four entries of data, and therefore the extra entry remains in use for the duration of the program, until it is emptied by the last DMAST.



This example has a static requirement of one MFIFO buffer entry and a dynamic requirement of four MFIFO buffer entries.

Unaligned Source Address to Aligned Destination Address

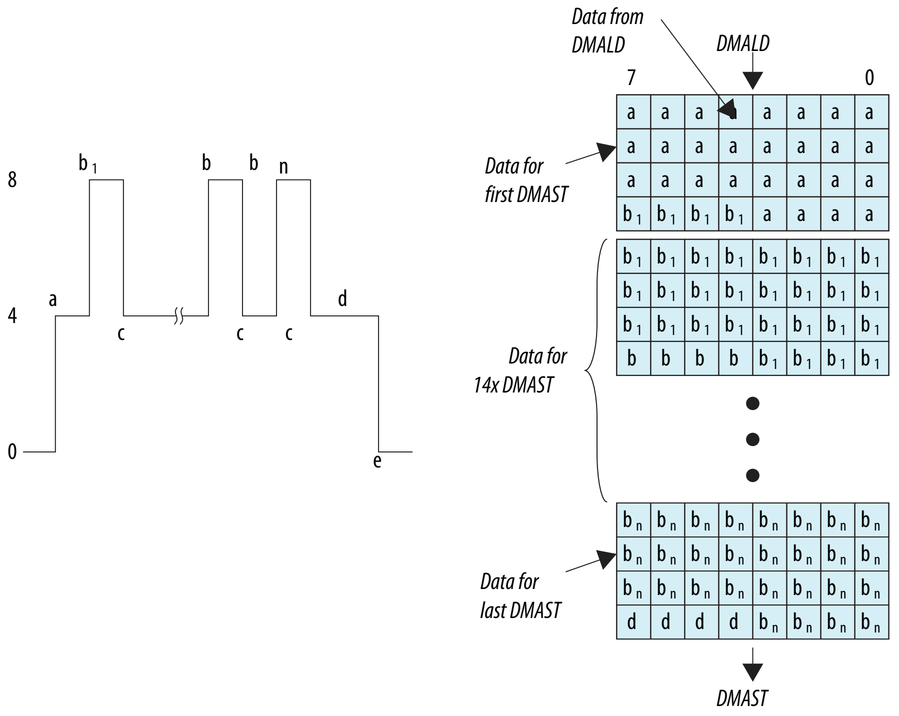
In this program, the source address is unaligned with the AXI data bus width but the destination address is aligned. The source address is not aligned to the source burst size so the first DMALD instruction reads in less data than the DMAST. Therefore, an extra DMALD is required to satisfy the first DMAST.

```
DMAMOV CCR, SB4 SS64 DB4 DS64
DMAMOV SAR, 0x1004
DMAMOV DAR, 0x4000
DMALD ; shown as a in the figure below
DMALP 15
    DMALD ; shown as b1, ... b, bn in the figure below
    DMAST ; shown as c in the figure below
DMALPEND
DMAMOV CCR, SB1 SS32 DB4 DS64
DMALD ; shown as d in the figure below
DMAST ; shown as e in the figure below
DMAEND
```

Figure 16-29: Unaligned to Aligned Program

The first DMALD instruction does not load sufficient data to enable the DMAC to execute a DMAST and therefore the program includes an additional DMALD, prior to the start of the loop.

After the first DMALD, the subsequent DMALDs align with the source burst size. This optimizes the p performance but it requires a larger number of MFIFO buffer entries.



Note: The DMALD shown as **d** does not increase the MFIFO buffer usage because it loads four bytes into an MFIFO buffer entry that the DMAC has already allocated to this channel.

This example has a static requirement of four MFIFO buffer entries and a dynamic requirement of four MFIFO buffer entries.

Unaligned Source Address to Aligned Destination Address with Excess Initial Load

This program is an alternative to that described in *Unaligned Source Address to Aligned Destination Address*. The program executes a different sequence of source bursts which might be less efficient, but require fewer MFIFO buffer entries.

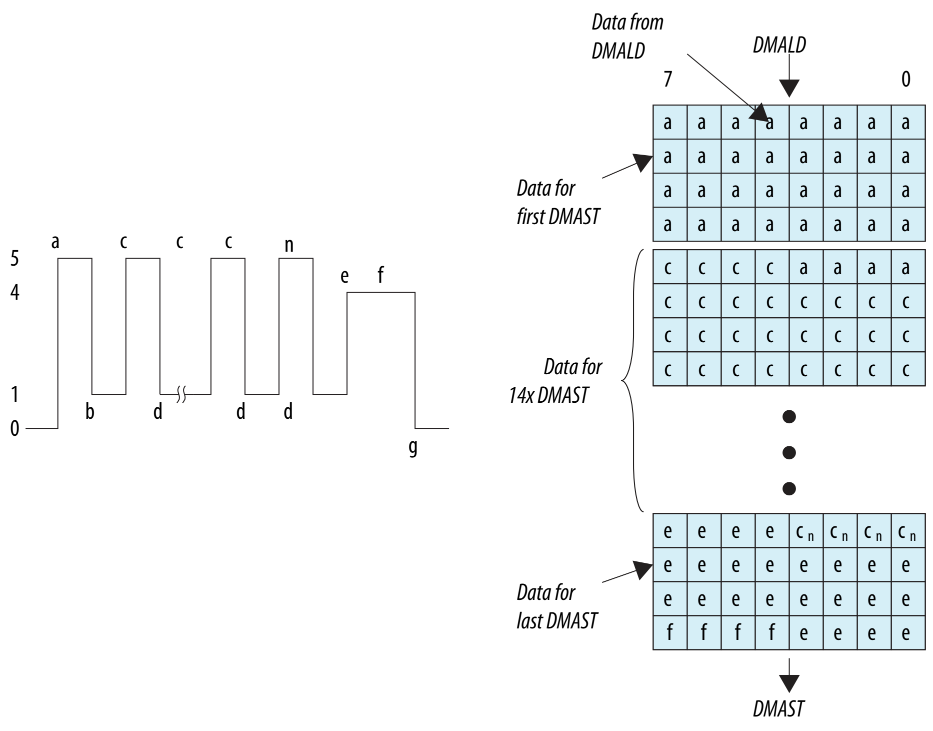
```
DMAMOV CCR, SB5 SS64 DB4 DS64
DMAMOV SAR, 0x1004
DMAMOV DAR, 0x4000
DMALD ; shown as a in the figure below
DMAST ; shown as b in the figure below
DMAMOV CCR, SB4 SS64 DB4 DS64
DMALP 14
    DMALD ; shown as c and cn in the figure below
DMALPEND
DMAMOV CCR, SB3 SS64 DB4 DS64
DMALD ; shown as e in the figure below
DMAMOV CCR, SB1 SS32 DB4 DS64
DMALD ; shown as f in the figure below
```

```
DMAST ; shown as g in the figure below
DMAEND
```

Figure 16-30: Unaligned to Aligned with Excess Initial Load

The first DMALD instruction loads five bursts of data to enable the DMAC to execute the first DMAST.

After the first DMALD, the subsequent DMALDs are not aligned to the source burst size, for example the second DMALD reads from address 0x1028. After the loop, the final two DMALDs read the data required to satisfy the final DMAST.



Note: The DMALD shown as **f** does not increase the MFIFO buffer usage because it loads four bytes into an MFIFO buffer entry that the DMAC has already allocated to this channel.

This example has a static requirement of one MFIFO buffer entry and a dynamic requirement of four MFIFO buffer entries.

Related Information

Unaligned Source Address to Aligned Destination Address on page 16-49

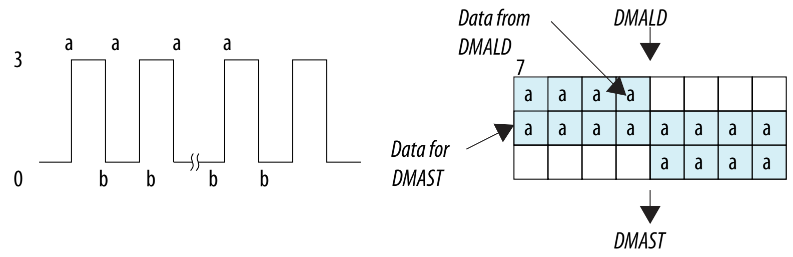
Aligned Burst Size Unaligned MFIFO Buffer

In this program, the destination address, which is narrower than the MFIFO buffer width, aligns with the burst size, but does not align with the MFIFO buffer width.

```
DMAMOV CCR, SB4 SS32 DB4 DS32
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4004
DMALP 16
 DMALD ; shown as a in the figure below
 DMAST ; shown as b in the figure below
```

```
DMALPEND
DMAEND
```

Figure 16-31: Aligned Burst with Unaligned MFIFO Buffer Width



In this example, the destination address is not 64-bit aligned, it requires three rather than the expected two MFIFO buffer entries.

This example has a static requirement of zero MFIFO buffer entries and a dynamic requirement of three MFIFO buffer entries.

Fixed Transfers

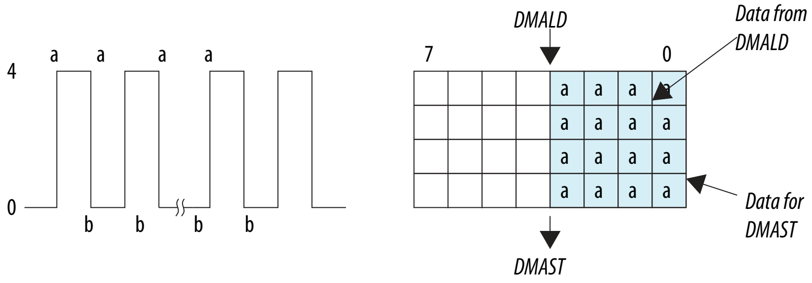
Fixed Destination with Aligned Address

In this program, the source address and destination address are aligned with the AXI data bus width, and the destination address is fixed.

```
DMAMOV CCR, SB2 SS64 DB4 DS32 DAF
DMAMOV SAR, 0x1000
DMAMOV DAR, 0x4000
DMALP 16
    DMALD ; shown as a in the figure below
    DMAST ; shown as b in the figure below
DMALPEND
DMAEND
```

Figure 16-32: Fixed Destination with Aligned Address

Each DMALD in the program loads two 64-bit data transfers into the MFIFO buffer. Because the destination address is a 32-bit fixed address, then the DMAC splits each 64-bit data item across two entries in the MFIFO buffer.

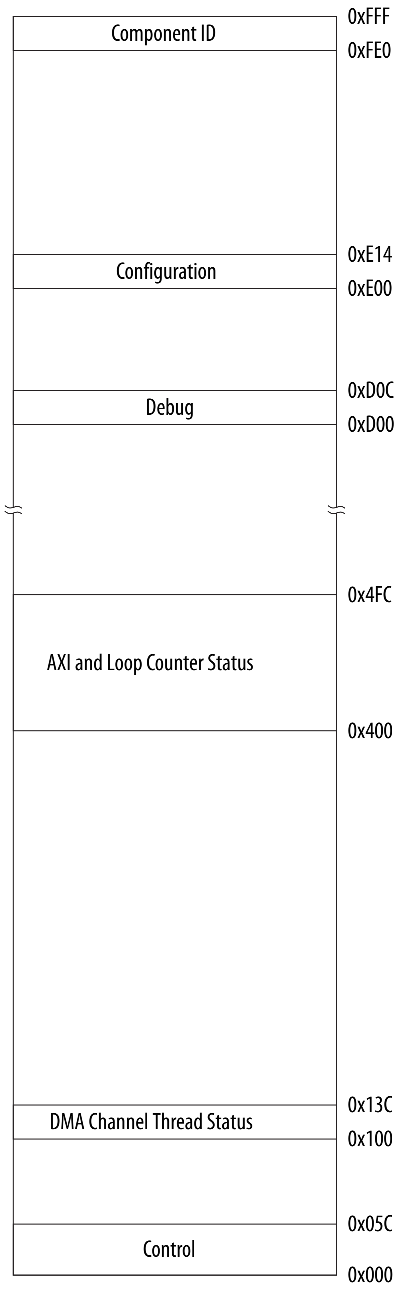


This example has a static requirement of zero MFIFO buffer entries and a dynamic requirement of four MFIFO buffer entries.

DMA Controller Registers

The following DMAC register map spans a 4 KB region, consists of the following sections.

Figure 16-33: DMAC Summary Register Map



- **Control registers**— allow you to control the DMAC.
- **DMA channel thread status registers**—provide the status of the DMA channel threads.
- **AXI and loop counter status registers**—provide the AXI transfer status and the loop counter status, for each DMA channel thread.
- **Debug registers**—enable the following functionality:
    - Allow you to send instructions to a thread when debugging the program code.
    - Allow system firmware to send instructions to the DMA manager thread.
- **Configuration registers**—enable system firmware to identify the configuration of the DMAC and control the behavior of the watchdog.
- **Component ID registers**— enable system firmware to identify peripherals. Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in an unpredictable behavior.

Related Information

- **Issuing Instructions to the DMAC using a Slave Interface** on page 16-8

Address Map and Register Definitions

The address map and register definitions for the DMA Controller consist of the following regions:

- Nonsecure DMA Module Address Space
- Secure DMA Module Address Space

Related Information

- **Non-Secure DMA Module Address Map** on page 16-56

This address space is allocated for non-secure DMA accesses. For detailed information about the use of this address space, **click here** to access the ARM documentation for the DMA-330.

- **Secure DMA Module Address Map** on page 16-54

This address space is allocated for secure DMA accesses. For detailed information about the use of this address space, **click here** to access the ARM documentation for the DMA-330.

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor* chapter.

Secure DMA Module Address Map

This address space is allocated for secure DMA accesses. For detailed information about the use of this address space, **click here** to access the ARM documentation for the DMA-330.

Table 16-5: Secure DMA Module Address Range

Module Instance	Start Address	End Address
secure_dma	0xFFE01000	0xFFE01FFF

Table 16-6: Secure DMA Register Space

Register Group	Description	Start Address	End Address
Control	This address space is allocated for DMA control registers.	0xFFE01000	0xFFE0105F

Register Group	Description	Start Address	End Address
Reserved	This address space is reserved.	0xFFE01060	0xFFE010FF
DMA Channel Thread Status	This address space is allocated for the registers that provide the status of the DMA channel threads.	0xFFE01100	0xFFE0113F
Reserved	This address space is reserved.	0xFFE01140	0xFFE013FF
AXI and Loop Counter Status	This address space is allocated for the registers that provide the AXI transfer status and loop counter status for each DMA channel thread.	0xFFE01400	0xFFE014F0
Reserved	This address space is reserved.	0xFFE014F4	0xFFE01CFF
Debug	This address space is allocated for the debug registers.	0xFFE01D00	0xFFE01D0F
Reserved	This address space is reserved.	0xFFE01D10	0xFFE01DFF
Configuration	This address space holds registers that can be read for configuration data and control watchdog behavior.	0xFFE01E00	0xFFE01E80
Reserved	This address space is reserved.	0xFFE01E84	0xFFE01FDF
Component ID	This address space holds peripheral ID information.	0xFFE01FE0	0xFFE01FFF

Non-Secure DMA Module Address Map

This address space is allocated for non-secure DMA accesses. For detailed information about the use of this address space, **click here** to access the ARM documentation for the DMA-330.

Table 16-7: Non-secure DMA Module Address Range

Module Instance	Start Address	End Address
non-secure_dma	0xFFE00000	0xFFE00FFF

Table 16-8: Non-secure DMA Register Space

Register Group	Description	Start Address	End Address
Control	This address space is allocated for DMA control registers.	0xFFE00000	0xFFE0005F
Reserved	This address space is reserved.	0xFFE00060	0xFFE000FF
DMA Channel Thread Status	This address space is allocated for the registers that provide the status of the DMA channel threads.	0xFFE00100	0xFFE0013F
Reserved	This address space is reserved.	0xFFE00140	0xFFE003FF
AXI and Loop Counter Status	This address space is allocated for the registers that provide the AXI transfer status and loop counter status for each DMA channel thread.	0xFFE00400	0xFFE004F0
Reserved	This address space is reserved.	0xFFE004F4	0xFFE00CFF
Debug	This address space is allocated for the debug registers.	0xFFE00D00	0xFFE00D0F
Reserved	This address space is reserved.	0xFFE00D10	0xFFE00DFF

Register Group	Description	Start Address	End Address
`Configuration`	This address space holds registers that can be read for configuration data and control watchdog behavior.	`0xFFE00E00`	`0xFFE00E80`
`Reserved`	This address space is reserved.	`0xFFE00E84`	`0xFFE00FDF`
`Component ID`	This address space holds peripheral ID information.	`0xFFE00FE0`	`0xFFE00FFF`

Document Revision History

Table 16-9: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address maps and register definitions
February 2014	2014.02.28	ECC updates
December 2013	1.2	Maintenance release
November 2012	1.1	Minor updates
January 2012	1.0	Initial release

Ethernet Media Access Controller 17

2014.06.30

cv_54017 Subscribe Send Feedback

The hard processor system (HPS) provides two Ethernet media access controller (EMAC) peripherals. Each EMAC can be used to transmit and receive data at 10/100/1000 Mbps over Ethernet connections in compliance with the IEEE 802.3 specification. The EMACs are instances of the Synopsys DesignWare 3504-0 Universal 10/100/1000 Ethernet MAC (DWC_gmac).

The EMAC has an extensive memory-mapped control and status register (CSR) set, which can be accessed by the ARM Cortex-A9® MPCore.

For an understanding of this chapter, you should be familiar with the basics of IEEE 802.3 media access control (MAC).

[40]

Related Information

http://standards.ieee.ort/findstds/index.html
For complete information about IEEE 802.3 MAC, refer to the *IEEE 802.3 2008 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, available on the IEEE Standards Association website.

Features of the Ethernet MAC

MAC

- IEEE 802.3-2008 compliant
- Data rates of 10/100/1000 Mbps
- IEEE 802.3x flow control in full-duplex
- Full duplex and half duplex modes
- IEEE 802.3x flow control automatic transmission of zero-quanta pause frame on flow control input deassertion

[40] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

- Optional forwarding of received pause control frames to the user
- Packet bursting and frame extension in 1000 Mbps half-duplex
- IEEE 802.3x flow control in full-duplex
- Back-pressure support for half-duplex
- IEEE 1588-2002 and IEEE 1588-2008 precision networked clock synchronization
- IEEE 802.3-az, version D2.0 for Energy Efficient Ethernet (EEE)
- IEEE 802.1Q virtual local area network (VLAN) tag detection for reception frames
- Preamble and start-of-frame data (SFD) insertion in transmit and deletion in receive paths
- Automatic cyclic redundancy check (CRC) and pad generation controllable on a per-frame basis
- Options for automatic pad/CRC stripping on receive frames
- Programmable frame length supporting standard and jumbo Ethernet frames (with size up to 16 KB)
- Programmable inter-frame gap (IFG), from 40- to 96-bit times in steps of eight

PHY Interface

- Reduced Gigabit Media Independent Interface (RGMII) for 10/100/1000
- Gigabit Media Independent Interface (GMII) through the FPGA fabric
- Serial Gigabit Media Independent Interface (SGMII) supported through the GMII to FPGA fabric
- Media Independent Interface (MII) through the FPGA fabric
- Management Data Input/Output (MDIO) (IEEE 802.3) or I^{2}C PHY management interface

DMA Interface

- 32-bit interface
- Programmable burst size for optimal bus utilization
- Single-channel mode transmit and receive engines
- Byte-aligned addressing mode for data buffer support
- Dual-buffer (ring) or linked-list (chained) descriptor chaining
- Descriptors can each transfer up to 8 KB of data

Management Interface

- 32-bit host interface to CSR set
- Comprehensive status reporting for normal operation and transfers with errors
- Configurable interrupt options for different operational conditions
- Per-frame transmit/receive complete interrupt control
- Separate status returned for transmission and reception packets

Acceleration

- Transmit and receive checksum offload for transmission control protocol (TCP), user datagram protocol (UDP), or Internet control message protocol (ICMP) over Internet protocol (IP)

Other Features

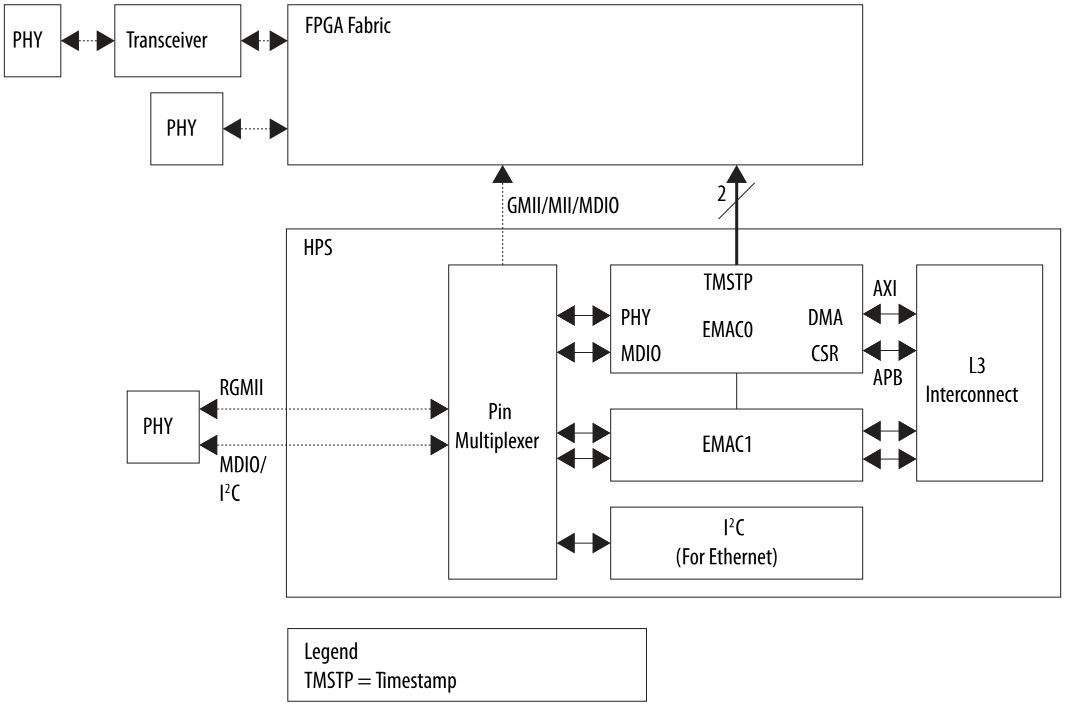
- Supports a variety of flexible address filtering modes
- Up to 31 additional 48-bit perfect destination address (DA) filters with masks for each byte
- Up to 31 48-bit source address (SA) comparison check with masks for each byte
- 256-bit hash filter (optional) for multicast and uni-cast DAs

- Option to pass all multicast addressed frames
- Promiscuous mode support to pass all frames without any filtering for network monitoring
- Passes all incoming packets (as per filter) with a status report

EMAC Block Diagram and System Integration

Figure 17-1: EMAC System Integration

EMAC integration from a high level point of view.



The EMACs are integrated into the HPS portion of the system on a chip (SoC) device. They communicate with the I/O pins.

EMAC to RGMII Interface

Table 17-1: External PHY Datapath In/Out Interface

EMAC Port		In/Out	Width	Description
clk_tx_i	Transmit Clock	In	1	This signal provides the transmit clock for RGMII (125/25/2.5 MHz in 1G/100M/10Mbps). All PHY transmit signals generated by the EMAC are synchronous to this clock.

EMAC Port		In/Out	Width	Description
phy_txd_o	PHY Transmit Data	Out	8	This is a group of eight transmit data signals driven by the MAC. Unused bits in the RGMII interface configuration are tied to low. RGMII: Bits [3:0] provide the RGMII transmit data. The data bus changes with both rising and falling edges of the transmit clock (clk_tx_i). The validity of the data is qualified with phy_txen_o. Synchronous to: clk_tx_i, clk_tx_180_i
phy_txen_o	PHY Transmit Data Enable	Out	1	This signal is driven by the EMAC component. RGMII: This signal is the control signal (rgmii_tctl) for the transmit data, and is driven on both edges of the clock. Synchronous to: clk_tx_i, clk_tx_180_i
rst_clk_tx_n_o	Transmit clock reset output.	Out	1	Transmit clock reset output.
clk_rx_i	Receive Clock	In	1	Clock frequency is 125/25/2.5 MHz in 1 G/100 M/10 Mbps modes. It is provided by the external PHY. All PHY signals received by the EMAC are synchronous to this clock.
phy_rxd_i	PHY Receive Data	In	8	This is a bundle of eight data signals received from the PHY. RGMII: Bits [3:0] provide the RGMII receive data. The data bus is sampled with both rising and falling edges of the receive clock (clk_rx_i). The validity of the data is qualified with phy_rxdv_i. Synchronous to: clk_rx_i, clk_rx_180_i
phy_rxdv_i	PHY Receive Data Valid	In	1	This signal is driven by PHY. RGMII: This is the receive control signal used to qualify the data received on phy_rxd. This signal is sampled on both edges of the clock. Synchronous to: clk_rx_i, clk_rx_180_i
rst_clk_rx_n_o	Receive clock reset output.	Out	1	Receive clock reset output.

EMAC Port		In/Out	Width	Description
phy_intf_sel_i[1:0]	PHY Interface Select	In	2	These pins select one of the PHY interfaces of the EMAC. This signal is sampled only during reset assertion and ignored after that. • 01: RGMII • 00, 10, and 11: Invalid
clk_ref_i	EMAC Reference Clock	In	1	This is the reference clock to the EMAC. The clock is emac0_clk or emac1_clk supplied by the clock manager. The system manager drives the phy_intf_sel signal to control which clock is used. The clock rate is 250 MHz.

EMAC to FPGA PHY Interface

Table 17-2: EMAC to FPGA PHY Interface Signals

Signal Name		In/Out	Width	Description
phy_tx_i	Transmit Clock	In	1	This is the transmit clock (2.5 MHz/25 MHz) provided by the MII PHYs only. This clock comes from the FPGA Interface and is used for TX data capture.
phy_txclk_o	Transmit Clock Output	Out	1	Transmit clock output to the PHY to sample data. For MII this clock is unused. For GMII (1G - 125 MHz) and RGMII (125/25/2.5 MHz), this clock is derived from the clk_ref_i 250 MHz input.

Signal Name		In/Out	Width	Description
phy_txd_o	PHY Transmit Data	Out	8	These are a group of eight transmit data signals driven by the MAC. Unused bits in the RGMII interface configuration are driven low. The following functions depend on which PHY interface is selected: • GMII: All eight bits provide the GMII transmit data byte. For the lower speed MII operation, only the low 4 bits are used. The validity of the data is qualified with phy_txen_o and phy_txer_o. Synchronous to: phy_ clk_tx_o. • RGMII: Bits [3:0] provide the RGMII transmit data. The data bus changes with both rising and falling edges of the transmit clock (clk_tx_o). The validity of the data is qualified with phy_txen_o. Synchronous to: phy_clk_tx_o (both the rising and falling edges).
phy_txen_o	PHY Transmit Data Enable	Out	1	This signal is driven by the EMAC component and has the following function listed below: • GMII: When high, indicates that valid data is being transmitted on the phy_txd bus. Synchronous to: clk_tx_o

Signal Name		In/Out	Width	Description
phy_txer_o	PHY Transmit Error	Out	1	This signal is driven by the MAC and has multiple functions depending on which PHY interface is selected, as explained below: • GMII / MII: When high, indicates a transmit error or carrier extension on the phy_txd bus. Also used for signaling of low power states in Energy Efficient Ethernet operation. Synchronous to: phy_clk_tx_o. • RGMII: Not used. Tied low in some configurations; driven low in others.
rst_clk_tx_n_o	Transmit Clock Reset output	Out	1	Transmit clock reset output to the FPGA fabric: This is the internal synchronized reset to clk_tx_int output from the EMAC. May be used by logic implemented in the FPGA fabric as desired.
phy_clk_rx_i	Receive Clock	In	1	Receive clock from external PHY. For RGMII, the clock frequency is 125/25/2.5 MHz in 1 G, 100 M, and 10 Mbps modes. For GMII, the clock frequency is 125 MHz.

Signal Name		In/Out	Width	Description
phy_rxd_i	PHY Receive Data	In	8	This is a bundle of eight data signals received from the PHY. It has multiple functions depending on which PHY interface is selected, as listed below: • GMII: All 8 bits provide the GMII receive data byte. The validity of the data is qualified with phy_rxdv_i and phy_rxer_i. For lower speed MII operation, only the bottom 4 bits are used. Synchronous to: phy_ clk_rx_i. • RGMII: Bits [3:0] provide the RGMII receive data. The data bus is sampled on both rising and falling edges of the receive clock (phy_clk_rx_i). The validity of the data is qualified with phy_rxdv_i. Synchronous to: phy_ clk_rx_i (both rising and falling edges).
phy_rxdv_i	PHY Receive Data Valid	In	1	This signal is driven by PHY and has multiple functions depending on which PHY interface is selected, as listed below: • GMII: When high, indicates that the data on the phy_rxd bus is valid. It remains asserted continuously from the first recovered byte of the frame through the final recovered byte. Synchronous to: phy_ clk_rx_i • RGMII: This is the receive control signal (RX_CTL) used to qualify the data received on phy_rxd. On the rising edge of the clock, rxdv is sampled. On the falling edge of the clock, rxdv XOR rxerr is sampled. Synchronous to: phy_clk_rx_i, both rising and falling edges.

Signal Name		In/Out	Width	Description
phy_rxer_i	PHY Receive Error	In	1	This signal is driven by the PHY. In GMII / MII mode, it indicates an error or carrier extension (GMII) in the received frame. Synchronous to: phy_clk_rx_i, This signal is not used in RGMII mode.
rst_clk_rx_n_o	Receive clock reset output.	Out	1	Receive clock reset output.
phy_crs_i	PHY Carrier Sense	In	1	This signal, valid only in the GMII / MII mode, is asserted by the PHY when either the transmit or receive medium is not idle. The PHY de-asserts this signal when both transmit and receive medium are idle. This signal is not synchronous to any clock.
phy_col_i	PHY Collision Detect	In	1	PHY Collision Detect. This signal, valid only in GMII / MII mode operating in half duplex, is asserted by the PHY when a collision is detected on the medium. This signal is not synchronous to any clock.
gmii_mdi_i	MDIO signal input	In	1	This signal is driven synchronously with the gmii_mdc_o clock.
gmii_mdo_o	MDIO signal data out	Out	1	MDIO signal data out. This signal is driven synchronously with the gmii_mdc_o clock.
gmii_mdo_o_e	MDIO signal output enable	Out	1	MDIO signal output enable. This signal is driven synchronously with the gmii_mdc_o clock.
gmii_mdc_o	Management Data Clock	Out	1	The maximum frequency of this clock is 2.5 MHz. This clock is generated from the application clock through a clock divider.

PHY Management Interface

The HPS can provide support for either MDIO or I^{2}C PHY management interfaces.

MDIO Interface

The MDIO interface signals are synchronous to l4_mp_clk in all supported modes.

Table 17-3: PHY MDIO Management Interface

Signal	In/Out	Width	Description
gmii_mdi_i	In	1	Management Data In. The PHY generates this signal to transfer register data during a read operation. This signal is driven synchronously with the gmii_mdc_o clock.
gmii_mdo_o	Out	1	Management Data Out. The EMAC uses this signal to transfer control and data information to the PHY.
gmii_mdo_o_e	Out	1	Management Data Output Enable. This enable signal drives the gmii_mdo_o signal from an external three-state I/O buffer. This signal is asserted whenever valid data is driven on the gmii_mdo_o signal. The active state of this signal is high.
gmii_mdc_o	Out	1	Management Data Clock. The EMAC provides timing reference for the gmii_mdi_i and gmii_mdo_o signals on MII through this aperiodic clock. The maximum frequency of this clock is 2.5 MHz. This clock is generated from the application clock through a clock divider.

I^{2}C External PHY Management Interface

Some PHY devices use the I^{2}C instead of MDIO for their control interface. Small form factor pluggable (SFP) optical or pluggable modules are often among those with this interface.

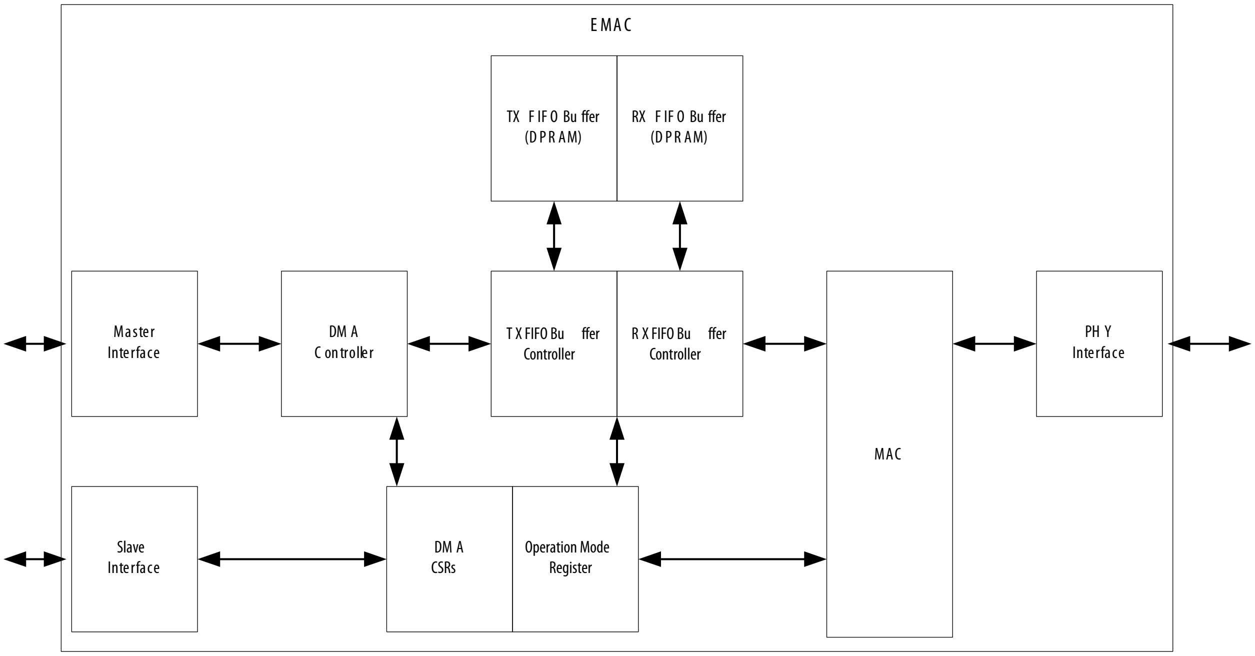
The HPS can use two of the four general purpose I^{2}C peripherals for controlling the PHY devices.

IEEE 1588

The EMAC supports IEEE 1588 operation in all modes with a resolution of one µs. It can be used by the ARM® Cortex™-A9 microprocessor unit (MPU) subsystem to maintain synchronization between the time counters internal to the two MACs.

Functional Description of the EMAC

Figure 17-2: EMAC High-Level Block Diagram with Interfaces



There are two host interfaces to the MAC. The management host interface, a 32-bit slave interface, provides access to the CSR set. The data interface is a 32-bit interface. It controls data transfer between the direct memory access (DMA) controller channels and the rest of the HPS system through the NIC 301 L3 interconnect.

There is a built-in DMA controller which is optimized for data transfer between the MAC controller and system memory. The DMA controller has independent transmit and receive engines, and a CSR set. The transmit engine transfers data from system memory to the device port, while the receive engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal host intervention.

The EMAC also contains FIFO buffer memory to buffer and regulate the Ethernet frames between the application system memory and the EMAC controller. On transmit, the Ethernet frames read into the transmit FIFO buffer (1024 x 42 bits), and eventually trigger the MAC to perform the transfer. Received Ethernet frames are stored in the receive FIFO buffer, also indicating the FIFO buffer bits fill level to the DMA controller. The DMA controller then initiates the configured burst transfers. Both receive and transmit transfer status are taken from the MAC and transferred to the DMA.

Host Interfaces

There are two host interfaces in the EMAC: a slave and a master. The master is connected to the L3 master peripheral switch interface in the L3 interconnect block.

Slave

The EMAC CSR set access is provided by a slave interface. The slave is connected to the level 4 (L4) bus.

DMA Master

The DMA interface is provided by a master interface. Two types of data are transferred on the interface: data descriptors and actual data packets. The interface is very efficient in transferring full duplex Ethernet packet traffic. Read and write data transfers from different DMA channels can be performed simultaneously on this port. The only exceptions to this are transmit descriptor reads and write-backs which cannot happen simultaneously.

DMA transfers are split into a software configurable number of burst transactions on the interface. The AXI_Bus_Mode register in the dmagrp group is used to configure bursting behavior.

The interface assigns a unique ID for each DMA channel and also for each read DMA or write DMA request in a channel. Data transfers with distinct IDs can be reordered and interleaved.

Write data transfers are generally performed as posted writes with OK responses returned as soon as the interconnect has accepted the last beat of a data burst. Descriptors (status or timestamp) however are always transferred as non-posted writes in order to prevent race conditions with the transfer complete interrupt logic.

The slave may issue an error response. When that happens, the EMAC disables the DMA channel which generated the original request and asserts an interrupt signal. The host needs to reset the EMAC with a hard or soft reset to restart the DMA to recover from this condition.

The EMAC supports up to 16 outstanding transactions on the interface. Buffering outstanding transactions smooths out back pressure behavior. This is important when resource contention bottlenecks arise under high system load conditions.

Cache Control Interface

The system manager provides the values for the master cache outputs through this interface. These inputs are used as the outputs to the L3 interconnect extending the capabilities of this block with respect to the cacheable characteristics of master transfers.

To configure EMAC DMA controller to perform cacheable accesses, configure the cache bits in the system manager. Register bits should be accessed only when the master interface is guaranteed to be in an inactive state.

Related Information

System Manager on page 5-1

External PHY

The following PHY interfaces are supported for the HPS:

- RGMII for 10/100/1000

The EMAC also has a control interface used for configuration and status monitoring of the PHY. In this case, the PHY is the slave device. There are two choices of control interface:

- MDIO
- I^{2}C interface

The MDIO interface is built into the EMAC while the I^{2}C interface uses separate I^{2}C peripherals residing on the HPS. The interfaces are multiplexed externally to the EMAC.

Transmit and Receive Data FIFO Buffers

Each EMAC component has associated transmit and receive data FIFO buffer instances. Both FIFO buffer instances are 1024 x 42 bits. The FIFO buffer word consists of:

- Data: 32 bits
- Sideband:
    - Byte enables (BE): 2 bits
    - End of frame (EOF): 1 bit
    - Error correction code (ECC): 7 bits

The data and sideband are protected by the 7-bit single error correct, double error detect (SEC-DED) code word. These FIFO buffer RAMs also contain ECC enable, error injection and status pins. The enable and error injection pins are inputs driven by the system manager and the status pins are outputs driven to the MPU subsystem.

Note: The ECC block provides outputs to notify the system manager when single-bit correctable errors are detected (and corrected) and when double-bit uncorrectable errors are detected. The ECC logic also allows the injection of single-bit and double-bit errors for test purposes.

IEEE 1588-2002 Timestamps

The IEEE 1588-2002 standard defines the Precision Time Protocol (PTP) which enables precise synchronization of clocks in a distributed network of devices. The PTP applies to systems communicating by local area networks supporting multicast messaging. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. It is frequently used in automation systems where a collection of communicating machines such as robots must be synchronized and hence operate over a common time base.[(†)]

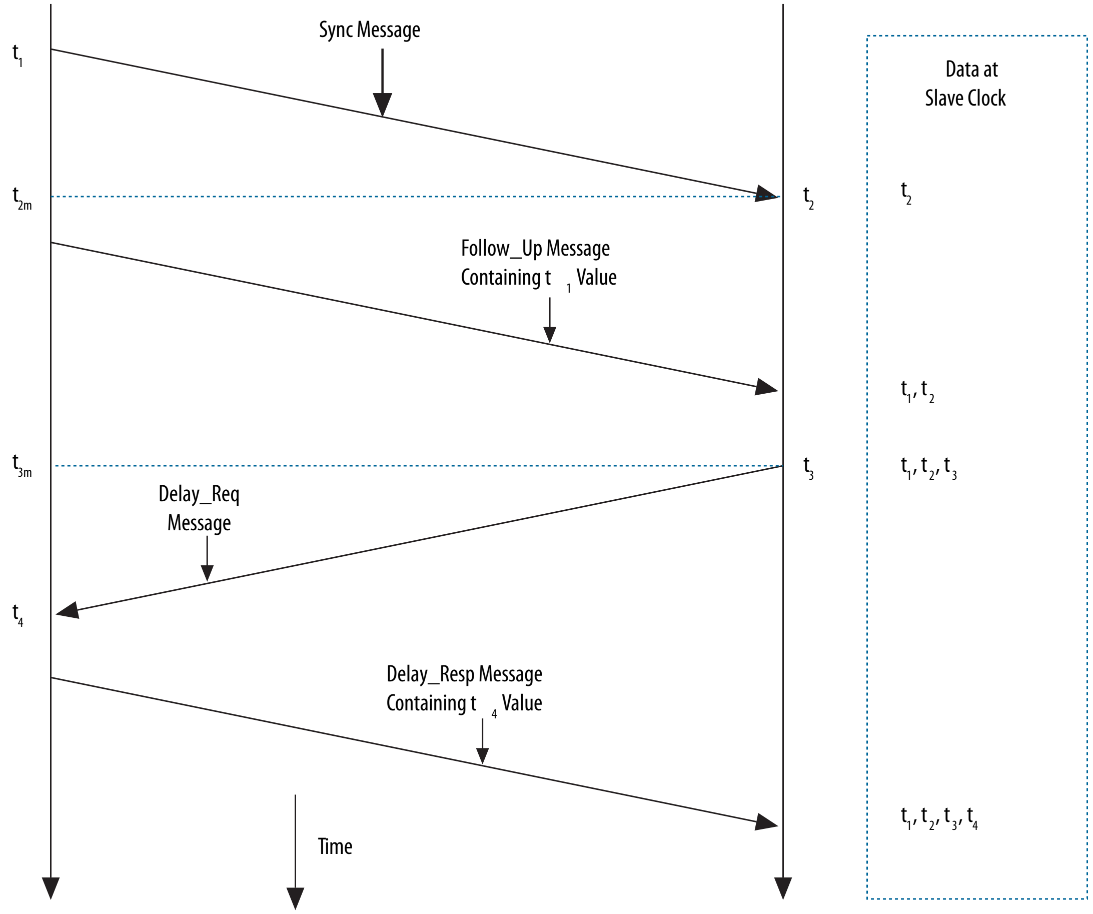
The PTP is transported over UDP/IP. The system or network is classified into Master and Slave nodes for distributing the timing and clock information.†

The following figure shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

[(†)] Portions © 2013 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

Figure 17-3: Networked Time Synchronization



The PTP uses the following process for synchronizing a slave node to a master node by exchanging the PTP messages:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t1. This time must be captured, for Ethernet ports, at the PHY interface.†
2. The slave receives the sync message and also captures the exact time, t2, using its timing reference.†
3. The master sends a follow_up message to the slave, which contains t1 information for later use.†
4. The slave sends a delay_req message to the master, noting the exact time, t3, at which this frame leaves the PHY interface.†
5. The master receives the message, capturing the exact time, t4, at which it enters its system.†
6. The master sends the t4 information to the slave in the delay_resp message.†
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.†

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the PHY interface. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.†

The EMAC is intended to support IEEE 1588 operation in all modes with a resolution of one. When the two EMACs are operating in an IEEE 1588 environment, the MPU subsystem is responsible for maintaining synchronization between the time counters internal to the two MACs.

The IEEE 1588 interface to the FPGA allows the FPGA to provide a source for the emac_ptp_ref_clk input as well to allow it to monitor the pulse per second output from each EMAC controller.

The EMAC component provides a hardware assisted implementation of the IEEE 1588 protocol. Hardware support is for timestamp maintenance. Timestamps are updated when receiving any frame on the PHY interface and the receive descriptor is updated with this value. Timestamps are also updated when the SFD of a frame is transmitted and updates the transmit descriptor accordingly.†

Related Information

http://standards.ieee.ort/findstds/index.html

For details about the IEEE 1588-2002 standard, refer to IEEE Standard 1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, available on the IEEE Standards Association website.†

EMAC to FPGA IEEE1588 Timestamp Interface

Table 17-4: EMAC to FPGA IEEE1588 Timestamp Interface

Signal Name		In/Out	Width	Description
f2h_emac_ptp_ref_clk	Timestamp Precision Time Protocol	In	1	Timestamp PTP Clock reference from the FPGA. Used as PTP clock reference for each EMAC when the FPGA has implemented timestamp capture interface. Common for all three EMACs.
ptp_pps_o	Pulse Per Second	Out	1	This signal is asserted based on the PPS mode selected in the Register 459 (PPS Control Register). Otherwise, this pulse signal is asserted every time the seconds counter is incremented. Synchronous to f2h_emac_f2hptp_ref_clk. Can only be sampled if FPGA clock is used as timestamp reference.
ptp_aux_ts_trig_i	Auxiliary Timestamp Trigger.	In	1	This signal is asserted to take an auxiliary snapshot of the time and store it in the 4-deep auxiliary timestamp FIFO. A rising edge on this port is used to trigger the auxiliary snapshot. The signal is synchronized internally with clk_ptp_ref_i which results in an additional delay of three cycles.

Reference Timing Source

To get a snapshot of the time, the EMAC takes the reference clock input and uses it to generate the reference time (64-bit) internally and capture timestamps.†

System Time Register Module

The 64-bit time is maintained in this module and updated using the input reference clock, osc1_clk. The osc1_clk clock comes from the clock manager and the emac_ptp_ref_clk clock comes from the FPGA fabric. This time is the source for taking snapshots (timestamps) of Ethernet frames being transmitted or received at the PHY interface.

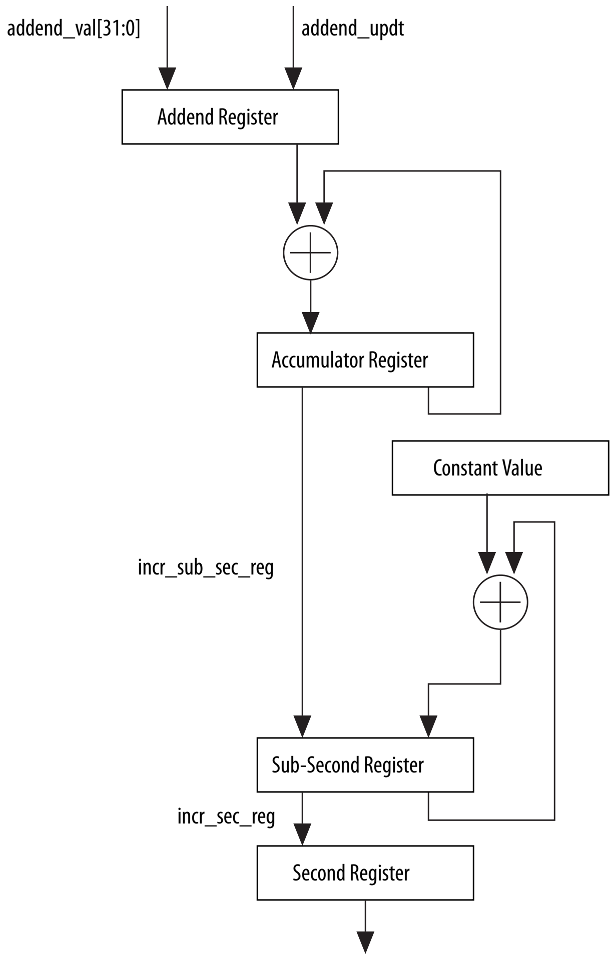
The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Timestamp Update register. For initialization, each EMAC's system time counter is written with the value in the Timestamp Update registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, a slave clock's frequency drift with respect to the master clock is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP sync message intervals.†

In this method, an accumulator sums up the contents of the Timestamp_Addend register, as shown in the figure below. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

Note: You must connect a PTP clock with a frequency higher than the frequency required for the specified accuracy.†

Figure 17-4: Algorithm for System Time Update Using Fine Method



The System Time Update logic requires a 50 MHz clock frequency to achieve 20-ns accuracy. The frequency division ratio (FreqDivisionRatio) is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock (clk_ptp_ref_i) is, for example, 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Hence, the default addend value to be set in the register is 232 / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 232 / 1.30, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (232 / 1.32) must be programmed.†

In the above figure, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20-ns steps).

The software must calculate the drift in frequency based on the Sync messages and update the Addend register accordingly.†

Initially, the slave clock is set with `FreqCompensationValue0` in the Addend register. This value is as follows:†

```
FreqCompensationValue0 = 232 / FreqDivisionRatio†
```

If MasterToSlaveDelay is initially assumed to be the same for consecutive sync messages, the algorithm described below must be applied. After a few sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.†

The algorithm is as follows:†

- At time $\mathtt{MasterSyncTime}_n$ the master sends the slave clock a sync message. The slave receives this message when its local clock is $\mathtt{SlaveClockTime}_n$ and computes $\mathtt{MasterClockTime}_n$ as:†

  $$\mathtt{MasterClockTime}_n = \mathtt{MasterSyncTime}_n + \mathtt{MasterToSlaveDelay}_n$$ †

- The master clock count for current sync cycle, $\mathtt{MasterClockCount}_n$ is given by:†

  $$\mathtt{MasterClockCount}_n = \mathtt{MasterClockTime}_n - \mathtt{MasterClockTime}_{n-1} \texttt{ (assuming that MasterToSlaveDelay is the same for sync cycles n and n - 1)}$$ †

- The slave clock count for current sync cycle, $\mathtt{SlaveClockCount}_n$ is given by:†

  $$\mathtt{SlaveClockCount}_n = \mathtt{SlaveClockTime}_n - \mathtt{SlaveClockTime}_{n-1}$$ †

- The difference between master and slave clock counts for current sync cycle, $\mathtt{ClockDiffCount}_n$ is given by:†

  $$\mathtt{ClockDiffCount}_n = \mathtt{MasterClockCount}_n - \mathtt{SlaveClockCount}_n$$ †

- The frequency-scaling factor for slave clock, $\mathtt{FreqScaleFactor}_n$ is given by:†

  $$\mathtt{FreqScaleFactor}_n = (\mathtt{MasterClockCount}_n + \mathtt{ClockDiffCount}_n) / \mathtt{SlaveClockCount}_n$$ †

- The frequency compensation value for Addend register, $\mathtt{FreqCompensationValue}_n$ is given by: †

  $$\mathtt{FreqCompensationValue}_n = \mathtt{FreqScaleFactor}_n \times \mathtt{FreqCompensationValue}_{n-1} - 1$$ †

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions.†

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more Sync cycles.†

Transmit Path Functions

The MAC captures a timestamp when the SFD of a frame is sent on the PHY interface. The frames for which you want to capture timestamps are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame. The MAC does not process the transmitted frames to identify the PTP frames. You need to specify the frames for which you want to capture timestamps. The MAC returns the timestamp, along with the Transmit status of the frame, to hardware implemented in the FPGA. You can use the control bits in the transmit descriptor. The MAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific PTP frame.†

Receive Path Functions

The MAC captures the timestamp of all frames received on the PHY interface. The DMA returns the timestamp to the software in the corresponding receive descriptor. The timestamp is written only to the last receive descriptor.†

Timestamp Error Margin

According to the IEEE 1588 specifications, a timestamp must be captured at the SFD of the transmitted and received frames at the PHY interface. Because the PHY interface receive and transmit clocks are not synchronous to the reference timestamp clock (osc1_clk) a small amount of drift is introduced when a timestamp is moved between asynchronous clock domains. In the transmit path, the captured and reported timestamp has a maximum error margin of two PTP clocks. It means that the captured timestamp has the reference timing source value that is given within two clocks after the SFD is transmitted on the PHY interface.

Similarly, in the receive path, the error margin is three PHY interface clocks, plus up to two PTP clocks. You can ignore the error margin because of the PHY interface clocks by assuming that this constant delay is present in the system (or link) before the SFD data reaches the PHY interface of the MAC.†

Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, that is, from MAC clock domain to the FPGA clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of the PHY interface and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (20 ns resulting in 50 MHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.†

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the PHY interface clock frequency is fixed by the IEEE 1588 specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC.†

Table 17-5: Minimum PTP Clock Frequency Example

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency
100-Mbps full-duplex operation	168 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 16 clocks of preamble)	(3 * PTP) + (4 * MII) <= 168 * MII, that is, ~0.5 MHz (168 – 4) * 40 ns ÷ 3 = 2180 ns period)
1000-Mbps half duplex operation	24 GMII clocks (4 for a jam pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	(3 * PTP) + 4 * GMII <= 24 * GMII, that is, 18.75 MHz

Related Information

http://standards.ieee.ort/findstds/index.html
For details about jam patterns, refer to the *IEEE Std 802.3 2008 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, available on the IEEE Standards Association website.

IEEE 1588-2008 Advanced Timestamps

In addition to the basic timestamp features mentioned in IEEE 1588-2002 Timestamps, the EMAC supports the following advanced timestamp features defined in the IEEE 1588-2008 standard.†

- Supports the IEEE 1588-2008 (version 2) timestamp format. †
- Provides an option to take a timestamp of all frames or only PTP type frames. †
- Provides an option to take a timestamp of only event messages. †
- Provides an option to take the timestamp based on the clock type: ordinary, boundary, end-to-end, or peer-to-peer. †
- Provides an option to configure the EMAC to be a master or slave for ordinary and boundary clock. †
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status. †
- Provides an option to measure sub-second time in digital or binary format. †

Related Information

http://standards.ieee.ort/findstds/index.html
For more information about advanced timestamp features, refer to the *IEEE Standard 1588 - 2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control System, available on the IEEE Standards Association website.*

Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) messages in addition to SYNC, Delay Request, Follow-up, and Delay Response messages.†

Clock Types

The EMAC supports the following clock types defined in the IEEE 1588-2008 standard:

- Ordinary clock†
- Boundary clock†
- End-to-End transparent clock†
- Peer-to-Peer transparent clock†

Reference Timing Source

The EMAC supports the following reference timing source features defined in the IEEE 1588-2008 standard:

- 48-bit seconds Field†
- Fixed pulse-per-second output†
- Flexible pulse-per-second output†
- Auxiliary snapshots (timestamps) with external events

Transmit Path Functions

The advanced timestamp feature is supported through the descriptors format.

Receive Path Functions

The MAC processes the received frames to identify valid PTP frames. You can control the snapshot of the time, to be sent to the application, by using the following options:†

- Enable timestamp for all frames.†
- Enable timestamp for IEEE 1588 version 2 or version 1 timestamp.†
- Enable timestamp for PTP frames transmitted directly over Ethernet or UDP/IP Ethernet.†
- Enable timestamp snapshot for the received frame for IPv4 or IPv6.†
- Enable timestamp snapshot for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP) only.†
- Enable the node to be a master or slave and select the timestamp type. This controls the type of messages for which timestamps are taken.†

The DMA returns the timestamp to the software inside the corresponding transmit or receive descriptor.

Auxiliary Snapshot

The auxiliary snapshot feature allows you to store a snapshot (timestamp) of the system time based on an external event. The event is considered to be the rising edge of the sideband signal ptp_aux_ts_trig_i. One auxiliary snapshot input is available. The depth of the auxiliary snapshot FIFO buffer is 16.

The timestamps taken for any input are stored in a common FIFO buffer. The host can read a register to know which input's timestamp is available for reading at the top of this FIFO buffer. The MAC stores these timestamps in a FIFO buffer. Only 64 bits of the timestamp are stored in the FIFO buffer. When a timestamp is stored, the MAC indicates this to the host with an interrupt. The value of the timestamp is read through a FIFO buffer register access. †

IEEE 802.3az Energy Efficient Ethernet

Energy Efficient Ethernet (EEE) standardized by IEEE 802.3-az, version D2.0 is supported by the EMAC. It is supported by the MAC operating in 10/100/1000 Mbps rates. EEE is only supported when the EMAC is configured to operate with the RGMII PHY interface operating in full-duplex mode. It does not support half-duplex mode.

EEE enables the MAC to operate in Low-Power Idle (LPI) mode. Either end point of an Ethernet link can disable functionality to save power during periods of low link utilization. The MAC controls whether the system should enter or exit LPI mode and communicates this to the PHY.†

Related Information

http://www.ieee802.org/3/
For details about the *IEEE 802.3az Energy Efficient Ethernet standard*, refer to the IEEE 802.3 Ethernet Working Group website.†

LPI Timers

Two timers internal to the EMAC are associated with LPI mode:

- LPI Link Status (LS) Timer†
- LPI TW Timer†

The LPI LS timer counts, in ms, the time expired since the link status is up. This timer is cleared every time the link goes down and is incremented when the link is up again and the terminal count as programmed by the software is reached. The PHY interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as one second in the IEEE standard 802.3-az, version D2.0. The LPI LS timer is 10 bits wide. Therefore, the software can program up to 1023 ms.†

The LPI TW timer counts, in µs, the time expired since the deassertion of LPI. The terminal count of the timer is the value of resolved transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. The MAC supports the LPI TW timer in units of µs. The LPI TW timer is 16 bits wide. Therefore, the software can program up to 65535 µs.†

The EMAC generates the LPI interrupt when the transmit or receive channel enters or exits the LPI state.†

Checksum Offload

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the EMAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. Supported offloading types:

- Transmit IP header checksum†
- Transmit TCP/UDP/ICMP checksum†
- Receive IP header checksum†
- Receive full checksum†

Frame Filtering

The EMAC implements the following types of filtering for receive frames.

Source Address or Destination Address Filtering

The Address Filtering Module checks the destination and source address field of each incoming packet.†

Unicast Destination Address Filter

Up to 128 MAC addresses for unicast perfect filtering are supported. The filter compares all 48 bits of the received unicast address with the programmed MAC address for any match. Default MacAddr0 is always enabled, other addresses MacAddr1–MacAddr127 are selected with an individual enable bit. For MacAddr1–MacAddr31 addresses, you can mask each byte during comparison with the corresponding received DA byte. This enables group address filtering for the DA. The MacAddr32-MacAddr127 addresses do not have mask control and all six bytes of the MAC address are compared with the received six bytes of DA.†

In hash filtering mode, the filter performs imperfect filtering for unicast addresses using a 256-bit hash table. It uses the upper ten bits of the CRC of the received destination address to index the content of the hash table. A value of 0 selects Bit 0 of the selected register, and a value of 111111 binary selects Bit 63 of the Hash Table register. If the corresponding bit is set to one, the unicast frame is said to have passed the hash filter; otherwise, the frame has failed the hash filter.†

Multicast Destination Address Filter

The MAC can be programmed to pass all multicast frames. In Perfect Filtering mode, the multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported. In hash filtering mode, the filter performs imperfect filtering using a 256-bit hash table. For hash filtering, it uses the upper ten bits of the CRC of the received multicast address to index the contents of the hash table. A value of 0 selects Bit 0 of the selected register and a value of 111111 binary selects Bit 63 of the Hash Table register. If the corresponding bit is set to one, then the multicast frame is said to have passed the hash filter; otherwise, the frame has failed the hash filter.†

Hash or Perfect Address Filter

The filter can be configured to pass a frame when its DA matches either the hash filter or the Perfect filter. This configuration applies to both unicast and multicast frames.†

Broadcast Address Filter

The filter does not filter any broadcast frames in the default mode. However, if the MAC is programmed to reject all broadcast frames, the filter drops any broadcast frame.†

Unicast Source Address Filter

The MAC can also perform a perfect filtering based on the source address field of the received frames. Group filtering with SA is also supported. You can filter a group of addresses by masking one or more bytes of the address.†

Inverse Filtering Operation (Invert the Filter Match Result at Final Output)

For both Destination and Source address filtering, there is an option to invert the filter-match result at the final output. The result of the unicast or multicast destination address filter is inverted in this mode.†

VLAN Filtering

The EMAC supports the two kinds of VLAN filtering:

- VLAN tag-based filtering†
- VLAN hash filtering†

VLAN Tag-Based Filtering

In the VLAN tag-based frame filtering, the MAC compares the VLAN tag of the received frame and provides the VLAN frame status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of the received VLAN tag to determine the perfect match. If VLAN tag filtering is enabled, the MAC forwards the VLAN-tagged frames along with VLAN tag match status and drops the VLAN frames that do not match. You can also enable the inverse matching for VLAN frames. In addition, you can enable matching of SVLAN tagged frames along with the default Customer Virtual Local Area Network (C-VLAN) tagged frames.†

VLAN Hash Filtering with a 16-Bit Hash Table

The MAC provides VLAN hash filtering with a 16-bit hash table. The MAC also supports the inverse matching of the VLAN frames. In inverse matching mode, when the VLAN tag of a frame matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a frame is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.†

Layer 3 and Layer 4 Filters

Layer 3 filtering refers to source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering. The frames are filtered in the following ways:†

- Matched frames†
- Unmatched frames†
- Non-TCP or UDP IP frames†

Matched Frames

The MAC forwards the frames, which match all enabled fields, to the application along with the status. The MAC gives the matched field status only if one of the following conditions is true:†

- All enabled Layer 3 and Layer 4 fields match.†
- At least one of the enabled field matches and other fields are bypassed or disabled.†

Using the CSR set, you can define up to four filters, identified as filter 0 through filter 3. When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides status of the lowest filter with filter 0 being the lowest and filter 3 being the highest. For example, if filter 0 and filter 1 match, the MAC gives the status corresponding to filter 0.†

Unmatched Frames

The MAC drops the frames that do not match any of the enabled fields. You can use the inverse match feature to block or drop a frame with specific TCP or UDP over IP fields and forward all other frames. You can configure the EMAC so that when a frame is dropped, it receives a partial frame with appropriate abort status or drops it completely.†

NonTCP or UDP IP Frames

By default, all non-TCP or UDP IP frames are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP frames.†

Clocks and Resets

Table 17-6: Ethernet MAC Controller Clocks

Name	Nominal Frequency	Functional Usage	Notes
clk_ref_i	250 Mhz	Reference Clock to the EMAC	If supplied from clock interface, clock is emac0_clk or emac1_clk
clk_tx_i	125/25/2.5 MHz	Auto negotiates speed down to 10/100 Mbps	
clk_rx_i		PHY provides this reference to MAC	All PHY signals received by the MAC are synchronous to this clock

Clock Gating for EEE

For the RGMII PHY interface, you can gate the transmit clock for Energy Efficient Ethernet (EEE) applications.

Related Information

Programming Guidelines for Energy Efficient Ethernet on page 17-57

Resets

Table 17-7: Ethernet MAC Controller Reset Signals

Name	Functional Usage	Notes
rst_clk_tx_n_o	Transmit clock reset output	Used to reset external PHY transmit clock domain logic
rst_clk_rx_n_o	Receive clock reset output	Used to reset external PHY receive clock domain logic

Interrupts

Interrupts are generated as a result of specific events in the EMAC and external PHY device. The interrupt status register indicates all conditions which may trigger an interrupt and the interrupt enable register determines which interrupts can propagate.

Ethernet MAC Programming Model

DMA Controller

The DMA has independent transmit and receive engines, and a CSR space. The transmit engine transfers data from system memory to the device port or MAC transaction layer (MTL), while the receive engine transfers data from the device port to the system memory. The controller use descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU for situations such as frame transmit and receive transfer completion, and other normal/ error conditions.

The DMA and the Host driver communicate through two data structures:†

- Control and Status registers (CSR)†
- Descriptor lists and data buffers †

Descriptor Lists and Data Buffers†

The DMA transfers data frames received by the MAC to the receive Buffer in the Host memory, and transmit data frames from the transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.†

There are two descriptor lists: one for reception and one for transmission. The base address of each list is written into Register 3 (Receive Descriptor List Address Register) and Register 4 (Transmit Descriptor List Address Register), respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both receive and transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.†

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled.†

Figure 17-5: Descriptor Ring Structure

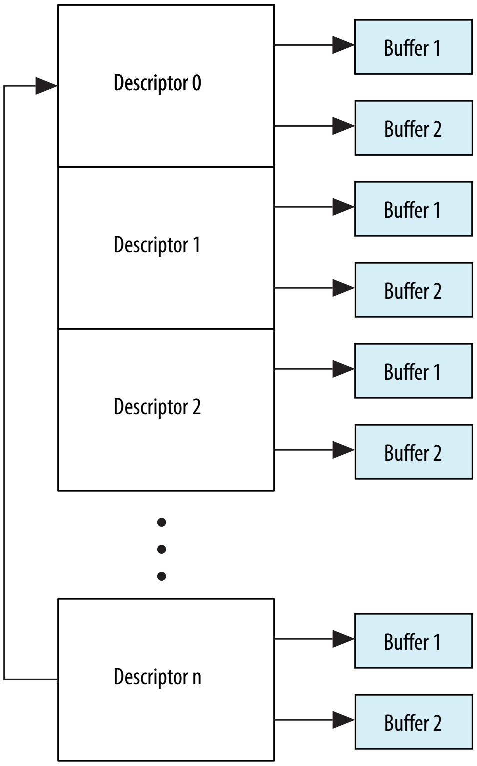
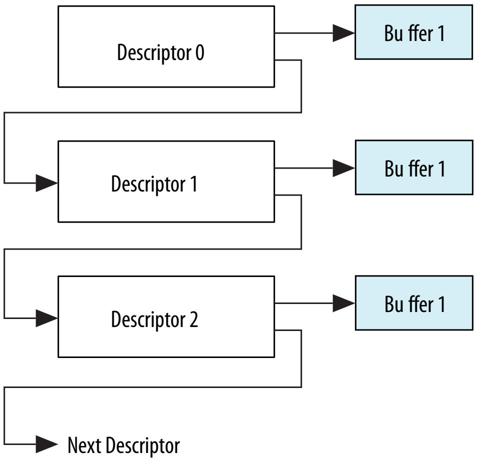


Figure 17-6: Descriptor Chain Structure



Note: You can select a descriptor structure during RTL configuration. The control bits in the descriptor structure are assigned so that the application can use an 8 KB buffer. All descriptions refer to the default descriptor structure.

Related Information

- **Descriptors** on page 17-39
  Detailed bit map of the descriptor structure
- **Ethernet MAC Address Map and Register Definitions** on page 17-60
  Information about Control and Status registers

Initialization for the EMAC

1. Write to Register 0 (Bus Mode Register) to set Host bus access parameters. †
2. Write to Register 7 (Interrupt Enable Register) to mask unnecessary interrupt causes. †
3. Create the transmit and receive descriptor lists, and then write to DMA Register 3 (Receive Descriptor List Address Register) and Register 4 (Transmit Descriptor List Address Register), providing the DMA with the starting address of each list. †
4. Write to Register 1 (MAC Frame Filter), Register 2 (Hash Table High Register), and Register 3 (Hash Table Low Register) for desired filtering options. †
5. Write to Register 1 (MAC Frame Filter) to configure the operating mode and enable the transmit operation (Bit 3: Transmitter Enable). The PS and DM bits are set based on the auto-negotiation result (read from the PHY). †
6. Write to Register 6 (Operation Mode Register) to set Bits 13 and 1 to start transmission and reception. †
7. Write to Register 0 (MAC Configuration Register) to enable the receive operation (Bit 2: Receiver Enable). †

The transmit and receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The receive and transmit engines then begin processing receive and transmit operations. The transmit and receive processes are independent of each other and can be started or stopped separately. †

Host Bus Burst Access

The DMA attempts to execute fixed-length Burst transfers on the master interface if configured to do so through FB bit of Register 0 (Bus Mode Register). The maximum Burst length is indicated and limited by the PBL field (Bits [13:8]) Register 0 (Bus Mode Register). The receive and transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 * 8/bus width) burst-size for the 16- bytes to be read.

The transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL transmit FIFO buffer or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the master interface. When the interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4, 8, or 16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL receive FIFO buffer or when the end of frame (when it is less than the configured burst-length) is detected in the receive FIFO buffer. The DMA indicates the start address and the number of transfers required to the master interface. When the interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4, 8, or 16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the interface, then dummy transfers are performed in order to complete the fixed-burst. Otherwise (FB bit of Register 0 (Bus Mode Register) is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer initiated is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16), because the interface does not support more than INCR16.

Host Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame. The software driver should discard the dummy bytes based on the start address of the buffer and size of the frame.†

Example: Buffer Read

If the transmit buffer address is 0x00000FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA reads five full words from address 0x00000FF0, but when transferring data to the MTL transmit FIFO buffer, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the MTL transmit FIFO buffer, unless it is the end-of-frame.

Example: Buffer Write

If the receive buffer address is 0x0000FF2 (for 64-bit data bus) and 16 bytes of a received frame need to be transferred, then the DMA writes 3 full words from address 0x00000FF0. But the first 2 bytes of first transfer and the last 6 bytes of the third transfer have dummy data.

Buffer Size Calculations

The DMA does not update the size fields in the transmit and receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations. †

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the MAC. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL. †

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The receive DMA always transfers the start of next frame with a new descriptor. †

Note: Even when the start address of a receive buffer is not aligned to the data width of system bus, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the receive descriptor to have a 0x1002 offset. The receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset. †

Transmission

Transmission functions use transmit descriptors.

Related Information

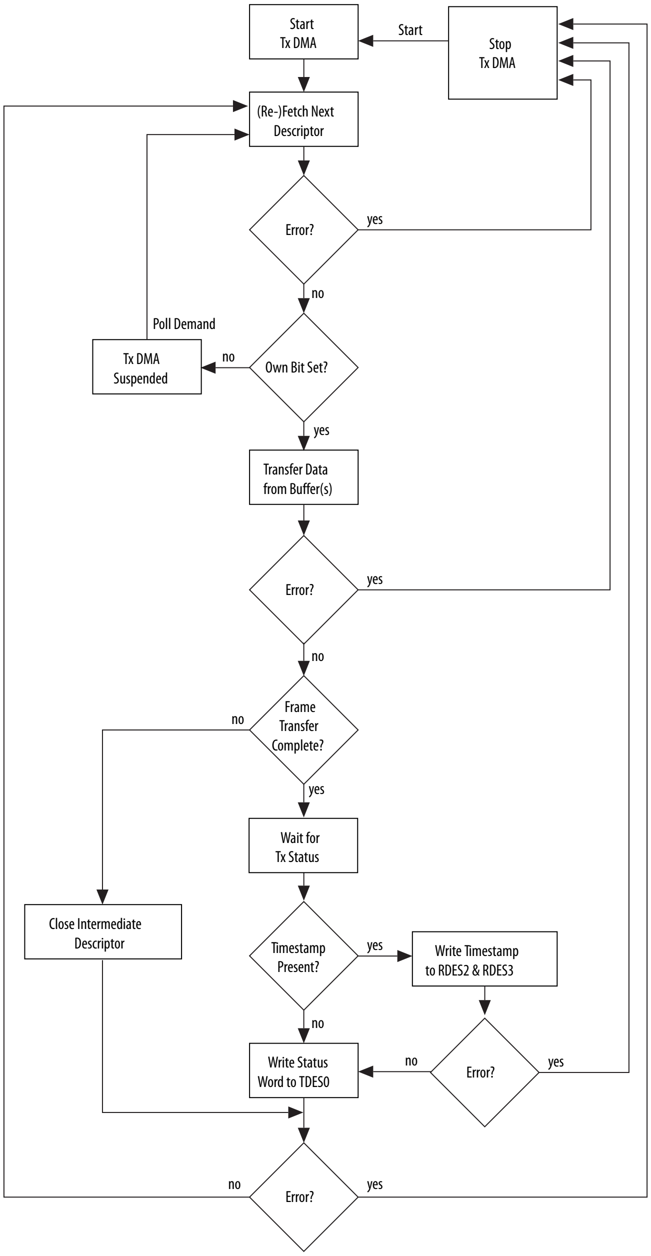
Transmit Descriptor on page 17-39

TX DMA Operation: Default (Non-OSF) Mode

The transmit DMA engine in default mode proceeds as follows: †

1. The Host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet frame data. †
2. When Bit 13 (ST) of Register 6 (Operation Mode Register) is set, the DMA enters the Run state. †
3. While in the Run state, the DMA polls the transmit descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host (TDES0[31] = 0), or if an error condition occurs, transmission is suspended and both the Bit 2 (Transmit Buffer Unavailable) and Bit 16 (Normal Interrupt Summary) of the Register 5 (Status Register) are set. The transmit Engine proceeds to **step 9**.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1), the DMA decodes the transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the transmit data from the Host memory and transfers the data to the MTL for transmission. †
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Repeat **step 3**, **step 4**, and **step 5** until the end-of-Ethernet-frame data is transferred to the MTL. †
7. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the Host now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3. †
8. Bit 0 (Transmit Interrupt) of Register 5 (Status Register) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last descriptor. The DMA engine then returns to **step 3**. †
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to **step 3**) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. †

Figure 17-7: TX DMA Operation in Default Mode



Related Information

TX DMA Operation: Default (Non-OSF) Mode on page 17-29

TX DMA Operation: OSF Mode

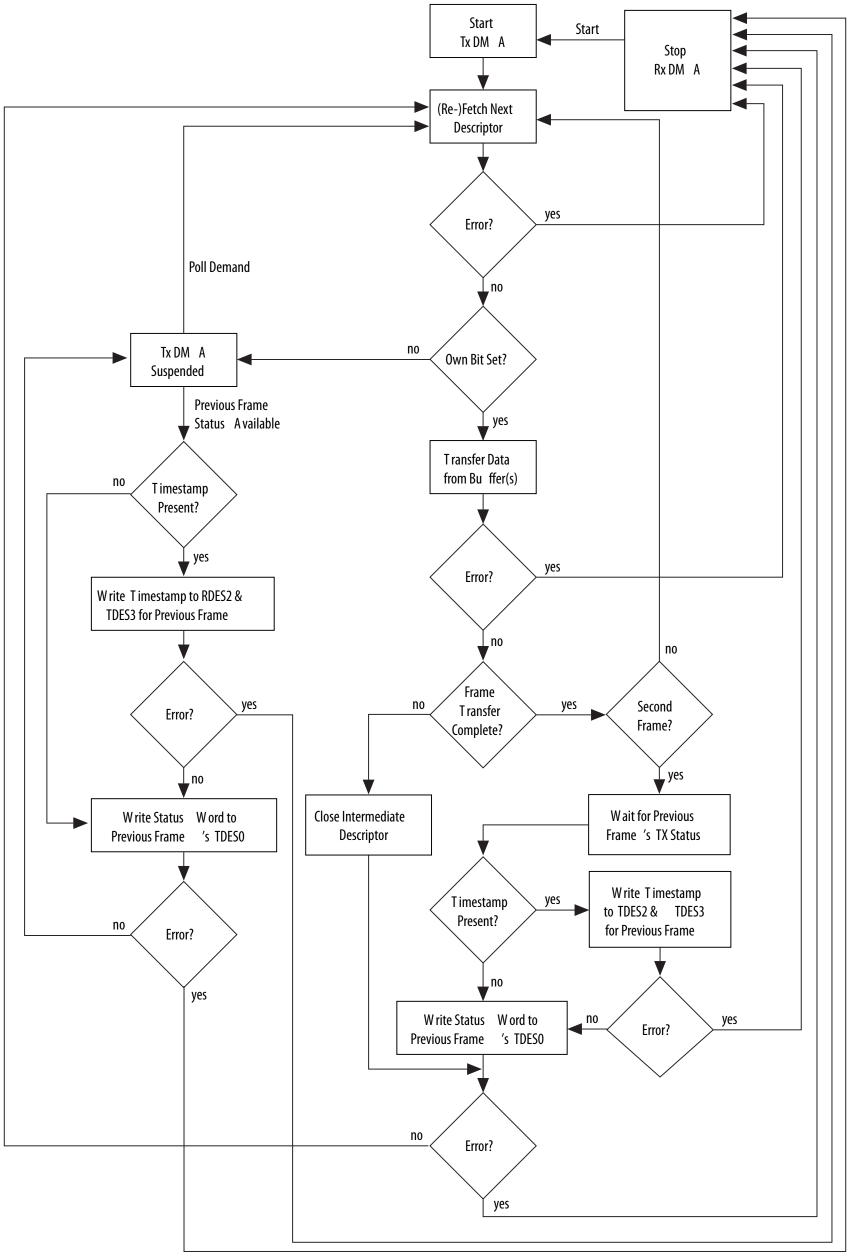
While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first [if Bit 2 (OSF) in Register 6 (Operation Mode Register) is set]. As the transmit process finishes transferring the first frame, it immediately polls the transmit descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information. †

In OSF mode, the Run state transmit DMA operates in the following sequence: †

1. The DMA operates as described in steps 1 - 6 of the **TX DMA Operation: Default (Non-OSF) Mode** on page 17-29 section.
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor. †
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to **step 7**.†
4. The DMA fetches the transmit frame from the Host memory and transfers the frame to the MTL until the End-of-frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors. †
5. The DMA waits for the previous frame's frame transmission status and timestamp. Once the status is available, the DMA writes the timestamp to TDES2 and TDES3, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared Own bit, to the corresponding TDES0, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3. †
6. If enabled, the transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to **step 3** (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (**step 7**). †
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA writes the timestamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to Suspend mode. †
8. The DMA can exit Suspend mode and enter the Run state (go to **step 1** or **step 2** depending on pending status) only after receiving a Transmit Poll demand (Register 1 (Transmit Poll Demand Register). †

Note: As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than two different descriptors for correct and proper operation. †

Figure 17-8: TX DMA Operation in OSF Mode


Start Tx DMA
Start
Stop Rx DMA
(Re-)Fetch Next Descriptor
Error?
yes
no
Poll Demand
Tx DMA Suspended
no
Own Bit Set?
yes
Previous Frame Status Available
Transfer Data from Buffer(s)
no
Timestamp Present?
yes
Write Timestamp to RDES2 & TDES3 for Previous Frame
Error?
yes
no
Error?
yes
no
Frame Transfer Complete?
no
yes
Second Frame?
no
yes
Write Status Word to Previous Frame's TDES0
Close Intermediate Descriptor
Wait for Previous Frame's TX Status
Timestamp Present?
yes
Write Timestamp to TDES2 & TDES3 for Previous Frame
no
Error?
yes
no
Error?
no
Write Status Word to Previous Frame's TDES0
no
Error?
yes
yes
Error?
no
yes

Transmit Frame Processing

The transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Len fields contain valid data. If the transmit descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes. †

Frames can be data-chained and can span several buffers. Frames must be delimited by the First Descriptor (TDES1[29]) and the Last Descriptor (TDES1[30]), respectively. †

As transmission starts, the First Descriptor must have (TDES1[29]) set. When this occurs, frame data transfers from the Host buffer to the MTL transmit FIFO buffer. Concurrently, if the current frame has the Last Descriptor (TDES1[30]) clear, the transmit Process attempts to acquire the Next descriptor. The transmit Process expects this descriptor to have TDES1[29] clear. If TDES1[30] is clear, it indicates an intermediary buffer. If TDES1[30] is set, it indicates the last buffer of the frame. †

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 0 (TDES0) word of the descriptor that has the last segment set in Transmit Descriptor 1 (TDES1[30]). At this time, if Interrupt on Completion (TDES1[31]) is set, the Bit 0 (Transmit Interrupt) of Register 5 (Status Register) is set, the Next descriptor is fetched, and the process repeats. †

The actual frame transmission begins after the MTL transmit FIFO buffer has reached either a programmable transmit threshold (Bits [16:14] of Register 6 (Operation Mode Register)), or a full frame is contained in the FIFO buffer. There is also an option for Store and Forward Mode (Bit 21 of Register 6 (Operation Mode Register)). Descriptors are released (Own bit TDES0[31] clears) when the DMA finishes transferring the frame. †

Note: To ensure proper transmission of a frame and the next frame, you must specify a non-zero buffer size for the transmit descriptor that has the Last Descriptor (TDES1[30]) set. †

Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions: †

- The DMA detects a descriptor owned by the Host (TDES0[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command. †
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set. †

If the DMA goes into SUSPEND state because of the first condition, then both Bit 16 (Normal Interrupt Summary) and Bit 2 (Transmit Buffer Unavailable) of Register 5 (Status Register) are set. If the second condition occur, both Bit 15 (Abnormal Interrupt Summary) and Bit 5 (Transmit Underflow) of Register 5 (Status Register) are set, and the information is written to Transmit Descriptor 0, causing the suspension. †

In both cases, the position in the transmit List is retained. The retained position is that of the descriptor following the Last descriptor closed by the DMA. †

The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause. †

Reception

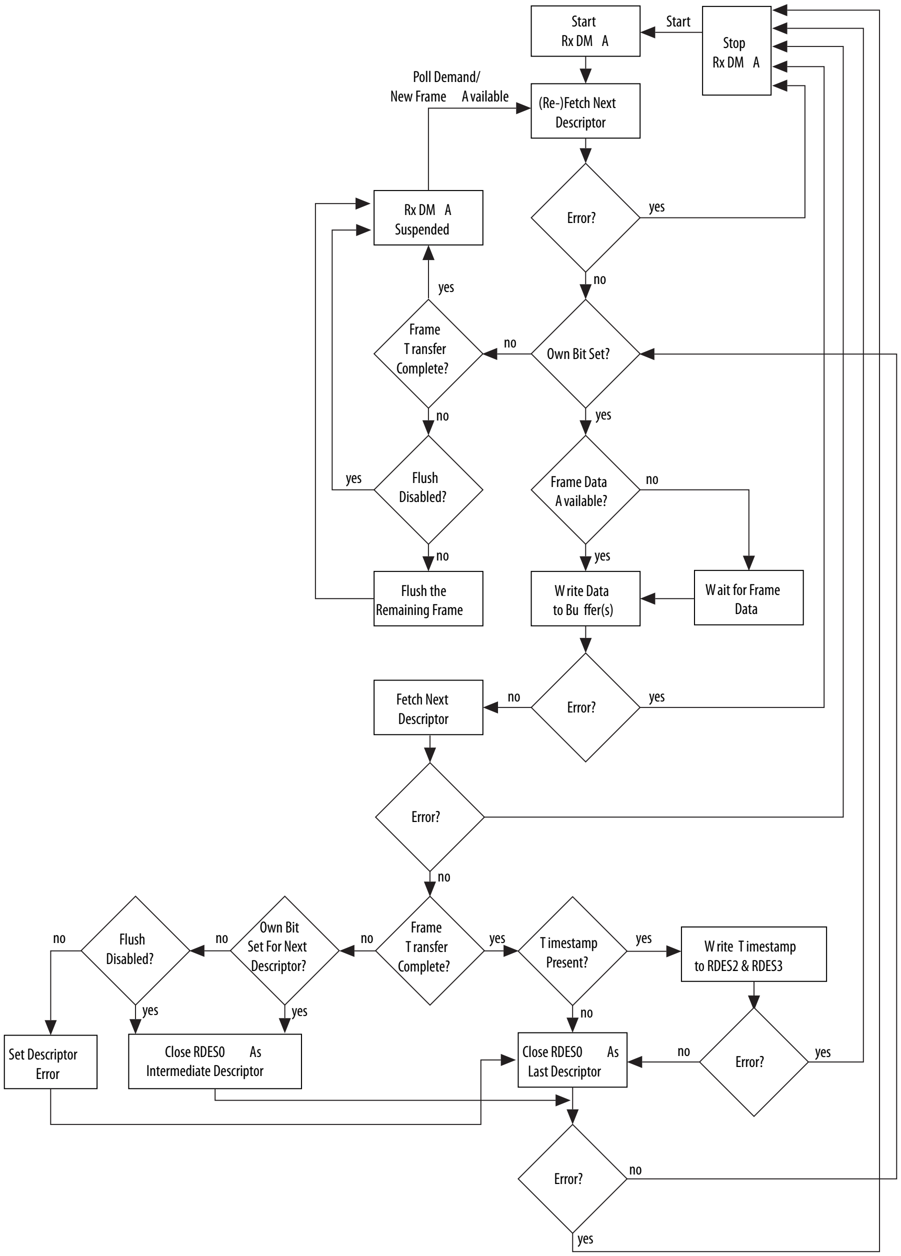
Receive functions use receive descriptors.

The receive DMA engine's reception sequence is depicted proceeds as follows:

1. The host sets up receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).†
2. When Bit 1 (SR) of Register 6 (Operation Mode Register) is set, the DMA enters the Run state. While in the Run state, the DMA polls the receive descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the Suspend state and jumps to step 9.†
3. The DMA decodes the receive data buffer address from the acquired descriptors.†
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.†
5. When the buffer is full or the frame transfer is complete, the receive engine fetches the next descriptor.†

6. If the current frame transfer is complete, the DMA proceeds to **step 7**. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled in Bit 24 of Register 6 (Operation Mode Register)). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to **step 8**. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to **step 4**.†
7. If IEEE 1588 timestamping is enabled, the DMA writes the timestamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MTL and writes the status word to the current descriptor's RDES0, with the Own bit cleared and the Last Segment bit set.†
8. The receive engine checks the latest descriptor's Own bit. If the host owns the descriptor (Own bit is 0), the Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) is set and the DMA receive engine enters the Suspended state (Step 9). If the DMA owns the descriptor, the engine returns to **step 4** and awaits the next frame.
9. Before the receive engine enters the Suspend state, partial frames are flushed from the receive FIFO buffer. You can control flushing using Bit 24 of Register 6 (Operation Mode Register). †
10. The receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's receive FIFO buffer. The engine proceeds to **step 2** and refetches the next descriptor. †

Figure 17-9: Receive DMA Operation



If software has enabled timestamping through CSR, when a valid timestamp value is not available for the frame (for example, because the receive FIFO buffer was full before the timestamp could be written to it), the DMA writes all-ones to RDES2 and RDES3. Otherwise (that is, if timestamping is not enabled), the RDES2 and RDES3 remain unchanged. †

Related Information

Receive Descriptor

Receive Descriptor Acquisition

The receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied: †

- Bit 1 (Start or Stop Receive) of Register 6 (Operation Mode Register) has been set immediately after being placed in the Run state. †
- The data buffer of current descriptor is full before the frame ends for the current transfer. †
- The controller has completed frame reception, but the current receive descriptor is not yet closed. †
- The receive process has been suspended because of a host-owned buffer (RDES0[31] = 0) and a new frame is received. †
- A Receive poll demand has been issued. †

Receive Frame Processing

The MAC transfers the received frames to the Host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the receive FIFO buffer of MTL, or when the complete frame is written to the FIFO buffer in Store-and-Forward mode. †

If the frame fails the address filtering, it is dropped in the MAC block itself (unless Bit 31 (Receive All) of Register 1 (MAC Frame Filter) is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be removed from the MTL receive FIFO buffer. †

After 64 (configurable threshold) bytes have been received, the MTL block requests the DMA block to begin transferring the frame data to the receive Buffer pointed by the current descriptor. The DMA sets First Descriptor (RDES0[9]) after the DMA Host interface becomes ready to receive a data transfer (if DMA is not fetching transmit data from the host), to delimit the frame. The descriptors are released when the Own (RDES[31]) bit is reset to 0, either as the Data buffer fills up or as the last segment of the frame is transferred to the receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES[8]) and First Descriptor (RDES[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets the Bit 6 (Receive Interrupt) of Register 5 (Status Register). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the receive Process sets the Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) and then enters the Suspend state. The position in the receive list is retained. †

Receive Process Suspended

If a new receive frame arrives while the receive Process is in Suspend state, the DMA refetches the current descriptor in the Host memory. If the descriptor is now owned by the DMA, the receive Process re-enters the Run state and starts frame reception. If the descriptor is still owned by the host, by default, the DMA discards the current frame at the top of the MTL RX FIFO buffer and increments the missed frame counter. If more than one frame is stored in the MTL EX FIFO buffer, the process repeats. †

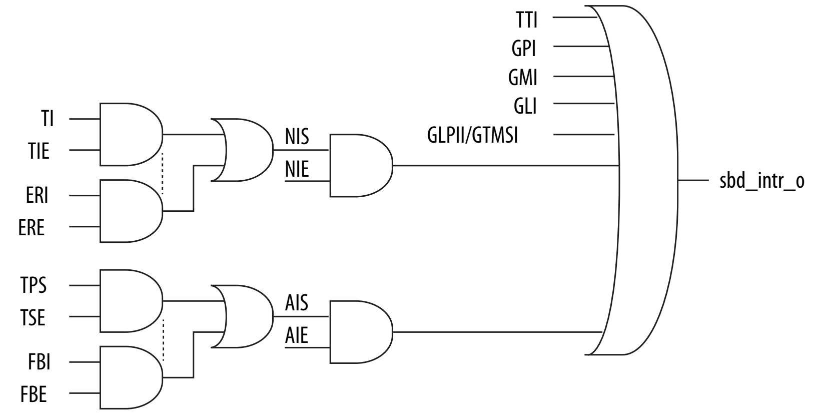
The discarding or flushing of the frame at the top of the MTL EX FIFO buffer can be avoided by disabling Flushing (Bit 24 of Register 6 (Operation Mode Register)). In such conditions, the receive process sets the Receive Buffer Unavailable status and returns to the Suspend state. †

Interrupts

Interrupts can be generated as a result of various events. The DMA Register 5 (Status Register) contains all the bits that might cause an interrupt. Register 7 (Interrupt Enable Register) contains an enable bit for each of the events that can cause an interrupt. †

There are two groups of interrupts, Normal and Abnormal, as described in Register 5 (Status Register). Interrupts are cleared by writing a 1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the sbd_intr_o interrupt signal is de-asserted. If the MAC is the cause for assertion of the interrupt, then any of the GLI, GMI, GPI, TTI, or GLPII bits of Register 5 (Status Register) are set high.

Figure 17-10: sbd_intr_o Generation[41]



Note: The Register 5 (Status Register) is the interrupt status register. The interrupt pin (sbd_intr_o) is asserted because of any event in this status register only if the corresponding interrupt enable bit is set in Register 7 (Interrupt Enable Register). †

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Bit 6 (Receive Interrupt) of Register 5 (Status Register) indicates that one or more frames were transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA. †

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the Register 5 (Status Register) for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in Register 5 (Status Register). For example, the controller generates a Bit 6 (Receive Interrupt) of Register 5 (Status Register) and the driver begins reading Register 5 (Status Register). Next, Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) occurs. The driver clears the receive interrupt. Even then, the sbd_intr_o signal is not de-asserted, because of the active or pending Receive Buffer Unavailable interrupt. †

Bits 7:0 (Interrupt Timer) of Register 9 (Receive Interrupt Watchdog Timer Register) is given for flexible control of receive Interrupt. When this Interrupt timer is programmed with a non-zero value, it gets activated as soon as the RX DMA completes a transfer of a received frame to system memory without asserting the receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31]. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RI is enabled in Register 7 (Interrupt Enable Register). This timer gets disabled before it runs out, when a frame is transferred to memory and the RI is set because it is enabled for that descriptor. †

Related Information

Receive Descriptor on page 17-46

[41] Signals NIS and AIS are registered.

Error Response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Register 5 (Status Register). The DMA controller can resume operation only after soft resetting or hard resetting the EMAC and reinitializing the DMA.

Descriptor Overview

This section describes the HPS EMAC DMA descriptor.

The DMA in the Ethernet subsystem transfers data based on a single descriptor, as explained in DMA Controller. The descriptor is created in the system memory.

The descriptor format can have 8 DWORDS (32 bytes). The descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes.

Related Information

- **DMA Controller** on page 17-25
- **Descriptors** on page 17-39
  Detailed bit map of the descriptor structure

Descriptor Endianness

The descriptor addresses must be aligned to the used bus width (Word for 32-bit bus). The data bus is configured as little-endian.

Related Information

Descriptors on page 17-39
Detailed bit map of the descriptor structure

Descriptors

The descriptor structure can have 8 DWORDS (32-bytes). The features of the descriptor structure are:

- The descriptor structure is implemented to support buffers of up to 8 KB (useful for Jumbo frames).
- There is a re-assignment of control and status bits in TDES0, TDES1, RDES0 (advanced timestamp or IPC full offload configuration), and RDES1. †
- The transmit descriptor stores the timestamp in TDES6 and TDES7 when you select the advanced timestamp. †
- This receive descriptor structure is also used for storing the extended status (RDES4) and timestamp (RDES6 and RDES7) when advanced timestamp, IPC Full Checksum Offload Engine, or Layer 3 and Layer 4 filter feature is selected. †
- You can select one of the following options for descriptor structure:
  - If timestamping is enabled in Register 448 (Timestamp Control Register) or Checksum Offload is enabled in Register 0 (MAC Configuration Register), the software needs to allocate 32-bytes (8 DWORDS) of memory for every descriptor. For this, the software should set Bit 7 (Descriptor Size) of Register 0 (Bus Mode Register).
  - If timestamping or Checksum Offload is not enabled, the extended descriptors (DES4 to DES7) are not required. Therefore, the software can use descriptors with the default size of 16 bytes (4 DWORDS). For this, the software should reset Bit 7 (Descriptor Size) of Register 0 (Bus Mode Register) to 0.
- When descriptor is selected without Timestamp or Receive IPC Full Checksum Offload Engine (Type 2) feature, the descriptor size is always 4 DWORDs (DES0-DES3). Therefore, the software can use descriptors with the default size of 16 bytes.

Transmit Descriptor

The application software must program the control bits TDES0[31:18] during the transmit descriptor initialization. When the DMA updates the descriptor, it writes back all the control bits except the OWN bit (which it clears) and updates the status bits[7:0].

With the advance timestamp support, the snapshot of the timestamp to be taken can be enabled for a given frame by setting Bit 25 (TTSE) of TDES0. When the descriptor is closed (that is, when the OWN bit is cleared), the timestamp is written into TDES6 and TDES7. This is indicated by the status Bit 17 (TTSS) of TDES0.

Note: When advanced timestamp feature is enabled, the software should set Bit 7 of Register 0 (Bus Mode Register), so that the DMA operates with extended descriptor size. When this control bit is reset, the TDES4-TDES7 descriptor space is not valid. †

Figure 17-11: Transmit Descriptor Fields - Format

<table>
<tr><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>TDES0</td><td>OWN</td><td colspan="5">Ctrl [30:26]</td><td>TTSE</td><td colspan="7">Ctrl [24:18]</td><td>TTSS</td><td colspan="10">Status [16:7]</td><td colspan="4">Ctrl/Status [6:3]</td><td colspan="3">Status [2:0]</td></tr>
<tr><td>TDES1</td><td colspan="3">Ctrl [31:29]</td><td colspan="13">Buffer 2 Byte Count [28:16]</td><td colspan="3">RES</td><td colspan="13">Buffer 1 Byte Count [12:0]</td></tr>
<tr><td>TDES2</td><td colspan="32">Buffer 1 Address [31:0]</td></tr>
<tr><td>TDES3</td><td colspan="32">Buffer 2 Address [31:0] or Next Descriptor Address [31:0]</td></tr>
<tr><td>TDES4</td><td colspan="32">Reserved</td></tr>
<tr><td>TDES5</td><td colspan="32">Reserved</td></tr>
<tr><td>TDES6</td><td colspan="32">Transmit Timestamp Low [31:0]</td></tr>
<tr><td>TDES7</td><td colspan="32">Transmit Timestamp High [31:0]</td></tr>
</table>

The DMA always reads or fetches four DWORDS of the descriptor from system memory to obtain the buffer and control information. †

Figure 17-12: Transmit Descriptor Fetch (Read) Format

<table>
<tr><th></th><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>TDES0</td><td>OWN</td><td colspan="5">Ctrl [30:26]</td><td>TTSE</td><td colspan="7">Ctrl [24:18]</td><td colspan="11">Reserved for Status [17:7]</td><td colspan="4">SLOT Number [6:3]</td><td colspan="3">Reserved for Status [2:0]</td></tr>
<tr><td>TDES1</td><td colspan="3">Ctrl [31:29]</td><td colspan="13">Buffer 2 Byte Count [28:16]</td><td colspan="3">RES</td><td colspan="13">Buffer 1 Byte Count [12:0]</td></tr>
<tr><td>TDES2</td><td colspan="32">Buffer 1 Address [31:0]</td></tr>
<tr><td>TDES3</td><td colspan="32">Buffer 2 Address [31:0] or Next Descriptor Address [31:0]</td></tr>
</table>

Table 17-8: Transmit Descriptor Word 0 (TDES0)

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit. †
30	IC: Interrupt on Completion When set, this bit sets the Transmit Interrupt (Register 5[0]) after the present frame has been transmitted. †
29	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1 or TBS2 field in TDES1 should have a non-zero value. †
28	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame. †
27	DC: Disable CRC When this bit is set, the MAC does not append a CRC to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set. †

Bit	Description
26	DP: Disable Pad When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set. †
25	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware timestamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.
24	Reserved
23:22	CIC: Checksum Insertion Control. These bits control the checksum calculation and insertion. The following list describes the bit encoding: ▪ 0: Checksum Insertion Disabled. ▪ 1: Only IP header checksum calculation and insertion are enabled. ▪ 2: IP header checksum and payload checksum calculation and insertion are enabled, but pseudoheader checksum is not calculated in hardware. ▪ 3: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudoheader checksum is calculated in hardware. This field is valid when the First Segment control bit (TDES0[28]) is set.
21	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring. †
20	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a "don't care" value. TDES0[21] takes precedence over TDES0[20]. †
19:18	Reserved
17	TTSS: Transmit Timestamp Status This field is used as a status bit to indicate that a timestamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a timestamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment control bit (TDES0[29]) is set. †

Bit	Description
16	IHE: IP Header Error When set, this bit indicates that the MAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IPheader version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5. †
15	ES: Error Summary Indicates the logical OR of the following bits: ▪ TDES0[14]: Jabber Timeout ▪ TDES0[13]: Frame Flush ▪ TDES0[11]: Loss of Carrier ▪ TDES0[10]: No Carrier ▪ TDES0[9]: Late Collision ▪ TDES0[8]: Excessive Collision ▪ TDES0[2]: Excessive Deferral ▪ TDES0[1]: Underflow Error ▪ TDES0[16]: IP Header Error ▪ TDES0[12]: IP Payload Error †
14	JT: Jabber Timeout When set, this bit indicates the MAC transmitter has experienced a jabber time-out. This bit is only set when Bit 22 (Jabber Disable) of Register 0 (MAC Configuration Register) is not set. †
13	FF: Frame Flushed When set, this bit indicates that the DMA or MTL flushed the frame because of a software Flush command given by the CPU. †
12	IPE: IP Payload Error When set, this bit indicates that MAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch. †

Bit	Description
11	LC: Loss of Carrier When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision when the MAC operates in the half-duplex mode. †
10	NC: No Carrier When set, this bit indicates that the Carrier Sense signal form the PHY was not asserted during transmission. †
9	Reserved
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If Bit 9 (Disable Retry) bit in the Register 0 (MAC Configuration Register) is set, this bit is set after the first collision, and the transmission of the frame is aborted. †
7	VF: VLAN Frame When set, this bit indicates that the transmitted frame is a VLAN-type frame. †
6:3	CC: Collision Count (Status field) These status bits indicate the number of collisions that occurred before the frame was transmitted. This count is not valid when the Excessive Collisions bit (TDES0[8]) is set. The EMAC updates this status field only in the half-duplex mode.
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbps mode or if Jumbo frame is enabled) if Bit 4 (Deferral Check) bit in Register 0 (MAC Configuration Register) is set high. †
1	UF: Underflow Error When set, this bit indicates that the MAC aborted the frame because the data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]). †
0	DB: Deferred Bit When set, this bit indicates that the MAC defers before transmission because of the presence of carrier. This bit is valid only in the half-duplex mode. †

Table 17-9: Transmit Descriptor Word 1 (TDES1)

Bit	Description
31:29	Reserved
28:16	TBS2: Transmit Buffer 2 Size This field indicates the second data buffer size in bytes. This field is not valid if TDES0[20] is set. †
15:13	Reserved †
12:0	TBS1: Transmit Buffer 1 Size These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]). †

Table 17-10: Transmit Descriptor 2 (TDES2)

Bit	Description
31:09	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. †

Table 17-11: Transmit Descriptor 3 (TDES3)

Bit	Description
31:09	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.) †

Table 17-12: Transmit Descriptor 6 (TDES6)

Bit	Description
31:09	TTSL: Transmit Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set. †

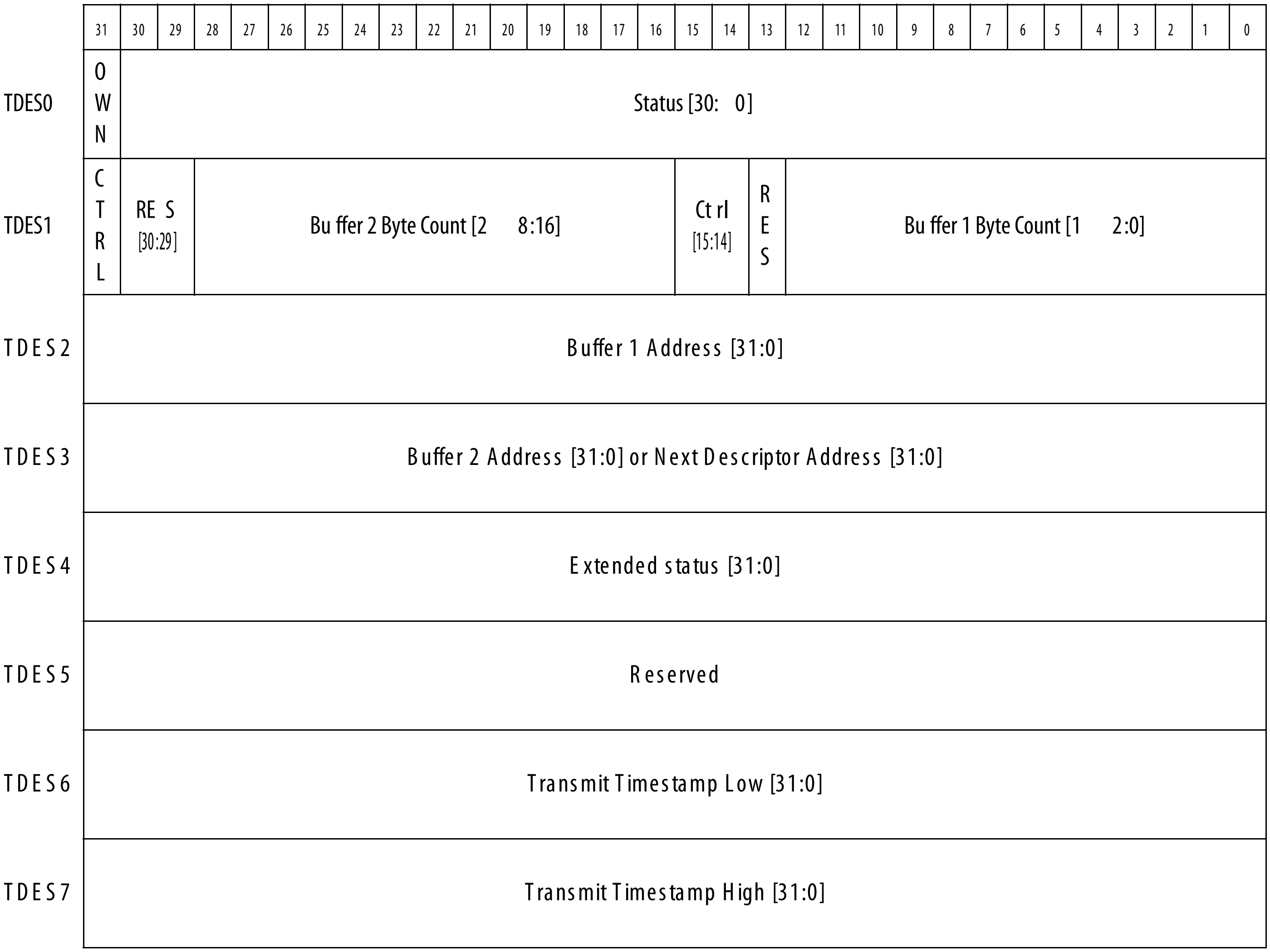
Table 17-13: Transmit Descriptor 7 (TDES7)

Bit	Description
31:09	TTSH: Transmit Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set. †

Receive Descriptor

The receive descriptor can have 32 bytes of descriptor data (8 DWORDs) when advanced timestamp or IPC Full Offload feature is selected. When either of these features is enabled, the Software should set Bit 7 of Register 0 (Bus Mode Register) so that the DMA operates with extended descriptor size. When this control bit is reset, the RDES0[0] is always cleared and the RDES4-RDES7 descriptor space is not valid. †

Figure 17-13: Receive Descriptor Fields Format



Receive Descriptor Field 0 (RDES0)

Table 17-14: Receive Descriptor Field 0 (RDES0)

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the EMAC. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the MAC. †
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame. †
15	ES: Error Summary Indicates the logical OR of the following bits: • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: Giant Frame • RDES4[4:3]: IP Header or Payload Error (Receive Descriptor Field 4 (RDES4)) • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set. †
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set. †
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC. †

Bit	Description
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. †
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged because of buffer overflow in MTL. Note: This bit is set only when the DMA transfers a partial frame to the application. This happens only when the RX FIFO buffer is operating in the threshold mode. In the store-and-forward mode, all partial frames are dropped completely in RX FIFO buffer. †
10	VLAN: VLAN Tag When set, this bit indicates that the frame to which this descriptor is pointing is a VLAN frame tagged by the MAC. The VLAN tagging depends on checking the VLAN fields of received frame based on the Register 7 (VLAN Tag Register) setting. †
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next descriptor contains the beginning of the frame. †
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame †
7	Timestamp Available, IP Checksum Error (Type1), or Giant Frame When advanced timestamp feature is present, when set, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When IP Checksum Engine (Type 1) is selected, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the EMAC did not match the received checksum bytes. Otherwise, this bit, when set, indicates the Giant frame Status. Giant frames are larger than 1,518-byte (or 1,522-byte for VLAN or 2,000-byte when Bit 27 (2KPE) of MAC Configuration register is set) normal frames and larger than 9,018-byte (9,022-byte for VLAN) frame when Jumbo frame processing is enabled.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in the half-duplex mode. †

Bit	Description
5	FT: Frame Type When set, this bit indicates that the receive frame is an Ethernet-type frame (the LT field is greater than or equal to 0x0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes.
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout. †
3	RE: Receive Error When set, this bit indicates that the `gmii_rxer_i` signal is asserted while `gmii_rxdv_i` is asserted during frame reception.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode. †
1	CE: CRC Error When set, this bit indicates that a CRC error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set. †
0	Extended Status Available/RX MAC Address When either advanced timestamp or IP Checksum Offload (Type 2) is present, this bit, when set, indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When Advance Timestamp Feature or IPC Full Offload is not selected, this bit indicates RX MAC Address status. When set, this bit indicates that the RX MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the RX MAC Address Register 0 value matched the DA field. †

Related Information

- Receive Descriptor Field 4 (RDES4) on page 17-51
- Receive Descriptor Field 6 (RDES6) on page 17-54
- Receive Descriptor Field 7 (RDES7) on page 17-55

Receive Descriptor Field 1 (RDES1)

Table 17-15: Receive Descriptor Field 1 (RDES1)

Bit	Description
31	DIC: Disable Interrupt on Completion When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host because of RI for that frame. †
30:29	Reserved †
28:16	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, even if the value of RDES3 (buffer2 address pointer) in Receive Descriptor Field 3 (RDES3) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. For more information about calculating buffer sizes, refer to `Buffer Size Calculations`.
15	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring. †
14	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14]. †
13	Reserved †
12:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, even if the value of RDES2 (buffer1 address pointer), in Receive Descriptor Field 2 (RDES2), is not aligned. When the buffer size is not a multiple of 4, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14). For more information about calculating buffer sizes, refer to `Buffer Size Calculations`.

Related Information

- Buffer Size Calculations on page 17-28
- Receive Descriptor Field 2 (RDES2) on page 17-51
- Receive Descriptor Field 3 (RDES3) on page 17-51

Receive Descriptor Fields (RDES2) and (RDES3)

Receive Descriptor Field 2 (RDES2)

Table 17-16: Receive Descriptor Field 2 (RDES2)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the value programmed in RDES2[1:0] for its address generation when the RDES2 value is used to store the start of the frame. The DMA performs a write operation with the RDES2[1:0] bits as 0 during the transfer of the start of the frame but the frame is shifted as per the actual buffer address pointer. The DMA ignores RDES2[1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored. For more information about buffer address alignment, refer to `Host Data Buffer Alignment`.

Receive Descriptor Field 3 (RDES3)

Table 17-17: Receive Descriptor Field 3 (RDES3)

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit, in Receive Descriptor Field 1 (RDES1), is set, this address contains the pointer to the physical memory where the Next descriptor is present. If RDES1[24], in Receive Descriptor Field 1 (RDES1), is set, the buffer (Next descriptor) address pointer must be bus width-aligned (RDES3[1:0] = 0. LSBs are ignored internally.) However, when RDES1[24], in Receive Descriptor Field 1 (RDES1), is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the value programmed in RDES3 [1:0] for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3 [1:0] if the address pointer is to a buffer where the middle or last part of the frame is stored.

Related Information

Receive Descriptor Field 1 (RDES1) on page 17-50

Receive Descriptor Field 4 (RDES4)

The extended status is written only when there is status related to IPC or timestamp available. The availability of extended status is indicated by Bit 0 in RDES0. This status is available only when the Advance Timestamp or IPC Full Offload feature is selected.

Table 17-18: Receive Descriptor Field 4 (RDES4)

Bit	Description
31:28	Reserved †
27:26	Layer 3 and Layer 4 Filter Number Matched These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received frame. • 00: Filter 0 • 01: Filter 1 • 10: Filter 2 • 11: Filter 3 This field is valid only when Bit 24 or Bit 25 is set high. When more than one filter matches, these bits give only the lowest filter number. †
25	Layer 4 Filter Match When set, this bit indicates that the received frame matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: • Layer 3 fields are not enabled and all enabled Layer 4 fields match. • All enabled Layer 3 and Layer 4 filter fields match. When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits [27:26]. †
24	Layer 3 Filter Match When set, this bit indicates that the received frame matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: • All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed. • All enabled filter fields match. When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by Bits [27:26]. †
23:15	Reserved
14	Timestamp Dropped When set, this bit indicates that the timestamp was captured for this frame but got dropped in the MTL RX FIFO buffer because of overflow.
13	PTP Version When set, this bit indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format.

Bit	Description
12	PTP Frame Type When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information about IPv4 or IPv6 can be obtained from Bits 6 and 7.
11:8	Message Type These bits are encoded to give the type of the message received. • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011-1110: Reserved • 1111: PTP packet with Reserved message type
7	IPv6 Packet Received When set, this bit indicates that the received packet is an IPv6 packet. This bit is updated only when Bit 10 (IPC) of Register 0 (MAC Configuration Register) is set.
6	IPv4 Packet Received When set, this bit indicates that the received packet is an IPv4 packet. This bit is updated only when Bit 10 (IPC) of Register 0 (MAC Configuration Register) is set.
5	IP Checksum Bypassed When set, this bit indicates that the checksum offload engine is bypassed.
4	IP Payload Error When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the EMAC calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. This bit is valid when either Bit 7 or Bit 6 is set.

Bit	Description
3	IP Header Error When set, this bit indicates that either the 16-bit IPv4 header checksum calculated by the EMAC does not match the received checksum bytes, or the IP datagram version is not consistent with the Ethernet Type value. This bit is valid when either Bit 7 or Bit 6 is set.
2:0	IP Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the receive Checksum Offload Engine (COE). The COE also sets these bits to 0 if it does not process the IP datagram's payload due to an IP header error or fragmented IP. • 0: Unknown or did not process IP payload • 1: UDP • 2: TCP • 3: ICMP • 4–7: Reserved This bit is valid when either Bit 7 or Bit 6 is set.

Related Information

Receive Descriptor Field 0 (RDES0) on page 17-47

Receive Descriptor Fields (RDES6) and (RDES7)

Receive Descriptor Fields 6 (RDES6) and RDES7 contain the snapshot of the timestamp. The availability of the snapshot of the timestamp in RDES6 and RDES7 is indicated by Bit 7 in the RDES0 descriptor.

Related Information

Receive Descriptor Field 0 (RDES0) on page 17-47

Receive Descriptor Field 6 (RDES6)

Table 17-19: Receive Descriptor Field 6 (RDES6)

Bit	Description
31:0	RTSL: Receive Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]) in RDES0.

Related Information

Receive Descriptor Field 0 (RDES0) on page 17-47

Receive Descriptor Field 7 (RDES7)

Table 17-20: Receive Descriptor Field 7 (RDES7)

Bit	Description
31:0	RTSH: Receive Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]) in RDES0.

Related Information

Receive Descriptor Field 0 (RDES0) on page 17-47

Initializing DMA

This section provides the instructions for initializing the DMA/MAC registers in the proper sequence. Perform the following steps to initialize the DMA:

1. Provide a software reset. This resets all of the EMAC internal registers and logic. (DMA Register 0 (Bus Mode Register) – bit 0). †
2. Wait for the completion of the reset process (poll bit 0 of the DMA Register 0 (Bus Mode Register), which is only cleared after the reset operation is completed). †
3. Poll the bits of Register 11 (AHB or AXI Status) to confirm that all previously initiated (before software-reset) or ongoing transactions are complete.

   Note: If the application cannot poll the register after soft reset (because of performance reasons), then it is recommended that you continue with the next steps and check this register again (as mentioned in 12 on page 1-56) before triggering the DMA operations.†
4. Program the following fields to initialize the Bus Mode Register by setting values in DMA Register 0 (Bus Mode Register):†
   - Mixed Burst and AAL
   - Fixed burst or undefined burst†
   - Burst length values and burst mode values†
   - Descriptor Length (only valid if Ring Mode is used)†
   - TX and RX DMA Arbitration scheme†
5. Program the interface options in Register 10 (AXI Bus Mode Register). If fixed burst-length is enabled, then select the maximum burst-length possible on the bus (bits[7:1]).†
6. Create a proper descriptor chain for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (bit 31 of descriptor should be set). When OSF mode is used, at least two descriptors are required.
7. Make sure that your software creates three or more different transmit or receive descriptors in the chain before reusing any of the descriptors.†
8. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (Register 3 (Receive Descriptor List Address Register) and Register 4 (Transmit Descriptor List Address Register) respectively).†
9. Program the following fields to initialize the mode of operation in Register 6 (Operation Mode Register):

- Receive and Transmit Store And Forward†
- Receive and Transmit Threshold Control (RTC and TTC)†
- Hardware Flow Control enable†
- Flow Control Activation and De-activation thresholds for MTL Receive and Transmit FIFO buffers (RFA and RFD)†
- Error frame and undersized good frame forwarding enable†
- OSF Mode†

10. Clear the interrupt requests, by writing to those bits of the status register (interrupt bits only) that are set. For example, by writing 1 into bit 16, the normal interrupt summary clears this bit (DMA Register 5 (Status Register)).†
11. Enable the interrupts by programming the Register 7 (Interrupt Enable Register).†

    Note: Perform 12 on page 1-56 only if you did not perform 3 on page 1-55.†
12. Read Register 11 (AHB or AXI Status) to confirm that all previous transactions are complete.†

    Note: If any previous transaction is still in progress when you read the Register 11 (AHB or AXI Status), then it is strongly recommended to check the slave components addressed by the master interface.†
13. Start the receive and transmit DMA by setting SR (bit 1) and ST (bit 13) of the control register (DMA Register 6 (Operation Mode Register). †

Related Information

Descriptors on page 17-39

Detailed bit map of the descriptor structure

Initializing MAC

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is done before the DMA is set-up, then enable the MAC receiver (last step below) only after the DMA is active. Otherwise, received frames fill the RX FIFO buffer and overflow.

1. Program the EMAC Register 4 (GMII Address Register) for controlling the management cycles for external PHY. For example, Physical Layer Address PA (bits 15-11). In addition, set bit 0 (GMII Busy) for writing into PHY and reading from PHY. †
2. Read the 16-bit data of Register 5 (GMII Data Register) from the PHY for link up, speed of operation, and mode of operation, by specifying the appropriate address value in bits 15-11 of Register 4 (GMII Address Register). †
3. Provide the MAC address registers (Register 16 (MAC Address0 High Register) and Register 17 (MAC Address0 Low Register)). Because 128 MAC addresses are supported, you need to program the MAC addresses accordingly.
4. Program Register 2 (Hash Table High Register) and Register 3 (Hash Table Low Register).
5. Program the following fields to set the appropriate filters for the incoming frames in Register 1 (MAC Frame Filter): †

    - Receive All †
    - Promiscuous mode †
    - Hash or Perfect Filter †
    - Unicast, multicast, broadcast, and control frames filter settings †
6. Program the following fields for proper flow control in Register 6 (Flow Control Register): †

    - Pause time and other pause frame control bits †
    - Receive and Transmit Flow control bits †
    - Flow Control Busy/Backpressure Activate †

7. Program the Interrupt Mask register bits, as required, and if applicable, for your configuration. †
8. Program the appropriate fields in Register 0 (MAC Configuration Register). For example, Interframe gap while transmission and jabber disable. Based on the Auto-negotiation you can set the Duplex mode (bit 11) or port select (bit 15). †
9. Set Bit 3 (TE) and Bit 2 (RE) in Register 0 (MAC Configuration Register). †

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the EMAC DMA is actively transmitting or receiving. The Software should change these parameters only when the EMAC DMA transmitter and receiver are not active.

Performing Normal Receive and Transmit Operation

For normal operation, perform the following steps: †

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive). †
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data. †
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and issuing a poll demand by writing 0 into the TX/RX poll demand register (Register 1 (Transmit Poll Demand Register) and Register 2 (Receive Poll Demand Register). †
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process (Register 18 (Current Host Transmit Descriptor Register) and Register 19 (Current Host Receive Descriptor Register). †
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (Register 20 (Current Host Transmit Buffer Address Register) and Register 21 (Current Host Receive Buffer Address Register). †

Stopping and Starting Transmission

Perform the following steps to pause the transmission for some time: †

1. Disable the transmit DMA (if applicable), by clearing bit 13 (Start or Stop Transmission Command) of Register 6 (Operation Mode Register). †
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of Register 9 (Debug Register). †
3. Disable the MAC transmitter and MAC receiver by clearing Bit 3 (TE) and Bit 2 (RE) in Register 0 (MAC Configuration Register). †
4. Disable the receive DMA (if applicable), after making sure that the data in the RX FIFO buffer is transferred to the system memory (by reading Register 9 (Debug Register). †
5. Make sure that both TX FIFO buffer and RX FIFO buffer are empty. †
6. To re-start the operation, first start the DMAs, and then enable the MAC transmitter and receiver. †

Programming Guidelines for Energy Efficient Ethernet

Entering and Exiting the TX LPI Mode

The Energy Efficient Ethernet (EEE) feature is available in the EMAC. To use it, perform the following steps during EMAC initialization:

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values. †
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop RX clock in LPI mode.) †
3. Program Bits[16:5], LST, and Bits[15:0], TWT, in Register 13 (LPI Timers Control Register). †
4. Read the link status of the PHY chip by using the MDIO interface and update Bit 17 (PLS) of Register 12 (LPI Control and Status Register) accordingly. This update should be done whenever the link status in the PHY chip changes. †
5. Set Bit 16 (LPIEN) of Register 12 (LPI Control and Status Register) to make the MAC enter the LPI state. The MAC enters the LPI mode after completing the transmission in progress and sets Bit 0 (TLPIEN). †

   Note: To make the MAC enter the LPI state only after it completes the transmission of all queued frames in the TX FIFO buffer, you should set Bit 19 (LPITXA) in Register 12 (LPI Control and Status Register). †

   Note: To switch off the transmit clock during the LPI state, use the `sbd_tx_clk_gating_ctrl_o` signal for gating the clock input. †

   Note: To switch off the CSR clock or power to the rest of the system during the LPI state, you should wait for the TLPIEN interrupt of Register 12 (LPI Control and Status Register) to be generated. Restore the clocks before performing the 6 on page 1-58 when you want to come out of the LPI state. †
6. Reset Bit 16 (LPIEN) of Register 12 (LPI Control and Status Register) to bring the MAC out of the LPI state. †

The MAC waits for the time programmed in Bits [15:0], TWT, before setting the TLPIEX interrupt status bit and resuming the transmission. †

Gating Off the CSR Clock in the LPI Mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.†

Gating Off the CSR Clock in the RX LPI Mode

The following operations are performed when the MAC receives the LPI pattern from the PHY. †

1. The MAC RX enters the LPI mode and the RX LPI entry interrupt status [RLPIEN interrupt of Register 12 (`LPI_Control_Status`)] is set. †
2. The interrupt pin (`sbd_intr_o`) is asserted. The `sbd_intr_o` interrupt is cleared when the host reads the Register 12 (`LPI_Control_Status`). †

After the `sbd_intr_o` interrupt is asserted and the MAC TX is also in the LPI mode, you can gate-off the CSR clock. If the MAC TX is not in the LPI mode when you gate off the CSR clock, the events on the MAC transmitter do not get reported or updated in the CSR. †

For restoring the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on `lpi_intr_o` (synchronous to `clk_rx_i`). The `lpi_intr_o` interrupt is cleared when Register 12 is read. †

Gating Off the CSR Clock in the TX LPI Mode

The following operations are performed when Bit 16 (LPIEN) of Register 12 (LPI Control and Status Register) is set: †

1. The Transmit LPI Entry interrupt (TLPIEN bit of Register 12) is set. †
2. The interrupt pin (`sbd_intr_o`) is asserted. The `sbd_intr_o` interrupt is cleared when the host reads the Register 12. †

After the `sbd_intr_o` interrupt is asserted and the MAC RX is also in the LPI mode, you can gate off the CSR clock. If the MAC RX is not in the LPI mode when you gate off the CSR clock, the events on the MAC receiver do not get reported or updated in the CSR. †

For restoring the CSR clock, switch on the CSR clock when the MAC has to come out of the TX LPI mode. †

After the CSR clock is resumed, reset Bit 16 (LPIEN) of Register 12 (LPI Control and Status Register) to bring the MAC out of the LPI mode. †

Programming Guidelines for Flexible Pulse-Per-Second (PPS) Output

Generating Single Pulse on PPS

To generate single Pulse on PPS: †

1. Program 11 or 10 (for interrupt) in Bits [6:5], TRGTMODSEL, of Register 459 (PPS Control Register). This instructs the MAC to use the Target Time registers (register 455 and 456) for start time of PPS signal output. †
2. Program the start time value in the Target Time registers (register 455 and 456). †
3. Program the width of the PPS signal output in Register 473 (PPS0 Width Register). †
4. Program Bits [3:0], PPSCMD, of Register 459 (PPS Control Register) to 0001. This instructs the MAC to generate single pulse on the PPS signal output at the time programmed in the Target Time registers (register 455 and 456). †

Once the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time elapses. You can also program the behavior of the next pulse in advance. To program the next pulse: †

1. Program the start time for the next pulse in the Target Time registers (register 455 and 456). This time should be more than the time at which the falling edge occurs for the previous pulse. †
2. Program the width of the next PPS signal output in Register 473 (PPS0 Width Register). †
3. Program Bits [3:0], PPSCMD, of Register 459 (PPS Control Register) to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programmed in Target Time registers. If you give this command before the previous pulse becomes low, then the new command overwrites the previous command and the EMAC may generate only 1 extended pulse.

Generating a Pulse Train on PPS

To generate a pulse train on PPS: †

1. Program 11 or 10 (for interrupt) in Bits [6:5], TRGTMODSEL, of Register 459 (PPS Control Register). This instructs the MAC to use the Target Time registers (register 455 and 456) for start time of the PPS signal output. †
2. Program the start time value in the Target Time registers (register 455 and 456). †
3. Program the interval value between the train of pulses on the PPS signal output in Register 473 (PPS0 Width Register). †
4. Program the width of the PPS signal output in Register 473 (PPS0 Width Register). †
5. Program Bits[3:0], PPSCMD, of Register 459 (PPS Control Register) to 0010. This instructs the MAC to generate train of pulses on the PPS signal output with start time programmed in Target Time registers (register 455 and 456). By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands. †
6. Program the stop value in the Target Time registers (register 455 and 456). Ensure that Bit 31 (TSTRBUSY) of Register 456 (Target Time Nanoseconds Register) is reset before programming the Target Time registers (register 455 and 456) again. †
7. Program the PPSCMD field (bit 3:0) of Register 459 (PPS Control Register) to 0100. This stops the train of pulses on PPS signal output after the programmed stop time specified in 6 on page 1-60 elapses. †

You can stop the pulse train at any time by programming 0101 in the PPSCMD field. Similarly, you can cancel the Stop Pulse train command (given in 7 on page 1-60) by programming 0110 in the PPSCMD field before the time (programmed in 6 on page 1-60) elapses. You can cancel the pulse train generation by programming 0011 in the PPSCMD field before the programmed start time (in 2 on page 1-60) elapses. †

Generating an Interrupt without Affecting the PPS

The Bits [6:5], TRGTMODSEL, of the Register 459 (PPS Control Register) enable you to program the Target Time registers (register 455 and 456) to do any one of the following: †

- Generate only interrupts. †
- Generate interrupts and the PPS start and stop time. †
- Generate only PPS start and stop time. †

To program the Target Time registers (register 455 and 456) to generate only interrupt event: †

1. Program 00 (for interrupt) in Bits [6:5], TRGTMODSEL, of Register 459 (PPS Control Register). This instructs the MAC to use the Target Time registers (register 455 and 456) for target time interrupt. †
2. Program a target time value in the Target Time registers (register 455 and 456). This instructs the MAC to generate an interrupt when the target time elapses. If Bits [6:5], TRGTMODSEL, are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programmed Target Time register value.

Ethernet MAC Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- EMAC Module 0
- EMAC Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

EMAC Module Address Map

Registers in the EMAC module.

Module Instance	Base Address
emac0	0xFF700000
emac1	0xFF702000

GMAC Register Group

Register	Offset	Width	Access	Reset Value	Description
MAC_Configuration on page 17-126	0x0	32	RW	0x0	Register 0 (MAC Configuration Register)
MAC_Frame_Filter on page 17-134	0x4	32	RW	0x0	Register 1 (MAC Frame Filter)
GMII_Address on page 17-140	0x10	32	RW	0x0	Register 4 (GMII Address Register)
GMII_Data on page 17-142	0x14	32	RW	0x0	Register 5 (GMII Data Register)
Flow_Control on page 17-143	0x18	32	RW	0x0	Register 6 (Flow Control Register)
VLAN_Tag on page 17-146	0x1C	32	RW	0x0	Register 7 (VLAN Tag Register)
Version on page 17-148	0x20	32	RO	0x1037	Register 8 (Version Register)
Debug on page 17-148	0x24	32	RO	0x0	Register 9 (Debug Register)
LPI_Control_Status on page 17-152	0x30	32	RW	0x0	Register 12 (LPI Control and Status Register)
LPI_Timers_Control on page 17-155	0x34	32	RW	0x3E80000	Register 13 (LPI Timers Control Register)
Interrupt_Status on page 17-156	0x38	32	RO	0x0	Register 14 (Interrupt Register)
Interrupt_Mask on page 17-159	0x3C	32	RW	0x0	Register 15 (Interrupt Mask Register)
MAC_Address0_High on page 17-160	0x40	32	RW	0x8000FFFF	Register 16 (MAC Address0 High Register)
MAC_Address0_Low on page 17-161	0x44	32	RW	0xFFFFFFFF	Register 17 (MAC Address0 Low Register)
MAC_Address1_High on page 17-162	0x48	32	RW	0xFFFF	Register 18 (MAC Address1 High Register)
MAC_Address1_Low on page 17-165	0x4C	32	RW	0xFFFFFFFF	Register 19 (MAC Address1 Low Register)
MAC_Address2_High on page 17-166	0x50	32	RW	0xFFFF	Register 20 (MAC Address2 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address2_Low on page 17-169	0x54	32	RW	0xFFFFFFFF	Register 21 (MAC Address2 Low Register)
MAC_Address3_High on page 17-170	0x58	32	RW	0xFFFF	Register 22 (MAC Address3 High Register)
MAC_Address3_Low on page 17-173	0x5C	32	RW	0xFFFFFFFF	Register 23 (MAC Address3 Low Register)
MAC_Address4_High on page 17-174	0x60	32	RW	0xFFFF	Register 24 (MAC Address4 High Register)
MAC_Address4_Low on page 17-177	0x64	32	RW	0xFFFFFFFF	Register 25 (MAC Address4 Low Register)
MAC_Address5_High on page 17-178	0x68	32	RW	0xFFFF	Register 26 (MAC Address5 High Register)
MAC_Address5_Low on page 17-181	0x6C	32	RW	0xFFFFFFFF	Register 27 (MAC Address5 Low Register)
MAC_Address6_High on page 17-182	0x70	32	RW	0xFFFF	Register 28 (MAC Address6 High Register)
MAC_Address6_Low on page 17-185	0x74	32	RW	0xFFFFFFFF	Register 29 (MAC Address6 Low Register)
MAC_Address7_High on page 17-186	0x78	32	RW	0xFFFF	Register 30 (MAC Address7 High Register)
MAC_Address7_Low on page 17-189	0x7C	32	RW	0xFFFFFFFF	Register 31 (MAC Address7 Low Register)
MAC_Address8_High on page 17-190	0x80	32	RW	0xFFFF	Register 32 (MAC Address8 High Register)
MAC_Address8_Low on page 17-193	0x84	32	RW	0xFFFFFFFF	Register 33 (MAC Address8 Low Register)
MAC_Address9_High on page 17-194	0x88	32	RW	0xFFFF	Register 34 (MAC Address9 High Register)
MAC_Address9_Low on page 17-197	0x8C	32	RW	0xFFFFFFFF	Register 35 (MAC Address9 Low Register)
MAC_Address10_High on page 17-198	0x90	32	RW	0xFFFF	Register 36 (MAC Address10 High Register)
MAC_Address10_Low on page 17-201	0x94	32	RW	0xFFFFFFFF	Register 37 (MAC Address10 Low Register)
MAC_Address11_High on page 17-202	0x98	32	RW	0xFFFF	Register 38 (MAC Address11 High Register)
MAC_Address11_Low on page 17-205	0x9C	32	RW	0xFFFFFFFF	Register 39 (MAC Address11 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address12_High on page 17-206	0xA0	32	RW	0xFFFF	Register 40 (MAC Address12 High Register)
MAC_Address12_Low on page 17-209	0xA4	32	RW	0xFFFFFFFF	Register 41 (MAC Address12 Low Register)
MAC_Address13_High on page 17-210	0xA8	32	RW	0xFFFF	Register 42 (MAC Address13 High Register)
MAC_Address13_Low on page 17-213	0xAC	32	RW	0xFFFFFFFF	Register 43 (MAC Address13 Low Register)
MAC_Address14_High on page 17-214	0xB0	32	RW	0xFFFF	Register 44 (MAC Address14 High Register)
MAC_Address14_Low on page 17-217	0xB4	32	RW	0xFFFFFFFF	Register 45 (MAC Address14 Low Register)
MAC_Address15_High on page 17-218	0xB8	32	RW	0xFFFF	Register 46 (MAC Address15 High Register)
MAC_Address15_Low on page 17-221	0xBC	32	RW	0xFFFFFFFF	Register 47 (MAC Address15 Low Register)
SGMII_RGMII_SMII_Control_Status on page 17-222	0xD8	32	RO	0x0	Register 54 (SGMII/RGMII/SMII Status Register)
MMC_Control on page 17-223	0x100	32	RW	0x0	Register 64 (MMC Control Register)
MMC_Receive_Interrupt on page 17-225	0x104	32	RO	0x0	Register 65 (MMC Receive Interrupt Register)
MMC_Transmit_Interrupt on page 17-231	0x108	32	RO	0x0	Register 66 (MMC Transmit Interrupt Register)
MMC_Receive_Interrupt_Mask on page 17-236	0x10C	32	RW	0x0	Register 67 (MMC Receive Interrupt Mask Register)
MMC_Transmit_Interrupt_Mask on page 17-242	0x110	32	RW	0x0	Register 68 (MMC Transmit Interrupt Mask Register)
txoctetcount_gb on page 17-248	0x114	32	RO	0x0	Register 69 (txoctetcount_gb Register)
txframecount_gb on page 17-249	0x118	32	RO	0x0	Register 70 (txframecount_gb Register)
txbroadcastframes_g on page 17-249	0x11C	32	RO	0x0	Register 71 (txbroadcastframes_g Register)
txmulticastframes_g on page 17-250	0x120	32	RO	0x0	Register 72 (txmulticastframes_g Register)
tx64octets_gb on page 17-250	0x124	32	RO	0x0	Register 73 (tx64octets_gb Register)

Register	Offset	Width	Acces s	Reset Value	Description
tx65to127octets_gb on page 17-251	0x128	32	RO	0x0	Register 74 (tx65to127octets_gb Register)
tx128to255octets_gb on page 17-251	0x12C	32	RO	0x0	Register 75 (tx128to255octets_gb Register)
tx256to511octets_gb on page 17-252	0x130	32	RO	0x0	Register 76 (tx256to511octets_gb Register)
tx512to1023octets_gb on page 17-253	0x134	32	RO	0x0	Register 77 (tx512to1023octets_gb Register)
tx1024tomaxoctets_gb on page 17-253	0x138	32	RO	0x0	Register 78 (tx1024tomaxoctets_gb Register)
txunicastframes_gb on page 17-254	0x13C	32	RO	0x0	Register 79 (txunicastframes_gb Register)
txmulticastframes_gb on page 17-254	0x140	32	RO	0x0	Register 80 (txmulticastframes_gb Register)
txbroadcastframes_gb on page 17-255	0x144	32	RO	0x0	Register 81 (txbroadcastframes_gb Register)
txunderflowerror on page 17-255	0x148	32	RO	0x0	Register 82 (txunderflowerror Register)
txsinglecol_g on page 17-256	0x14C	32	RO	0x0	Register 83 (txsinglecol_g Register)
txmulticol_g on page 17-256	0x150	32	RO	0x0	Register 84 (txmulticol_g Register)
txdeferred on page 17-257	0x154	32	RO	0x0	Register 85 (txdeferred Register)
txlatecol on page 17-257	0x158	32	RO	0x0	Register 86 (txlatecol Register)
txexesscol on page 17-258	0x15C	32	RO	0x0	Register 87 (txexesscol Register)
txcarriererr on page 17-258	0x160	32	RO	0x0	Register 88 (txcarriererror Register)
txoctetcnt on page 17-259	0x164	32	RO	0x0	Register 89 (txoctetcount_g Register)
txframecount_g on page 17-260	0x168	32	RO	0x0	Register 90 (txframecount_g Register)
txexcessdef on page 17-260	0x16C	32	RO	0x0	Register 91 (txexcessdef Register)
txpauseframes on page 17-261	0x170	32	RO	0x0	Register 92 (txpauseframes Register)

Register	Offset	Width	Acces s	Reset Value	Description
txvlanframes_g on page 17-261	0x174	32	RO	0x0	Register 93 (txvlanframes_g Register)
txoversize_g on page 17-262	0x178	32	RO	0x0	Register 94 (txoversize_g Register)
rxframecount_gb on page 17-262	0x180	32	RO	0x0	Register 95 (rxframecount_gb Register)
rxoctetcount_gb on page 17-263	0x184	32	RO	0x0	Register 97 (rxoctetcount_gb Register)
rxoctetcount_g on page 17-263	0x188	32	RO	0x0	Register 98 (rxoctetcount_g Register)
rxbroadcastframes_g on page 17-264	0x18C	32	RO	0x0	Register 99 (rxbroadcastframes_g Register)
rxmulticastframes_g on page 17-264	0x190	32	RO	0x0	Register 100 (rxmulticastframes_g Register)
rxcrcerror on page 17-265	0x194	32	RO	0x0	Register 101 (rxcrcerror Register)
rxalignmenterror on page 17-265	0x198	32	RO	0x0	Register 102 (rxalignmenterror Register)
rxrunterror on page 17-266	0x19C	32	RO	0x0	Register 103 (rxrunterror Register)
rxjabbererror on page 17-266	0x1A0	32	RO	0x0	Register 104 (rxjabbererror Register)
rxundersize_g on page 17-267	0x1A4	32	RO	0x0	Register 105 (rxundersize_g Register)
rxoversize_g on page 17-268	0x1A8	32	RO	0x0	Register 106 (rxoversize_g Register)
rx64octets_gb on page 17-268	0x1AC	32	RO	0x0	Register 107 (rx64octets_gb Register)
rx65to127octets_gb on page 17-269	0x1B0	32	RO	0x0	Register 108 (rx65to127octets_gb Register)
rx128to255octets_gb on page 17-269	0x1B4	32	RO	0x0	Register 109 (rx128to255octets_gb Register)
rx256to511octets_gb on page 17-270	0x1B8	32	RO	0x0	Register 110 (rx256to511octets_gb Register)
rx512to1023octets_gb on page 17-270	0x1BC	32	RO	0x0	Register 111 (rx512to1023octets_gb Register)
rx1024tomaxoctets_gb on page 17-271	0x1C0	32	RO	0x0	Register 112 (rx1024tomaxoctets_gb Register)

Register	Offset	Width	Acces s	Reset Value	Description
rxunicastframes_g on page 17-272	0x1C4	32	RO	0x0	Register 113 (rxunicastframes_g Register)
rxlengtherror on page 17-272	0x1C8	32	RO	0x0	Register 114 (rxlengtherror Register)
rxoutofrangetype on page 17-273	0x1CC	32	RO	0x0	Register 115 (rxoutofrangetype Register)
rxpauseframes on page 17-273	0x1D0	32	RO	0x0	Register 116 (rxpauseframes Register)
rxfifooverflow on page 17-274	0x1D4	32	RO	0x0	Register 117 (rxfifooverflow Register)
rxvlanframes_gb on page 17-274	0x1D8	32	RO	0x0	Register 118 (rxvlanframes_gb Register)
rxwatchdogerror on page 17-275	0x1DC	32	RO	0x0	Register 119 (rxwatchdogerror Register)
rxrcverror on page 17-275	0x1E0	32	RO	0x0	Register 120 (rxrcverror Register)
rxctrlframes_g on page 17-276	0x1E4	32	RO	0x0	Register 121 (rxctrlframes_g Register)
MMC_IPC_Receive_Interrupt_Mask on page 17-276	0x200	32	RW	0x0	Register 128 (MMC Receive Checksum Offload Interrupt Mask Register)
MMC_IPC_Receive_Interrupt on page 17-283	0x208	32	RO	0x0	Register 130 (MMC Receive Checksum Offload Interrupt Register)
rxipv4_gd_frms on page 17-289	0x210	32	RO	0x0	Register 132 (rxipv4_gd_frms Register)
rxipv4_hdrerr_frms on page 17-290	0x214	32	RO	0x0	Register 133 (rxipv4_hdrerr_frms Register)
rxipv4_nopay_frms on page 17-290	0x218	32	RO	0x0	Register 134 (rxipv4_nopay_frms Register)
rxipv4_frag_frms on page 17-291	0x21C	32	RO	0x0	Register 135 (rxipv4_frag_frms Register)
rxipv4_udsbl_frms on page 17-291	0x220	32	RO	0x0	Register 136 (rxipv4_udsbl_frms Register)
rxipv6_gd_frms on page 17-292	0x224	32	RO	0x0	Register 137 (rxipv6_gd_frms Register)
rxipv6_hdrerr_frms on page 17-292	0x228	32	RO	0x0	Register 138 (rxipv6_hdrerr_frms Register)
rxipv6_nopay_frms on page 17-293	0x22C	32	RO	0x0	Register 139 (rxipv6_nopay_frms)

Register	Offset	Width	Acces s	Reset Value	Description
rxudp_gd_frms on page 17-293	0x230	32	RO	0x0	Register 140 (rxudp_gd_frms Register)
rxudp_err_frms on page 17-294	0x234	32	RO	0x0	Register 141 (rxudp_err_frms Register)
rxtcp_gd_frms on page 17-295	0x238	32	RO	0x0	Register 142 (rxtcp_gd_frms Register)
rxtcp_err_frms on page 17-295	0x23C	32	RO	0x0	Register 143 (rxtcp_err_frms Register)
rxicmp_gd_frms on page 17-296	0x240	32	RO	0x0	Register 144 (rxicmp_gd_frms Register)
rxicmp_err_frms on page 17-296	0x244	32	RO	0x0	Register 145 (rxicmp_err_frms Register)
rxipv4_gd_octets on page 17-297	0x250	32	RO	0x0	Register 148 (rxipv4_gd_octets Register)
rxipv4_hdrerr_octets on page 17-297	0x254	32	RO	0x0	Register 149 (rxipv4_hdrerr_octets)
rxipv4_nopay_octets on page 17-298	0x258	32	RO	0x0	Register 150 (rxipv4_nopay_octets Register)
rxipv4_frag_octets on page 17-299	0x25C	32	RO	0x0	Register 151 (rxipv4_frag_octets Register)
rxipv4_udsbl_octets on page 17-299	0x260	32	RO	0x0	Register 152 (rxipv4_udsbl_octets Register)
rxipv6_gd_octets on page 17-300	0x264	32	RO	0x0	Register 153 (rxipv6_gd_octets Register)
rxipv6_hdrerr_octets on page 17-300	0x268	32	RO	0x0	Register 154 (rxipv6_hdrerr_octets Register)
rxipv6_nopay_octets on page 17-301	0x26C	32	RO	0x0	Register 155 (rxipv6_nopay_octets Register)
rxudp_gd_octets on page 17-301	0x270	32	RO	0x0	Register 156 (rxudp_gd_octets Register)
rxudp_err_octets on page 17-302	0x274	32	RO	0x0	Register 157 (rxudp_err_octets Register)
rxtcp_gd_octets on page 17-303	0x278	32	RO	0x0	Register 158 (rxtcp_gd_octets Register)
rxtcperroctets on page 17-303	0x27C	32	RO	0x0	Register 159 (rxtcp_err_octets Register)
rxicmp_gd_octets on page 17-304	0x280	32	RO	0x0	Register 160 (rxicmp_gd_octets Register)

Register	Offset	Width	Acces s	Reset Value	Description
rxicmp_err_octets on page 17-304	0x284	32	RO	0x0	Register 161 (rxicmp_err_octets Register)
L3_L4_Control0 on page 17-305	0x400	32	RW	0x0	Register 256 (Layer 3 and Layer 4 Control Register 0)
Layer4_Address0 on page 17-307	0x404	32	RW	0x0	Register 257 (Layer 4 Address Register 0)
Layer3_Addr0_Reg0 on page 17-308	0x410	32	RW	0x0	Register 260 (Layer 3 Address 0 Register 0)
Layer3_Addr1_Reg0 on page 17-309	0x414	32	RW	0x0	Register 261 (Layer 3 Address 1 Register 0)
Layer3_Addr2_Reg0 on page 17-310	0x418	32	RW	0x0	Register 262 (Layer 3 Address 2 Register 0)
Layer3_Addr3_Reg0 on page 17-311	0x41C	32	RW	0x0	Register 263 (Layer 3 Address 3 Register 0)
L3_L4_Control1 on page 17-312	0x430	32	RW	0x0	Register 268 (Layer 3 and Layer 4 Control Register 1)
Layer4_Address1 on page 17-314	0x434	32	RW	0x0	Register 269 (Layer 4 Address Register 1)
Layer3_Addr0_Reg1 on page 17-315	0x440	32	RW	0x0	Register 272 (Layer 3 Address 0 Register 1)
Layer3_Addr1_Reg1 on page 17-316	0x444	32	RW	0x0	Register 273 (Layer 3 Address 1 Register 1)
Layer3_Addr2_Reg1 on page 17-317	0x448	32	RW	0x0	Register 274 (Layer 3 Address 2 Register 1)
Layer3_Addr3_Reg1 on page 17-318	0x44C	32	RW	0x0	Register 275 (Layer 3 Address 3 Register 1)
L3_L4_Control2 on page 17-318	0x460	32	RW	0x0	Register 280 (Layer 3 and Layer 4 Control Register 2)
Layer4_Address2 on page 17-321	0x464	32	RW	0x0	Register 281 (Layer 4 Address Register 2)
Layer3_Addr0_Reg2 on page 17-322	0x470	32	RW	0x0	Register 284 (Layer 3 Address 0 Register 2)
Layer3_Addr1_Reg2 on page 17-323	0x474	32	RW	0x0	Register 285 (Layer 3 Address 1 Register 2)
Layer3_Addr2_Reg2 on page 17-324	0x478	32	RW	0x0	Register 286 (Layer 3 Address 2 Register 2)
Layer3_Addr3_Reg2 on page 17-325	0x47C	32	RW	0x0	Register 287 (Layer 3 Address 3 Register 2)

Register	Offset	Width	Acces s	Reset Value	Description
L3_L4_Control3 on page 17-326	0x490	32	RW	0x0	Register 292 (Layer 3 and Layer 4 Control Register 3)
Layer4_Address3 on page 17-328	0x494	32	RW	0x0	Register 293 (Layer 4 Address Register 3)
Layer3_Addr0_Reg3 on page 17-329	0x4A0	32	RW	0x0	Register 296 (Layer 3 Address 0 Register 3)
Layer3_Addr1_Reg3 on page 17-330	0x4A4	32	RW	0x0	Register 297 (Layer 3 Address 1 Register 3)
Layer3_Addr2_Reg3 on page 17-331	0x4A8	32	RW	0x0	Register 298 (Layer 3 Address 2 Register 3)
Layer3_Addr3_Reg3 on page 17-332	0x4AC	32	RW	0x0	Register 299 (Layer 3 Address 3 Register 3)
Hash_Table_Reg0 on page 17-333	0x500	32	RW	0x0	Register 320 (Hash Table Register 0)
Hash_Table_Reg1 on page 17-333	0x504	32	RW	0x0	Register 321 (Hash Table Register 1)
Hash_Table_Reg2 on page 17-334	0x508	32	RW	0x0	Register 322 (Hash Table Register 2)
Hash_Table_Reg3 on page 17-334	0x50C	32	RW	0x0	Register 323 (Hash Table Register 3)
Hash_Table_Reg4 on page 17-335	0x510	32	RW	0x0	Register 324 (Hash Table Register 4)
Hash_Table_Reg5 on page 17-335	0x514	32	RW	0x0	Register 325 (Hash Table Register 5)
Hash_Table_Reg6 on page 17-336	0x518	32	RW	0x0	Register 326 (Hash Table Register 6)
Hash_Table_Reg7 on page 17-336	0x51C	32	RW	0x0	Register 327 (Hash Table Register 7)
VLAN_Incl_Reg on page 17-337	0x584	32	RW	0x0	Register 353 (VLAN Tag Inclusion or Replacement Register)
VLAN_Hash_Table_Reg on page 17-338	0x588	32	RW	0x0	Register 354 (VLAN Hash Table Register)
Timestamp_Control on page 17-339	0x700	32	RW	0x2000	Register 448 (Timestamp Control Register)
Sub_Second_Increment on page 17-344	0x704	32	RW	0x0	Register 449 (Sub-Second Increment Register)
System_Time_Seconds on page 17-345	0x708	32	RO	0x0	Register 450 (System Time - Seconds Register)

Register	Offset	Width	Acces s	Reset Value	Description
System_Time_Nanoseconds on page 17-346	0x70C	32	RO	0x0	Register 451 (System Time - Nanoseconds Register)
System_Time_Seconds_Update on page 17-346	0x710	32	RW	0x0	Register 452 (System Time - Seconds Update Register)
System_Time_Nanoseconds_Update on page 17-347	0x714	32	RW	0x0	Register 453 (System Time - Nanoseconds Update Register)
Timestamp_Addend on page 17-348	0x718	32	RW	0x0	Register 454 (Timestamp Addend Register)
Target_Time_Seconds on page 17-349	0x71C	32	RW	0x0	Register 455 (Target Time Seconds Register)
Target_Time_Nanoseconds on page 17-349	0x720	32	RW	0x0	Register 456 (Target Time Nanoseconds Register)
System_Time_Higher_Word_Seconds on page 17-350	0x724	32	RW	0x0	Register 457 (System Time - Higher Word Seconds Register)
Timestamp_Status on page 17-351	0x728	32	RO	0x0	Register 458 (Timestamp Status Register)
PPS_Control on page 17-353	0x72C	32	RW	0x0	Register 459 (PPS Control Register)
Auxiliary_Timestamp_Nanoseconds on page 17-356	0x730	32	RO	0x0	Register 460 (Auxiliary Timestamp - Nanoseconds Register)
Auxiliary_Timestamp_Seconds on page 17-356	0x734	32	RO	0x0	Register 461 (Auxiliary Timestamp - Seconds Register)
PPS0_Interval on page 17-357	0x760	32	RW	0x0	Register 472 (PPS0 Interval Register)
PPS0_Width on page 17-358	0x764	32	RW	0x0	Register 473 (PPS0 Width Register)
MAC_Address16_High on page 17-358	0x800	32	RW	0xFFFF	Register 512 (MAC Address16 High Register)
MAC_Address16_Low on page 17-362	0x804	32	RW	0xFFFFFFFF	Register 513 (MAC Address16 Low Register)
MAC_Address17_High on page 17-362	0x808	32	RW	0xFFFF	Register 514 (MAC Address17 High Register)
MAC_Address17_Low on page 17-365	0x80C	32	RW	0xFFFFFFFF	Register 515 (MAC Address17 Low Register)
MAC_Address18_High on page 17-366	0x810	32	RW	0xFFFF	Register 516 (MAC Address18 High Register)
MAC_Address18_Low on page 17-369	0x814	32	RW	0xFFFFFFFF	Register 517 (MAC Address18 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address19_High on page 17-370	0x818	32	RW	0xFFFF	Register 518 (MAC Address19 High Register)
MAC_Address19_Low on page 17-373	0x81C	32	RW	0xFFFFFFFF	Register 519 (MAC Address19 Low Register)
MAC_Address20_High on page 17-374	0x820	32	RW	0xFFFF	Register 520 (MAC Address20 High Register)
MAC_Address20_Low on page 17-377	0x824	32	RW	0xFFFFFFFF	Register 521 (MAC Address20 Low Register)
MAC_Address21_High on page 17-378	0x828	32	RW	0xFFFF	Register 522 (MAC Address21 High Register)
MAC_Address21_Low on page 17-381	0x82C	32	RW	0xFFFFFFFF	Register 523 (MAC Address21 Low Register)
MAC_Address22_High on page 17-382	0x830	32	RW	0xFFFF	Register 524 (MAC Address22 High Register)
MAC_Address22_Low on page 17-385	0x834	32	RW	0xFFFFFFFF	Register 525 (MAC Address22 Low Register)
MAC_Address23_High on page 17-386	0x838	32	RW	0xFFFF	Register 526 (MAC Address23 High Register)
MAC_Address23_Low on page 17-389	0x83C	32	RW	0xFFFFFFFF	Register 527 (MAC Address23 Low Register)
MAC_Address24_High on page 17-390	0x840	32	RW	0xFFFF	Register 528 (MAC Address24 High Register)
MAC_Address24_Low on page 17-393	0x844	32	RW	0xFFFFFFFF	Register 529 (MAC Address24 Low Register)
MAC_Address25_High on page 17-394	0x848	32	RW	0xFFFF	Register 530 (MAC Address25 High Register)
MAC_Address25_Low on page 17-397	0x84C	32	RW	0xFFFFFFFF	Register 531 (MAC Address25 Low Register)
MAC_Address26_High on page 17-398	0x850	32	RW	0xFFFF	Register 532 (MAC Address26 High Register)
MAC_Address26_Low on page 17-401	0x854	32	RW	0xFFFFFFFF	Register 533 (MAC Address26 Low Register)
MAC_Address27_High on page 17-402	0x858	32	RW	0xFFFF	Register 534 (MAC Address27 High Register)
MAC_Address27_Low on page 17-405	0x85C	32	RW	0xFFFFFFFF	Register 535 (MAC Address27 Low Register)
MAC_Address28_High on page 17-406	0x860	32	RW	0xFFFF	Register 536 (MAC Address28 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address28_Low on page 17-409	0x864	32	RW	0xFFFFFFFF	Register 537 (MAC Address28 Low Register)
MAC_Address29_High on page 17-410	0x868	32	RW	0xFFFF	Register 538 (MAC Address29 High Register)
MAC_Address29_Low on page 17-413	0x86C	32	RW	0xFFFFFFFF	Register 539 (MAC Address29 Low Register)
MAC_Address30_High on page 17-414	0x870	32	RW	0xFFFF	Register 540 (MAC Address30 High Register)
MAC_Address30_Low on page 17-417	0x874	32	RW	0xFFFFFFFF	Register 541 (MAC Address30 Low Register)
MAC_Address31_High on page 17-418	0x878	32	RW	0xFFFF	Register 542 (MAC Address31 High Register)
MAC_Address31_Low on page 17-421	0x87C	32	RW	0xFFFFFFFF	Register 543 (MAC Address31 Low Register)
MAC_Address32_High on page 17-422	0x880	32	RW	0xFFFF	Register 544 (MAC Address32 High Register)
MAC_Address32_Low on page 17-425	0x884	32	RW	0xFFFFFFFF	Register 545 (MAC Address32 Low Register)
MAC_Address33_High on page 17-426	0x888	32	RW	0xFFFF	Register 546 (MAC Address33 High Register)
MAC_Address33_Low on page 17-429	0x88C	32	RW	0xFFFFFFFF	Register 547 (MAC Address33 Low Register)
MAC_Address34_High on page 17-430	0x890	32	RW	0xFFFF	Register 548 (MAC Address34 High Register)
MAC_Address34_Low on page 17-433	0x894	32	RW	0xFFFFFFFF	Register 549 (MAC Address34 Low Register)
MAC_Address35_High on page 17-434	0x898	32	RW	0xFFFF	Register 550 (MAC Address35 High Register)
MAC_Address35_Low on page 17-437	0x89C	32	RW	0xFFFFFFFF	Register 551 (MAC Address35 Low Register)
MAC_Address36_High on page 17-438	0x8A0	32	RW	0xFFFF	Register 552 (MAC Address36 High Register)
MAC_Address36_Low on page 17-441	0x8A4	32	RW	0xFFFFFFFF	Register 553 (MAC Address36 Low Register)
MAC_Address37_High on page 17-442	0x8A8	32	RW	0xFFFF	Register 554 (MAC Address37 High Register)
MAC_Address37_Low on page 17-445	0x8AC	32	RW	0xFFFFFFFF	Register 555 (MAC Address37 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address38_High on page 17-446	0x8B0	32	RW	0xFFFF	Register 556 (MAC Address38 High Register)
MAC_Address38_Low on page 17-449	0x8B4	32	RW	0xFFFFFFFF	Register 557 (MAC Address38 Low Register)
MAC_Address39_High on page 17-450	0x8B8	32	RW	0xFFFF	Register 558 (MAC Address39 High Register)
MAC_Address39_Low on page 17-453	0x8BC	32	RW	0xFFFFFFFF	Register 559 (MAC Address39 Low Register)
MAC_Address40_High on page 17-454	0x8C0	32	RW	0xFFFF	Register 560 (MAC Address40 High Register)
MAC_Address40_Low on page 17-457	0x8C4	32	RW	0xFFFFFFFF	Register 561 (MAC Address40 Low Register)
MAC_Address41_High on page 17-458	0x8C8	32	RW	0xFFFF	Register 562 (MAC Address41 High Register)
MAC_Address41_Low on page 17-461	0x8CC	32	RW	0xFFFFFFFF	Register 563 (MAC Address41 Low Register)
MAC_Address42_High on page 17-462	0x8D0	32	RW	0xFFFF	Register 564 (MAC Address42 High Register)
MAC_Address42_Low on page 17-465	0x8D4	32	RW	0xFFFFFFFF	Register 565 (MAC Address42 Low Register)
MAC_Address43_High on page 17-466	0x8D8	32	RW	0xFFFF	Register 566 (MAC Address43 High Register)
MAC_Address43_Low on page 17-469	0x8DC	32	RW	0xFFFFFFFF	Register 567 (MAC Address43 Low Register)
MAC_Address44_High on page 17-470	0x8E0	32	RW	0xFFFF	Register 568 (MAC Address44 High Register)
MAC_Address44_Low on page 17-473	0x8E4	32	RW	0xFFFFFFFF	Register 569 (MAC Address44 Low Register)
MAC_Address45_High on page 17-474	0x8E8	32	RW	0xFFFF	Register 570 (MAC Address45 High Register)
MAC_Address45_Low on page 17-477	0x8EC	32	RW	0xFFFFFFFF	Register 571 (MAC Address45 Low Register)
MAC_Address46_High on page 17-478	0x8F0	32	RW	0xFFFF	Register 572 (MAC Address46 High Register)
MAC_Address46_Low on page 17-481	0x8F4	32	RW	0xFFFFFFFF	Register 573 (MAC Address46 Low Register)
MAC_Address47_High on page 17-482	0x8F8	32	RW	0xFFFF	Register 574 (MAC Address47 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address47_Low on page 17-485	0x8FC	32	RW	0xFFFFFFFF	Register 575 (MAC Address47 Low Register)
MAC_Address48_High on page 17-486	0x900	32	RW	0xFFFF	Register 576 (MAC Address48 High Register)
MAC_Address48_Low on page 17-489	0x904	32	RW	0xFFFFFFFF	Register 577 (MAC Address48 Low Register)
MAC_Address49_High on page 17-490	0x908	32	RW	0xFFFF	Register 578 (MAC Address49 High Register)
MAC_Address49_Low on page 17-493	0x90C	32	RW	0xFFFFFFFF	Register 579 (MAC Address49 Low Register)
MAC_Address50_High on page 17-494	0x910	32	RW	0xFFFF	Register 580 (MAC Address50 High Register)
MAC_Address50_Low on page 17-497	0x914	32	RW	0xFFFFFFFF	Register 581 (MAC Address50 Low Register)
MAC_Address51_High on page 17-498	0x918	32	RW	0xFFFF	Register 582 (MAC Address51 High Register)
MAC_Address51_Low on page 17-501	0x91C	32	RW	0xFFFFFFFF	Register 583 (MAC Address51 Low Register)
MAC_Address52_High on page 17-502	0x920	32	RW	0xFFFF	Register 584 (MAC Address52 High Register)
MAC_Address52_Low on page 17-505	0x924	32	RW	0xFFFFFFFF	Register 585 (MAC Address52 Low Register)
MAC_Address53_High on page 17-506	0x928	32	RW	0xFFFF	Register 586 (MAC Address53 High Register)
MAC_Address53_Low on page 17-509	0x92C	32	RW	0xFFFFFFFF	Register 587 (MAC Address53 Low Register)
MAC_Address54_High on page 17-510	0x930	32	RW	0xFFFF	Register 588 (MAC Address54 High Register)
MAC_Address54_Low on page 17-513	0x934	32	RW	0xFFFFFFFF	Register 589 (MAC Address54 Low Register)
MAC_Address55_High on page 17-514	0x938	32	RW	0xFFFF	Register 590 (MAC Address55 High Register)
MAC_Address55_Low on page 17-517	0x93C	32	RW	0xFFFFFFFF	Register 591 (MAC Address55 Low Register)
MAC_Address56_High on page 17-518	0x940	32	RW	0xFFFF	Register 592 (MAC Address56 High Register)
MAC_Address56_Low on page 17-521	0x944	32	RW	0xFFFFFFFF	Register 593 (MAC Address56 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address57_High on page 17-522	0x948	32	RW	0xFFFF	Register 594 (MAC Address57 High Register)
MAC_Address57_Low on page 17-525	0x94C	32	RW	0xFFFFFFFF	Register 595 (MAC Address57 Low Register)
MAC_Address58_High on page 17-526	0x950	32	RW	0xFFFF	Register 596 (MAC Address58 High Register)
MAC_Address58_Low on page 17-529	0x954	32	RW	0xFFFFFFFF	Register 597 (MAC Address58 Low Register)
MAC_Address59_High on page 17-530	0x958	32	RW	0xFFFF	Register 598 (MAC Address59 High Register)
MAC_Address59_Low on page 17-533	0x95C	32	RW	0xFFFFFFFF	Register 599 (MAC Address59 Low Register)
MAC_Address60_High on page 17-534	0x960	32	RW	0xFFFF	Register 600 (MAC Address60 High Register)
MAC_Address60_Low on page 17-537	0x964	32	RW	0xFFFFFFFF	Register 601 (MAC Address60 Low Register)
MAC_Address61_High on page 17-538	0x968	32	RW	0xFFFF	Register 602 (MAC Address61 High Register)
MAC_Address61_Low on page 17-541	0x96C	32	RW	0xFFFFFFFF	Register 603 (MAC Address61 Low Register)
MAC_Address62_High on page 17-542	0x970	32	RW	0xFFFF	Register 604 (MAC Address62 High Register)
MAC_Address62_Low on page 17-545	0x974	32	RW	0xFFFFFFFF	Register 605 (MAC Address62 Low Register)
MAC_Address63_High on page 17-546	0x978	32	RW	0xFFFF	Register 606 (MAC Address63 High Register)
MAC_Address63_Low on page 17-549	0x97C	32	RW	0xFFFFFFFF	Register 607 (MAC Address63 Low Register)
MAC_Address64_High on page 17-550	0x980	32	RW	0xFFFF	Register 608 (MAC Address64 High Register)
MAC_Address64_Low on page 17-553	0x984	32	RW	0xFFFFFFFF	Register 609 (MAC Address64 Low Register)
MAC_Address65_High on page 17-554	0x988	32	RW	0xFFFF	Register 610 (MAC Address65 High Register)
MAC_Address65_Low on page 17-557	0x98C	32	RW	0xFFFFFFFF	Register 611 (MAC Address65 Low Register)
MAC_Address66_High on page 17-558	0x990	32	RW	0xFFFF	Register 612 (MAC Address66 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address66_Low on page 17-561	0x994	32	RW	0xFFFFFFFF	Register 613 (MAC Address66 Low Register)
MAC_Address67_High on page 17-562	0x998	32	RW	0xFFFF	Register 614 (MAC Address67 High Register)
MAC_Address67_Low on page 17-565	0x99C	32	RW	0xFFFFFFFF	Register 615 (MAC Address67 Low Register)
MAC_Address68_High on page 17-566	0x9A0	32	RW	0xFFFF	Register 616 (MAC Address68 High Register)
MAC_Address68_Low on page 17-569	0x9A4	32	RW	0xFFFFFFFF	Register 617 (MAC Address68 Low Register)
MAC_Address69_High on page 17-570	0x9A8	32	RW	0xFFFF	Register 618 (MAC Address69 High Register)
MAC_Address69_Low on page 17-573	0x9AC	32	RW	0xFFFFFFFF	Register 619 (MAC Address69 Low Register)
MAC_Address70_High on page 17-574	0x9B0	32	RW	0xFFFF	Register 620 (MAC Address70 High Register)
MAC_Address70_Low on page 17-577	0x9B4	32	RW	0xFFFFFFFF	Register 621 (MAC Address70 Low Register)
MAC_Address71_High on page 17-578	0x9B8	32	RW	0xFFFF	Register 622 (MAC Address71 High Register)
MAC_Address71_Low on page 17-581	0x9BC	32	RW	0xFFFFFFFF	Register 623 (MAC Address71 Low Register)
MAC_Address72_High on page 17-582	0x9C0	32	RW	0xFFFF	Register 624 (MAC Address72 High Register)
MAC_Address72_Low on page 17-585	0x9C4	32	RW	0xFFFFFFFF	Register 625 (MAC Address72 Low Register)
MAC_Address73_High on page 17-586	0x9C8	32	RW	0xFFFF	Register 626 (MAC Address73 High Register)
MAC_Address73_Low on page 17-589	0x9CC	32	RW	0xFFFFFFFF	Register 627 (MAC Address73 Low Register)
MAC_Address74_High on page 17-590	0x9D0	32	RW	0xFFFF	Register 628 (MAC Address74 High Register)
MAC_Address74_Low on page 17-593	0x9D4	32	RW	0xFFFFFFFF	Register 629 (MAC Address74 Low Register)
MAC_Address75_High on page 17-594	0x9D8	32	RW	0xFFFF	Register 630 (MAC Address75 High Register)
MAC_Address75_Low on page 17-597	0x9DC	32	RW	0xFFFFFFFF	Register 631 (MAC Address75 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address76_High on page 17-598	0x9E0	32	RW	0xFFFF	Register 632 (MAC Address76 High Register)
MAC_Address76_Low on page 17-601	0x9E4	32	RW	0xFFFFFFFF	Register 633 (MAC Address76 Low Register)
MAC_Address77_High on page 17-602	0x9E8	32	RW	0xFFFF	Register 634 (MAC Address77 High Register)
MAC_Address77_Low on page 17-605	0x9EC	32	RW	0xFFFFFFFF	Register 635 (MAC Address77 Low Register)
MAC_Address78_High on page 17-606	0x9F0	32	RW	0xFFFF	Register 636 (MAC Address78 High Register)
MAC_Address78_Low on page 17-609	0x9F4	32	RW	0xFFFFFFFF	Register 637 (MAC Address78 Low Register)
MAC_Address79_High on page 17-610	0x9F8	32	RW	0xFFFF	Register 638 (MAC Address79 High Register)
MAC_Address79_Low on page 17-613	0x9FC	32	RW	0xFFFFFFFF	Register 639 (MAC Address79 Low Register)
MAC_Address80_High on page 17-614	0xA00	32	RW	0xFFFF	Register 640 (MAC Address80 High Register)
MAC_Address80_Low on page 17-617	0xA04	32	RW	0xFFFFFFFF	Register 641 (MAC Address80 Low Register)
MAC_Address81_High on page 17-618	0xA08	32	RW	0xFFFF	Register 642 (MAC Address81 High Register)
MAC_Address81_Low on page 17-621	0xA0C	32	RW	0xFFFFFFFF	Register 643 (MAC Address81 Low Register)
MAC_Address82_High on page 17-622	0xA10	32	RW	0xFFFF	Register 644 (MAC Address82 High Register)
MAC_Address82_Low on page 17-625	0xA14	32	RW	0xFFFFFFFF	Register 645 (MAC Address82 Low Register)
MAC_Address83_High on page 17-626	0xA18	32	RW	0xFFFF	Register 646 (MAC Address83 High Register)
MAC_Address83_Low on page 17-629	0xA1C	32	RW	0xFFFFFFFF	Register 647 (MAC Address83 Low Register)
MAC_Address84_High on page 17-630	0xA20	32	RW	0xFFFF	Register 648 (MAC Address84 High Register)
MAC_Address84_Low on page 17-633	0xA24	32	RW	0xFFFFFFFF	Register 649 (MAC Address84 Low Register)
MAC_Address85_High on page 17-634	0xA28	32	RW	0xFFFF	Register 650 (MAC Address85 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address85_Low on page 17-637	0xA2C	32	RW	0xFFFFFFFF	Register 651 (MAC Address85 Low Register)
MAC_Address86_High on page 17-638	0xA30	32	RW	0xFFFF	Register 652 (MAC Address86 High Register)
MAC_Address86_Low on page 17-641	0xA34	32	RW	0xFFFFFFFF	Register 653 (MAC Address86 Low Register)
MAC_Address87_High on page 17-642	0xA38	32	RW	0xFFFF	Register 654 (MAC Address87 High Register)
MAC_Address87_Low on page 17-645	0xA3C	32	RW	0xFFFFFFFF	Register 655 (MAC Address87 Low Register)
MAC_Address88_High on page 17-646	0xA40	32	RW	0xFFFF	Register 656 (MAC Address88 High Register)
MAC_Address88_Low on page 17-649	0xA44	32	RW	0xFFFFFFFF	Register 657 (MAC Address88 Low Register)
MAC_Address89_High on page 17-650	0xA48	32	RW	0xFFFF	Register 658 (MAC Address89 High Register)
MAC_Address89_Low on page 17-653	0xA4C	32	RW	0xFFFFFFFF	Register 659 (MAC Address89 Low Register)
MAC_Address90_High on page 17-654	0xA50	32	RW	0xFFFF	Register 660 (MAC Address90 High Register)
MAC_Address90_Low on page 17-657	0xA54	32	RW	0xFFFFFFFF	Register 661 (MAC Address90 Low Register)
MAC_Address91_High on page 17-658	0xA58	32	RW	0xFFFF	Register 662 (MAC Address91 High Register)
MAC_Address91_Low on page 17-661	0xA5C	32	RW	0xFFFFFFFF	Register 663 (MAC Address91 Low Register)
MAC_Address92_High on page 17-662	0xA60	32	RW	0xFFFF	Register 664 (MAC Address92 High Register)
MAC_Address92_Low on page 17-665	0xA64	32	RW	0xFFFFFFFF	Register 665 (MAC Address92 Low Register)
MAC_Address93_High on page 17-666	0xA68	32	RW	0xFFFF	Register 666 (MAC Address93 High Register)
MAC_Address93_Low on page 17-669	0xA6C	32	RW	0xFFFFFFFF	Register 667 (MAC Address93 Low Register)
MAC_Address94_High on page 17-670	0xA70	32	RW	0xFFFF	Register 668 (MAC Address94 High Register)
MAC_Address94_Low on page 17-673	0xA74	32	RW	0xFFFFFFFF	Register 669 (MAC Address94 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address95_High on page 17-674	0xA78	32	RW	0xFFFF	Register 670 (MAC Address95 High Register)
MAC_Address95_Low on page 17-677	0xA7C	32	RW	0xFFFFFFFF	Register 671 (MAC Address95 Low Register)
MAC_Address96_High on page 17-678	0xA80	32	RW	0xFFFF	Register 672 (MAC Address96 High Register)
MAC_Address96_Low on page 17-681	0xA84	32	RW	0xFFFFFFFF	Register 673 (MAC Address96 Low Register)
MAC_Address97_High on page 17-682	0xA88	32	RW	0xFFFF	Register 674 (MAC Address97 High Register)
MAC_Address97_Low on page 17-685	0xA8C	32	RW	0xFFFFFFFF	Register 675 (MAC Address97 Low Register)
MAC_Address98_High on page 17-686	0xA90	32	RW	0xFFFF	Register 676 (MAC Address98 High Register)
MAC_Address98_Low on page 17-689	0xA94	32	RW	0xFFFFFFFF	Register 677 (MAC Address98 Low Register)
MAC_Address99_High on page 17-690	0xA98	32	RW	0xFFFF	Register 678 (MAC Address99 High Register)
MAC_Address99_Low on page 17-693	0xA9C	32	RW	0xFFFFFFFF	Register 679 (MAC Address99 Low Register)
MAC_Address100_High on page 17-694	0xAA0	32	RW	0xFFFF	Register 680 (MAC Address100 High Register)
MAC_Address100_Low on page 17-697	0xAA4	32	RW	0xFFFFFFFF	Register 681 (MAC Address100 Low Register)
MAC_Address101_High on page 17-698	0xAA8	32	RW	0xFFFF	Register 682 (MAC Address101 High Register)
MAC_Address101_Low on page 17-701	0xAAC	32	RW	0xFFFFFFFF	Register 683 (MAC Address101 Low Register)
MAC_Address102_High on page 17-702	0xAB0	32	RW	0xFFFF	Register 684 (MAC Address102 High Register)
MAC_Address102_Low on page 17-705	0xAB4	32	RW	0xFFFFFFFF	Register 685 (MAC Address102 Low Register)
MAC_Address103_High on page 17-706	0xAB8	32	RW	0xFFFF	Register 686 (MAC Address103 High Register)
MAC_Address103_Low on page 17-709	0xABC	32	RW	0xFFFFFFFF	Register 687 (MAC Address103 Low Register)
MAC_Address104_High on page 17-710	0xAC0	32	RW	0xFFFF	Register 688 (MAC Address104 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address104_Low on page 17-713	0xAC4	32	RW	0xFFFFFFFF	Register 689 (MAC Address104 Low Register)
MAC_Address105_High on page 17-714	0xAC8	32	RW	0xFFFF	Register 690 (MAC Address105 High Register)
MAC_Address105_Low on page 17-717	0xACC	32	RW	0xFFFFFFFF	Register 691 (MAC Address105 Low Register)
MAC_Address106_High on page 17-718	0xAD0	32	RW	0xFFFF	Register 692 (MAC Address106 High Register)
MAC_Address106_Low on page 17-721	0xAD4	32	RW	0xFFFFFFFF	Register 693 (MAC Address106 Low Register)
MAC_Address107_High on page 17-722	0xAD8	32	RW	0xFFFF	Register 694 (MAC Address107 High Register)
MAC_Address107_Low on page 17-725	0xADC	32	RW	0xFFFFFFFF	Register 695 (MAC Address107 Low Register)
MAC_Address108_High on page 17-726	0xAE0	32	RW	0xFFFF	Register 696 (MAC Address108 High Register)
MAC_Address108_Low on page 17-729	0xAE4	32	RW	0xFFFFFFFF	Register 697 (MAC Address108 Low Register)
MAC_Address109_High on page 17-730	0xAE8	32	RW	0xFFFF	Register 698 (MAC Address109 High Register)
MAC_Address109_Low on page 17-733	0xAEC	32	RW	0xFFFFFFFF	Register 699 (MAC Address109 Low Register)
MAC_Address110_High on page 17-734	0xAF0	32	RW	0xFFFF	Register 700 (MAC Address110 High Register)
MAC_Address110_Low on page 17-737	0xAF4	32	RW	0xFFFFFFFF	Register 701 (MAC Address110 Low Register)
MAC_Address111_High on page 17-738	0xAF8	32	RW	0xFFFF	Register 702 (MAC Address111 High Register)
MAC_Address111_Low on page 17-741	0xAFC	32	RW	0xFFFFFFFF	Register 703 (MAC Address111 Low Register)
MAC_Address112_High on page 17-742	0xB00	32	RW	0xFFFF	Register 704 (MAC Address112 High Register)
MAC_Address112_Low on page 17-745	0xB04	32	RW	0xFFFFFFFF	Register 705 (MAC Address112 Low Register)
MAC_Address113_High on page 17-746	0xB08	32	RW	0xFFFF	Register 706 (MAC Address113 High Register)
MAC_Address113_Low on page 17-749	0xB0C	32	RW	0xFFFFFFFF	Register 707 (MAC Address113 Low Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address114_High on page 17-750	0xB10	32	RW	0xFFFF	Register 708 (MAC Address114 High Register)
MAC_Address114_Low on page 17-753	0xB14	32	RW	0xFFFFFFFF	Register 709 (MAC Address114 Low Register)
MAC_Address115_High on page 17-754	0xB18	32	RW	0xFFFF	Register 710 (MAC Address115 High Register)
MAC_Address115_Low on page 17-757	0xB1C	32	RW	0xFFFFFFFF	Register 711 (MAC Address115 Low Register)
MAC_Address116_High on page 17-758	0xB20	32	RW	0xFFFF	Register 712 (MAC Address116 High Register)
MAC_Address116_Low on page 17-761	0xB24	32	RW	0xFFFFFFFF	Register 713 (MAC Address116 Low Register)
MAC_Address117_High on page 17-762	0xB28	32	RW	0xFFFF	Register 714 (MAC Address117 High Register)
MAC_Address117_Low on page 17-765	0xB2C	32	RW	0xFFFFFFFF	Register 715 (MAC Address117 Low Register)
MAC_Address118_High on page 17-766	0xB30	32	RW	0xFFFF	Register 716 (MAC Address118 High Register)
MAC_Address118_Low on page 17-769	0xB34	32	RW	0xFFFFFFFF	Register 717 (MAC Address118 Low Register)
MAC_Address119_High on page 17-770	0xB38	32	RW	0xFFFF	Register 718 (MAC Address119 High Register)
MAC_Address119_Low on page 17-773	0xB3C	32	RW	0xFFFFFFFF	Register 719 (MAC Address119 Low Register)
MAC_Address120_High on page 17-774	0xB40	32	RW	0xFFFF	Register 720 (MAC Address120 High Register)
MAC_Address120_Low on page 17-777	0xB44	32	RW	0xFFFFFFFF	Register 721 (MAC Address120 Low Register)
MAC_Address121_High on page 17-778	0xB48	32	RW	0xFFFF	Register 722 (MAC Address121 High Register)
MAC_Address121_Low on page 17-781	0xB4C	32	RW	0xFFFFFFFF	Register 723 (MAC Address121 Low Register)
MAC_Address122_High on page 17-782	0xB50	32	RW	0xFFFF	Register 724 (MAC Address122 High Register)
MAC_Address122_Low on page 17-785	0xB54	32	RW	0xFFFFFFFF	Register 725 (MAC Address122 Low Register)
MAC_Address123_High on page 17-786	0xB58	32	RW	0xFFFF	Register 726 (MAC Address123 High Register)

Register	Offset	Width	Acces s	Reset Value	Description
MAC_Address123_Low on page 17-789	0xB5C	32	RW	0xFFFFFFFF	Register 727 (MAC Address123 Low Register)
MAC_Address124_High on page 17-790	0xB60	32	RW	0xFFFF	Register 728 (MAC Address124 High Register)
MAC_Address124_Low on page 17-793	0xB64	32	RW	0xFFFFFFFF	Register 729 (MAC Address124 Low Register)
MAC_Address125_High on page 17-794	0xB68	32	RW	0xFFFF	Register 730 (MAC Address125 High Register)
MAC_Address125_Low on page 17-797	0xB6C	32	RW	0xFFFFFFFF	Register 731 (MAC Address125 Low Register)
MAC_Address126_High on page 17-798	0xB70	32	RW	0xFFFF	Register 732 (MAC Address126 High Register)
MAC_Address126_Low on page 17-801	0xB74	32	RW	0xFFFFFFFF	Register 733 (MAC Address126 Low Register)
MAC_Address127_High on page 17-802	0xB78	32	RW	0xFFFF	Register 734 (MAC Address127 High Register)
MAC_Address127_Low on page 17-805	0xB7C	32	RW	0xFFFFFFFF	Register 735 (MAC Address127 Low Register)

DMA Register Group

Register	Offset	Width	Acces s	Reset Value	Description
Bus_Mode on page 17-808	0x1000	32	RW	0x20101	Register 0 (Bus Mode Register)
Transmit_Poll_Demand on page 17-812	0x1004	32	RW	0x0	Register 1 (Transmit Poll Demand Register)
Receive_Poll_Demand on page 17-813	0x1008	32	RW	0x0	Register 2 (Receive Poll Demand Register)
Receive_Descriptor_List_Address on page 17-813	0x100C	32	RW	0x0	Register 3 (Receive Descriptor List Address Register)
Transmit_Descriptor_List_Address on page 17-814	0x1010	32	RW	0x0	Register 4 (Transmit Descriptor List Address Register)
Status on page 17-815	0x1014	32	RW	0x0	Register 5 (Status Register)
Operation_Mode on page 17-820	0x1018	32	RW	0x0	Register 6 (Operation Mode Register)
Interrupt_Enable on page 17-826	0x101C	32	RW	0x0	Register 7 (Interrupt Enable Register)
Missed_Frame_And_Buffer_Overflow_Counter on page 17-830	0x1020	32	RO	0x0	Register 8 (Missed Frame and Buffer Overflow Counter Register)

Register	Offset	Width	Access	Reset Value	Description
Receive_Interrupt_Watchdog_Timer on page 17-831	0x1024	32	RW	0x0	Register 9 (Receive Interrupt Watchdog Timer Register)
AXI_Bus_Mode on page 17-832	0x1028	32	RW	0x110001	Register 10 (AXI Bus Mode Register)
AHB_or_AXI_Status on page 17-835	0x102C	32	RO	0x0	Register 11 (AHB or AXI Status Register)
Current_Host_Transmit_Descriptor on page 17-836	0x1048	32	RO	0x0	Register 18 (Current Host Transmit Descriptor Register)
Current_Host_Receive_Descriptor on page 17-836	0x104C	32	RO	0x0	Register 19 (Current Host Receive Descriptor Register)
Current_Host_Transmit_Buffer_Address on page 17-837	0x1050	32	RO	0x0	Register 20 (Current Host Transmit Buffer Address Register)
Current_Host_Receive_Buffer_Address on page 17-837	0x1054	32	RO	0x0	Register 21 (Current Host Receive Buffer Address Register)
HW_Feature on page 17-838	0x1058	32	RO	0x10D7F37	Register 22 (HW Feature Register)

GMAC Register Group Register Descriptions

GMAC Register Group

Offset: 0x0

MAC_Configuration on page 17-126
The MAC Configuration register establishes receive and transmit operating modes.

MAC_Frame_Filter on page 17-134
The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

GMII_Address on page 17-140
The GMII Address register controls the management cycles to the external PHY through the management interface.

GMII_Data on page 17-142
The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4 (GMII Address Register). This register also stores the Read data from the PHY register located at the address specified by Register 4.

Flow_Control on page 17-143
The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC's Flow control block. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

VLAN_Tag on page 17-146
The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. Because the VLAN Tag register is double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.

Version on page 17-148
The Version registers identifies the version of the EMAC. This register contains two bytes: one specified by Synopsys to identify the core release number, and the other specified by Altera.

Debug on page 17-148
The Debug register gives the status of all main blocks of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

LPI_Control_Status on page 17-152
The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

LPI_Timers_Control on page 17-155
The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Interrupt_Status on page 17-156
The Interrupt Status register identifies the events in the MAC that can generate interrupt. All interrupt events are generated only when the corresponding optional feature is enabled.

Interrupt_Mask on page 17-159
The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o.

MAC_Address0_High on page 17-160
The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211. Because the MAC address registers are double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

MAC_Address0_Low on page 17-161
The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.

MAC_Address1_High on page 17-162
The MAC Address1 High register holds the upper 16 bits of the 2nd 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address1_Low on page 17-165
The MAC Address1 Low register holds the lower 32 bits of the 2nd 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address2_High on page 17-166
The MAC Address2 High register holds the upper 16 bits of the 3rd 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address2 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address2_Low on page 17-169
The MAC Address2 Low register holds the lower 32 bits of the 3rd 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address3_High on page 17-170
The MAC Address3 High register holds the upper 16 bits of the 4th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address3 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address3_Low on page 17-173
The MAC Address3 Low register holds the lower 32 bits of the 4th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address4_High on page 17-174
The MAC Address4 High register holds the upper 16 bits of the 5th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address4 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address4_Low on page 17-177
The MAC Address4 Low register holds the lower 32 bits of the 5th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address5_High on page 17-178
The MAC Address5 High register holds the upper 16 bits of the 6th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address5 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address5_Low on page 17-181
The MAC Address5 Low register holds the lower 32 bits of the 6th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address6_High on page 17-182
The MAC Address6 High register holds the upper 16 bits of the 7th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address6 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address6_Low on page 17-185
The MAC Address6 Low register holds the lower 32 bits of the 7th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address7_High on page 17-186
The MAC Address7 High register holds the upper 16 bits of the 8th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address7 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address7_Low on page 17-189
The MAC Address7 Low register holds the lower 32 bits of the 8th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address8_High on page 17-190
The MAC Address8 High register holds the upper 16 bits of the 9th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address8 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address8_Low on page 17-193
The MAC Address8 Low register holds the lower 32 bits of the 9th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address9_High on page 17-194
The MAC Address9 High register holds the upper 16 bits of the 10th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address9 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address9_Low on page 17-197
The MAC Address9 Low register holds the lower 32 bits of the 10th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address10_High on page 17-198
The MAC Address10 High register holds the upper 16 bits of the 11th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address10 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address10_Low on page 17-201
The MAC Address10 Low register holds the lower 32 bits of the 11th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address11_High on page 17-202
The MAC Address11 High register holds the upper 16 bits of the 12th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address11 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address11_Low on page 17-205
The MAC Address11 Low register holds the lower 32 bits of the 12th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address12_High on page 17-206
The MAC Address12 High register holds the upper 16 bits of the 13th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address12 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address12_Low on page 17-209
The MAC Address12 Low register holds the lower 32 bits of the 13th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address13_High on page 17-210
The MAC Address13 High register holds the upper 16 bits of the 14th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address13 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address13_Low on page 17-213
The MAC Address13 Low register holds the lower 32 bits of the 14th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address14_High on page 17-214
The MAC Address14 High register holds the upper 16 bits of the 15th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address14 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address14_Low on page 17-217
The MAC Address14 Low register holds the lower 32 bits of the 15th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address15_High on page 17-218
The MAC Address15 High register holds the upper 16 bits of the 16th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address15_Low on page 17-221
The MAC Address15 Low register holds the lower 32 bits of the 16th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

SGMII_RGMII_SMII_Control_Status on page 17-222
The SGMII/RGMII/SMII Status register indicates the status signals received by the RGMII interface (selected at reset) from the PHY.

MMC_Control on page 17-223
The MMC Control register establishes the operating mode of the management counters. Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.

MMC_Receive_Interrupt on page 17-225
The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens: * Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter). * Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

MMC_Transmit_Interrupt on page 17-231
The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

MMC_Receive_Interrupt_Mask on page 17-236
The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.

MMC_Transmit_Interrupt_Mask on page 17-242
The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

txoctetcount_gb on page 17-248
Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames

txframecount_gb on page 17-249
Number of good and bad frames transmitted, exclusive of retried frames

txbroadcastframes_g on page 17-249
Number of good broadcast frames transmitted

txmulticastframes_g on page 17-250
Number of good multicast frames transmitted

tx64octets_gb on page 17-250
Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames

tx65to127octets_gb on page 17-251
Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames

tx128to255octets_gb on page 17-251
Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames

tx256to511octets_gb on page 17-252
Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames

tx512to1023octets_gb on page 17-253
Number of good and bad frames transmitted with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames

tx1024tomaxoctets_gb on page 17-253
Number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames

txunicastframes_gb on page 17-254
Number of good and bad unicast frames transmitted

txmulticastframes_gb on page 17-254
Number of good and bad multicast frames transmitted

txbroadcastframes_gb on page 17-255
Number of good and bad broadcast frames transmitted

txunderflowerror on page 17-255
Number of frames aborted due to frame underflow error

txsinglecol_g on page 17-256
Number of successfully transmitted frames after a single collision in Half-duplex mode

txmulticol_g on page 17-256
Number of successfully transmitted frames after more than a single collision in Half-duplex mode

txdeferred on page 17-257
Number of successfully transmitted frames after a deferral in Halfduplex mode

txlatecol on page 17-257
Number of frames aborted due to late collision error

txexesscol on page 17-258
Number of frames aborted due to excessive (16) collision errors

txcarriererr on page 17-258
Number of frames aborted due to carrier sense error (no carrier or loss of carrier)

txoctetcnt on page 17-259
Number of bytes transmitted, exclusive of preamble, in good frames only

txframecount_g on page 17-260
Number of good frames transmitted

txexcessdef on page 17-260
Number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times)

txpauseframes on page 17-261
Number of good PAUSE frames transmitted

txvlanframes_g on page 17-261
Number of good VLAN frames transmitted, exclusive of retried frames

txoversize_g on page 17-262
Number of good and bad frames received

rxframecount_gb on page 17-262
Number of good and bad frames received

rxoctetcount_gb on page 17-263
Number of bytes received, exclusive of preamble, in good and bad frames

rxoctetcount_g on page 17-263
Number of bytes received, exclusive of preamble, only in good frames

rxbroadcastframes_g on page 17-264
Number of good broadcast frames received

rxmulticastframes_g on page 17-264
Number of good multicast frames received

rxcrcerror on page 17-265
Number of frames received with CRC error

rxalignmenterror on page 17-265
Number of frames received with alignment (dribble) error. Valid only in 10/100 mode

rxrunterror on page 17-266
Number of frames received with runt (<64 bytes and CRC error) error

rxjabbererror on page 17-266
Number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames

rxundersize_g on page 17-267
Number of frames received with length less than 64 bytes, without any errors

rxoversize_g on page 17-268
Number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames), without errors

rx64octets_gb on page 17-268
Number of good and bad frames received with length 64 bytes, exclusive of preamble

rx65to127octets_gb on page 17-269
Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble

rx128to255octets_gb on page 17-269
Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble

rx256to511octets_gb on page 17-270
Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble

rx512to1023octets_gb on page 17-270
Number of good and bad frames received with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble

rx1024tomaxoctets_gb on page 17-271
Number of good and bad frames received with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames

rxunicastframes_g on page 17-272
Number of good unicast frames received

rxlengtherror on page 17-272
Number of frames received with length error (length type field not equal to frame size), for all frames with valid length field

rxoutofrangetype on page 17-273
Number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536)

rxpauseframes on page 17-273
Number of good and valid PAUSE frames received

rxfifooverflow on page 17-274
Number of missed received frames due to FIFO overflow

rxvlanframes_gb on page 17-274
Number of good and bad VLAN frames received

rxwatchdogerror on page 17-275
Number of frames received with error due to watchdog timeout error (frames with a data load larger than 2,048 bytes)

rxrcverror on page 17-275
Number of frames received with Receive error or Frame Extension error on the GMII or MII interface.

rxctrlframes_g on page 17-276
Number of received good control frames.

MMC_IPC_Receive_Interrupt_Mask on page 17-276
This register maintains the mask for the interrupt generated from the receive IPC statistic counters.

MMC_IPC_Receive_Interrupt on page 17-283
This register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Checksum Offload Interrupt register is 32-bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (bits[7:0]) must be read to clear the interrupt bit.

rxipv4_gd_frms on page 17-289
Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload

rxipv4_hdrerr_frms on page 17-290
Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors

rxipv4_nopay_frms on page 17-290
Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine

rxipv4_frag_frms on page 17-291
Number of good IPv4 datagrams with fragmentation

rxipv4_udsbl_frms on page 17-291
Number of good IPv4 datagrams received that had a UDP payload with checksum disabled

rxipv6_gd_frms on page 17-292
Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads

rxipv6_hdrerr_frms on page 17-292
Number of IPv6 datagrams received with header errors (length or version mismatch)

rxipv6_nopay_frms on page 17-293
Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers

rxudp_gd_frms on page 17-293
Number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented

rxudp_err_frms on page 17-294
Number of good IP datagrams whose UDP payload has a checksum error

rxtcp_gd_frms on page 17-295
Number of good IP datagrams with a good TCP payload

rxtcp_err_frms on page 17-295
Number of good IP datagrams whose TCP payload has a checksum error

rxicmp_gd_frms on page 17-296
Number of good IP datagrams with a good ICMP payload

rxicmp_err_frms on page 17-296
Number of good IP datagrams whose ICMP payload has a checksum error

rxipv4_gd_octets on page 17-297
Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data

rxipv4_hdrerr_octets on page 17-297
Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter

rxipv4_nopay_octets on page 17-298
Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter

rxipv4_frag_octets on page 17-299
Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter

rxipv4_udsbl_octets on page 17-299
Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes

rxipv6_gd_octets on page 17-300
Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data

rxipv6_hdrerr_octets on page 17-300
Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 headers Length field is used to update this counter

rxipv6_nopay_octets on page 17-301
Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter

rxudp_gd_octets on page 17-301
Number of bytes received in a good UDP segment. This counter does not count IP header bytes

rxudp_err_octets on page 17-302
Number of bytes received in a UDP segment that had checksum errors

rxtcp_gd_octets on page 17-303
Number of bytes received in a good TCP segment

rxtcperroctets on page 17-303
Number of bytes received in a TCP segment with checksum errors

rxicmp_gd_octets on page 17-304
Number of bytes received in a good ICMP segment

rxicmp_err_octets on page 17-304
Number of bytes received in an ICMP segment with checksum errors

L3_L4_Control0 on page 17-305
This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Layer4_Address0 on page 17-307
Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Layer3_Addr0_Reg0 on page 17-308
For IPv4 frames, the Layer 3 Address 0 Register 0 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr1_Reg0 on page 17-309
For IPv4 frames, the Layer 3 Address 1 Register 0 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr2_Reg0 on page 17-310
For IPv4 frames, the Layer 3 Address 2 Register 0 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr3_Reg0 on page 17-311
For IPv4 frames, the Layer 3 Address 3 Register 0 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

L3_L4_Control1 on page 17-312
This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Layer4_Address1 on page 17-314
Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Layer3_Addr0_Reg1 on page 17-315
For IPv4 frames, the Layer 3 Address 0 Register 1 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr1_Reg1 on page 17-316
For IPv4 frames, the Layer 3 Address 1 Register 1 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field

Layer3_Addr2_Reg1 on page 17-317
For IPv4 frames, the Layer 3 Address 2 Register 1 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr3_Reg1 on page 17-318
For IPv4 frames, the Layer 3 Address 3 Register 1 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

L3_L4_Control2 on page 17-318
This register controls the operations of the filter 2 of Layer 3 and Layer 4.

Layer4_Address2 on page 17-321
Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Layer3_Addr0_Reg2 on page 17-322
For IPv4 frames, the Layer 3 Address 0 Register 2 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr1_Reg2 on page 17-323
For IPv4 frames, the Layer 3 Address 1 Register 2 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr2_Reg2 on page 17-324
For IPv4 frames, the Layer 3 Address 2 Register 2 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr3_Reg2 on page 17-325
For IPv4 frames, the Layer 3 Address 3 Register 2 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

L3_L4_Control3 on page 17-326
This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Layer4_Address3 on page 17-328
Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Layer3_Addr0_Reg3 on page 17-329
For IPv4 frames, the Layer 3 Address 0 Register 3 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr1_Reg3 on page 17-330
For IPv4 frames, the Layer 3 Address 1 Register 3 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr2_Reg3 on page 17-331
For IPv4 frames, the Layer 3 Address 2 Register 3 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Layer3_Addr3_Reg3 on page 17-332
For IPv4 frames, the Layer 3 Address 3 Register 3 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Hash_Table_Reg0 on page 17-333
This register contains the first 32 bits of the hash table. The 256-bit Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming frame is passed through the CRC logic and the upper eight bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 8b'10111111 selects Bit 31 of the Hash Table Register 5. The hash value of the destination address is calculated in the following way: 1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32). 2. Perform bitwise reversal for the value obtained in Step 1. 3. Take the upper 8 bits from the value obtained in Step 2. If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Bit 1 (Pass All Multicast) is set in Register 1 (MAC Frame Filter), then all multicast frames are accepted regardless of the multicast hash values. Because the Hash Table register is double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written. Note: Because of double-synchronization, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Hash_Table_Reg1 on page 17-333
This register contains the second 32 bits of the hash table.

Hash_Table_Reg2 on page 17-334
This register contains the third 32 bits of the hash table.

Hash_Table_Reg3 on page 17-334
This register contains the fourth 32 bits of the hash table.

Hash_Table_Reg4 on page 17-335
This register contains the fifth 32 bits of the hash table.

Hash_Table_Reg5 on page 17-335
This register contains the sixth 32 bits of the hash table.

Hash_Table_Reg6 on page 17-336
This register contains the seventh 32 bits of the hash table.

Hash_Table_Reg7 on page 17-336
This register contains the eighth 32 bits of the hash table.

VLAN_Incl_Reg on page 17-337
The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit frames.

VLAN_Hash_Table_Reg on page 17-338
The 16-bit Hash table is used for group address filtering based on VLAN tag when Bit 18 (VTHM) of Register 7 (VLAN Tag Register) is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on Bit 16 (ETV) of VLAN Tag Register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way: 1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32). 2. Perform bitwise reversal for the value obtained in Step 1. 3. Take the upper four bits from the value obtained in Step 2. If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. Because the Hash Table register is double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written. Notes: * Because of double-synchronization, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Timestamp_Control on page 17-339
This register controls the operation of the System Time generator and the processing of PTP packets for timestamping in the Receiver.

Sub_Second_Increment on page 17-344
In the Coarse Update mode (TSCFUPDT bit in Register 448), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In the Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

System_Time_Seconds on page 17-345
The System Time -Seconds register, along with System-TimeNanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to l3_sp_clk).

System_Time_Nanoseconds on page 17-346
The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set, each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.

System_Time_Seconds_Update on page 17-346
The System Time - Seconds Update register, along with the System Time - Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both of these registers before setting the TSINIT or TSUPDT bits in the Timestamp Control register.

System_Time_Nanoseconds_Update on page 17-347
Update system time

Timestamp_Addend on page 17-348
This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in Register 448). This register content is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

Target_Time_Seconds on page 17-349
The Target Time Seconds register, along with Target Time Nanoseconds register, is used to schedule an interrupt event (Register 458[1] when Advanced Timestamping is enabled; otherwise, TS interrupt bit in Register14[9]) when the system time exceeds the value programmed in these registers.

Target_Time_Nanoseconds on page 17-349
Target time

System_Time_Higher_Word_Seconds on page 17-350
System time higher word

Timestamp_Status on page 17-351
Timestamp status. All bits except Bits[27:25] get cleared when the host reads this register.

PPS_Control on page 17-353
Controls timestamp Pulse-Per-Second output

Auxiliary_Timestamp_Nanoseconds on page 17-356
This register, along with Register 461 (Auxiliary Timestamp Seconds Register), gives the 64-bit timestamp stored as auxiliary snapshot. The two registers together form the read port of a 64-bit wide FIFO with a depth of 16. Multiple snapshots can be stored in this FIFO. The ATSNS bits in the Timestamp Status register indicate the fill-level of this FIFO. The top of the FIFO is removed only when the last byte of Register 461 (Auxiliary Timestamp - Seconds Register) is read. In the little-endian mode, this means when Bits[31:24] are read. In big-endian mode, it corresponds to the reading of Bits[7:0] of Register 461 (Auxiliary Timestamp - Seconds Register).

Auxiliary_Timestamp_Seconds on page 17-356
Contains the higher 32 bits (Seconds field) of the auxiliary timestamp.

PPS0_Interval on page 17-357
The PPS0 Interval register contains the number of units of sub-second increment value between the rising edges of PPS0 signal output (ptp_pps_o[0]).

PPS0_Width on page 17-358
The PPS0 Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (ptp_pps_o[0]).

MAC_Address16_High on page 17-358
The MAC Address16 High register holds the upper 16 bits of the 17th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address16 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address16_Low on page 17-362
The MAC Address16 Low register holds the lower 32 bits of the 17th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address17_High on page 17-362
The MAC Address17 High register holds the upper 16 bits of the 18th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address17 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address17_Low on page 17-365
The MAC Address17 Low register holds the lower 32 bits of the 18th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address18_High on page 17-366
The MAC Address18 High register holds the upper 16 bits of the 19th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address18 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address18_Low on page 17-369
The MAC Address18 Low register holds the lower 32 bits of the 19th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address19_High on page 17-370
The MAC Address19 High register holds the upper 16 bits of the 20th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address19 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address19_Low on page 17-373
The MAC Address19 Low register holds the lower 32 bits of the 20th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address20_High on page 17-374
The MAC Address20 High register holds the upper 16 bits of the 21th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address20 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address20_Low on page 17-377
The MAC Address20 Low register holds the lower 32 bits of the 21th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address21_High on page 17-378
The MAC Address21 High register holds the upper 16 bits of the 22th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address21 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address21_Low on page 17-381
The MAC Address21 Low register holds the lower 32 bits of the 22th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address22_High on page 17-382
The MAC Address22 High register holds the upper 16 bits of the 23th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address22 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address22_Low on page 17-385
The MAC Address22 Low register holds the lower 32 bits of the 23th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address23_High on page 17-386
The MAC Address23 High register holds the upper 16 bits of the 24th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address23 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address23_Low on page 17-389
The MAC Address23 Low register holds the lower 32 bits of the 24th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address24_High on page 17-390
The MAC Address24 High register holds the upper 16 bits of the 25th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address24 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address24_Low on page 17-393
The MAC Address24 Low register holds the lower 32 bits of the 25th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address25_High on page 17-394
The MAC Address25 High register holds the upper 16 bits of the 26th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address25 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address25_Low on page 17-397
The MAC Address25 Low register holds the lower 32 bits of the 26th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address26_High on page 17-398
The MAC Address26 High register holds the upper 16 bits of the 27th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address26 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address26_Low on page 17-401
The MAC Address26 Low register holds the lower 32 bits of the 27th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address27_High on page 17-402
The MAC Address27 High register holds the upper 16 bits of the 28th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address27 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address27_Low on page 17-405
The MAC Address27 Low register holds the lower 32 bits of the 28th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address28_High on page 17-406
The MAC Address28 High register holds the upper 16 bits of the 29th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address28 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address28_Low on page 17-409
The MAC Address28 Low register holds the lower 32 bits of the 29th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address29_High on page 17-410
The MAC Address29 High register holds the upper 16 bits of the 30th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address29 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address29_Low on page 17-413
The MAC Address29 Low register holds the lower 32 bits of the 30th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address30_High on page 17-414
The MAC Address30 High register holds the upper 16 bits of the 31th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address30 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address30_Low on page 17-417
The MAC Address30 Low register holds the lower 32 bits of the 31th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address31_High on page 17-418
The MAC Address31 High register holds the upper 16 bits of the 32th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address31 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address31_Low on page 17-421
The MAC Address31 Low register holds the lower 32 bits of the 32th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address32_High on page 17-422
The MAC Address32 High register holds the upper 16 bits of the 33th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address32 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address32_Low on page 17-425
The MAC Address32 Low register holds the lower 32 bits of the 33th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address33_High on page 17-426
The MAC Address33 High register holds the upper 16 bits of the 34th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address33 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address33_Low on page 17-429
The MAC Address33 Low register holds the lower 32 bits of the 34th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address34_High on page 17-430
The MAC Address34 High register holds the upper 16 bits of the 35th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address34 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address34_Low on page 17-433
The MAC Address34 Low register holds the lower 32 bits of the 35th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address35_High on page 17-434
The MAC Address35 High register holds the upper 16 bits of the 36th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address35 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address35_Low on page 17-437
The MAC Address35 Low register holds the lower 32 bits of the 36th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address36_High on page 17-438
The MAC Address36 High register holds the upper 16 bits of the 37th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address36 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address36_Low on page 17-441
The MAC Address36 Low register holds the lower 32 bits of the 37th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address37_High on page 17-442
The MAC Address37 High register holds the upper 16 bits of the 38th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address37 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address37_Low on page 17-445
The MAC Address37 Low register holds the lower 32 bits of the 38th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address38_High on page 17-446
The MAC Address38 High register holds the upper 16 bits of the 39th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address38 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address38_Low on page 17-449
The MAC Address38 Low register holds the lower 32 bits of the 39th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address39_High on page 17-450
The MAC Address39 High register holds the upper 16 bits of the 40th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address39 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address39_Low on page 17-453
The MAC Address39 Low register holds the lower 32 bits of the 40th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address40_High on page 17-454
The MAC Address40 High register holds the upper 16 bits of the 41th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address40 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address40_Low on page 17-457
The MAC Address40 Low register holds the lower 32 bits of the 41th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address41_High on page 17-458
The MAC Address41 High register holds the upper 16 bits of the 42th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address41 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address41_Low on page 17-461
The MAC Address41 Low register holds the lower 32 bits of the 42th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address42_High on page 17-462
The MAC Address42 High register holds the upper 16 bits of the 43th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address42 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address42_Low on page 17-465
The MAC Address42 Low register holds the lower 32 bits of the 43th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address43_High on page 17-466
The MAC Address43 High register holds the upper 16 bits of the 44th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address43 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address43_Low on page 17-469
The MAC Address43 Low register holds the lower 32 bits of the 44th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address44_High on page 17-470
The MAC Address44 High register holds the upper 16 bits of the 45th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address44 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address44_Low on page 17-473
The MAC Address44 Low register holds the lower 32 bits of the 45th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address45_High on page 17-474
The MAC Address45 High register holds the upper 16 bits of the 46th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address45 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address45_Low on page 17-477
The MAC Address45 Low register holds the lower 32 bits of the 46th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address46_High on page 17-478
The MAC Address46 High register holds the upper 16 bits of the 47th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address46 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address46_Low on page 17-481
The MAC Address46 Low register holds the lower 32 bits of the 47th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address47_High on page 17-482
The MAC Address47 High register holds the upper 16 bits of the 48th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address47 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address47_Low on page 17-485
The MAC Address47 Low register holds the lower 32 bits of the 48th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address48_High on page 17-486
The MAC Address48 High register holds the upper 16 bits of the 49th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address48 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address48_Low on page 17-489
The MAC Address48 Low register holds the lower 32 bits of the 49th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address49_High on page 17-490
The MAC Address49 High register holds the upper 16 bits of the 50th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address49 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address49_Low on page 17-493
The MAC Address49 Low register holds the lower 32 bits of the 50th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address50_High on page 17-494
The MAC Address50 High register holds the upper 16 bits of the 51th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address50 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address50_Low on page 17-497
The MAC Address50 Low register holds the lower 32 bits of the 51th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address51_High on page 17-498
The MAC Address51 High register holds the upper 16 bits of the 52th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address51 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address51_Low on page 17-501
The MAC Address51 Low register holds the lower 32 bits of the 52th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address52_High on page 17-502
The MAC Address52 High register holds the upper 16 bits of the 53th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address52 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address52_Low on page 17-505
The MAC Address52 Low register holds the lower 32 bits of the 53th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address53_High on page 17-506
The MAC Address53 High register holds the upper 16 bits of the 54th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address53 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address53_Low on page 17-509
The MAC Address53 Low register holds the lower 32 bits of the 54th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address54_High on page 17-510
The MAC Address54 High register holds the upper 16 bits of the 55th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address54 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address54_Low on page 17-513
The MAC Address54 Low register holds the lower 32 bits of the 55th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address55_High on page 17-514
The MAC Address55 High register holds the upper 16 bits of the 56th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address55 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address55_Low on page 17-517
The MAC Address55 Low register holds the lower 32 bits of the 56th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address56_High on page 17-518
The MAC Address56 High register holds the upper 16 bits of the 57th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address56 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address56_Low on page 17-521
The MAC Address56 Low register holds the lower 32 bits of the 57th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address57_High on page 17-522
The MAC Address57 High register holds the upper 16 bits of the 58th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address57 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address57_Low on page 17-525
The MAC Address57 Low register holds the lower 32 bits of the 58th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address58_High on page 17-526
The MAC Address58 High register holds the upper 16 bits of the 59th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address58 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address58_Low on page 17-529
The MAC Address58 Low register holds the lower 32 bits of the 59th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address59_High on page 17-530
The MAC Address59 High register holds the upper 16 bits of the 60th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address59 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address59_Low on page 17-533
The MAC Address59 Low register holds the lower 32 bits of the 60th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address60_High on page 17-534
The MAC Address60 High register holds the upper 16 bits of the 61th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address60 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address60_Low on page 17-537
The MAC Address60 Low register holds the lower 32 bits of the 61th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address61_High on page 17-538
The MAC Address61 High register holds the upper 16 bits of the 62th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address61 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address61_Low on page 17-541
The MAC Address61 Low register holds the lower 32 bits of the 62th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address62_High on page 17-542
The MAC Address62 High register holds the upper 16 bits of the 63th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address62 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address62_Low on page 17-545
The MAC Address62 Low register holds the lower 32 bits of the 63th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address63_High on page 17-546
The MAC Address63 High register holds the upper 16 bits of the 64th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address63 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address63_Low on page 17-549
The MAC Address63 Low register holds the lower 32 bits of the 64th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address64_High on page 17-550
The MAC Address64 High register holds the upper 16 bits of the 65th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address64 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address64_Low on page 17-553
The MAC Address64 Low register holds the lower 32 bits of the 65th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address65_High on page 17-554
The MAC Address65 High register holds the upper 16 bits of the 66th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address65 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address65_Low on page 17-557
The MAC Address65 Low register holds the lower 32 bits of the 66th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address66_High on page 17-558
The MAC Address66 High register holds the upper 16 bits of the 67th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address66 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address66_Low on page 17-561
The MAC Address66 Low register holds the lower 32 bits of the 67th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address67_High on page 17-562
The MAC Address67 High register holds the upper 16 bits of the 68th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address67 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address67_Low on page 17-565
The MAC Address67 Low register holds the lower 32 bits of the 68th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address68_High on page 17-566
The MAC Address68 High register holds the upper 16 bits of the 69th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address68 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address68_Low on page 17-569
The MAC Address68 Low register holds the lower 32 bits of the 69th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address69_High on page 17-570
The MAC Address69 High register holds the upper 16 bits of the 70th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address69 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address69_Low on page 17-573
The MAC Address69 Low register holds the lower 32 bits of the 70th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address70_High on page 17-574
The MAC Address70 High register holds the upper 16 bits of the 71th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address70 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address70_Low on page 17-577
The MAC Address70 Low register holds the lower 32 bits of the 71th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address71_High on page 17-578
The MAC Address71 High register holds the upper 16 bits of the 72th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address71 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address71_Low on page 17-581
The MAC Address71 Low register holds the lower 32 bits of the 72th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address72_High on page 17-582
The MAC Address72 High register holds the upper 16 bits of the 73th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address72 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address72_Low on page 17-585
The MAC Address72 Low register holds the lower 32 bits of the 73th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address73_High on page 17-586
The MAC Address73 High register holds the upper 16 bits of the 74th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address73 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address73_Low on page 17-589
The MAC Address73 Low register holds the lower 32 bits of the 74th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address74_High on page 17-590
The MAC Address74 High register holds the upper 16 bits of the 75th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address74 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address74_Low on page 17-593
The MAC Address74 Low register holds the lower 32 bits of the 75th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address75_High on page 17-594
The MAC Address75 High register holds the upper 16 bits of the 76th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address75 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address75_Low on page 17-597
The MAC Address75 Low register holds the lower 32 bits of the 76th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address76_High on page 17-598
The MAC Address76 High register holds the upper 16 bits of the 77th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address76 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address76_Low on page 17-601
The MAC Address76 Low register holds the lower 32 bits of the 77th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address77_High on page 17-602
The MAC Address77 High register holds the upper 16 bits of the 78th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address77 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address77_Low on page 17-605
The MAC Address77 Low register holds the lower 32 bits of the 78th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address78_High on page 17-606
The MAC Address78 High register holds the upper 16 bits of the 79th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address78 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address78_Low on page 17-609
The MAC Address78 Low register holds the lower 32 bits of the 79th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address79_High on page 17-610
The MAC Address79 High register holds the upper 16 bits of the 80th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address79 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address79_Low on page 17-613
The MAC Address79 Low register holds the lower 32 bits of the 80th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address80_High on page 17-614
The MAC Address80 High register holds the upper 16 bits of the 81th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address80 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address80_Low on page 17-617
The MAC Address80 Low register holds the lower 32 bits of the 81th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address81_High on page 17-618
The MAC Address81 High register holds the upper 16 bits of the 82th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address81 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address81_Low on page 17-621
The MAC Address81 Low register holds the lower 32 bits of the 82th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address82_High on page 17-622
The MAC Address82 High register holds the upper 16 bits of the 83th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address82 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address82_Low on page 17-625
The MAC Address82 Low register holds the lower 32 bits of the 83th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address83_High on page 17-626
The MAC Address83 High register holds the upper 16 bits of the 84th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address83 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address83_Low on page 17-629
The MAC Address83 Low register holds the lower 32 bits of the 84th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address84_High on page 17-630
The MAC Address84 High register holds the upper 16 bits of the 85th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address84 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address84_Low on page 17-633
The MAC Address84 Low register holds the lower 32 bits of the 85th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address85_High on page 17-634
The MAC Address85 High register holds the upper 16 bits of the 86th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address85 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address85_Low on page 17-637
The MAC Address85 Low register holds the lower 32 bits of the 86th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address86_High on page 17-638
The MAC Address86 High register holds the upper 16 bits of the 87th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address86 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address86_Low on page 17-641
The MAC Address86 Low register holds the lower 32 bits of the 87th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address87_High on page 17-642
The MAC Address87 High register holds the upper 16 bits of the 88th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address87 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address87_Low on page 17-645
The MAC Address87 Low register holds the lower 32 bits of the 88th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address88_High on page 17-646
The MAC Address88 High register holds the upper 16 bits of the 89th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address88 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address88_Low on page 17-649
The MAC Address88 Low register holds the lower 32 bits of the 89th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address89_High on page 17-650
The MAC Address89 High register holds the upper 16 bits of the 90th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address89 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address89_Low on page 17-653
The MAC Address89 Low register holds the lower 32 bits of the 90th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address90_High on page 17-654
The MAC Address90 High register holds the upper 16 bits of the 91th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address90 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address90_Low on page 17-657
The MAC Address90 Low register holds the lower 32 bits of the 91th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address91_High on page 17-658
The MAC Address91 High register holds the upper 16 bits of the 92th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address91 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address91_Low on page 17-661
The MAC Address91 Low register holds the lower 32 bits of the 92th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address92_High on page 17-662
The MAC Address92 High register holds the upper 16 bits of the 93th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address92 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address92_Low on page 17-665
The MAC Address92 Low register holds the lower 32 bits of the 93th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address93_High on page 17-666
The MAC Address93 High register holds the upper 16 bits of the 94th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address93 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address93_Low on page 17-669
The MAC Address93 Low register holds the lower 32 bits of the 94th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address94_High on page 17-670
The MAC Address94 High register holds the upper 16 bits of the 95th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address94 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address94_Low on page 17-673
The MAC Address94 Low register holds the lower 32 bits of the 95th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address95_High on page 17-674
The MAC Address95 High register holds the upper 16 bits of the 96th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address95 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address95_Low on page 17-677
The MAC Address95 Low register holds the lower 32 bits of the 96th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address96_High on page 17-678
The MAC Address96 High register holds the upper 16 bits of the 97th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address96 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address96_Low on page 17-681
The MAC Address96 Low register holds the lower 32 bits of the 97th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address97_High on page 17-682
The MAC Address97 High register holds the upper 16 bits of the 98th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address97 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address97_Low on page 17-685
The MAC Address97 Low register holds the lower 32 bits of the 98th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address98_High on page 17-686
The MAC Address98 High register holds the upper 16 bits of the 99th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address98 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address98_Low on page 17-689
The MAC Address98 Low register holds the lower 32 bits of the 99th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address99_High on page 17-690
The MAC Address99 High register holds the upper 16 bits of the 100th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address99 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address99_Low on page 17-693
The MAC Address99 Low register holds the lower 32 bits of the 100th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address100_High on page 17-694
The MAC Address100 High register holds the upper 16 bits of the 101th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address100 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address100_Low on page 17-697
The MAC Address100 Low register holds the lower 32 bits of the 101th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address101_High on page 17-698
The MAC Address101 High register holds the upper 16 bits of the 102th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address101 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address101_Low on page 17-701
The MAC Address101 Low register holds the lower 32 bits of the 102th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address102_High on page 17-702
The MAC Address102 High register holds the upper 16 bits of the 103th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address102 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address102_Low on page 17-705
The MAC Address102 Low register holds the lower 32 bits of the 103th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address103_High on page 17-706
The MAC Address103 High register holds the upper 16 bits of the 104th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address103 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address103_Low on page 17-709
The MAC Address103 Low register holds the lower 32 bits of the 104th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address104_High on page 17-710
The MAC Address104 High register holds the upper 16 bits of the 105th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address104 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address104_Low on page 17-713
The MAC Address104 Low register holds the lower 32 bits of the 105th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address105_High on page 17-714
The MAC Address105 High register holds the upper 16 bits of the 106th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address105 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address105_Low on page 17-717
The MAC Address105 Low register holds the lower 32 bits of the 106th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address106_High on page 17-718
The MAC Address106 High register holds the upper 16 bits of the 107th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address106 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address106_Low on page 17-721
The MAC Address106 Low register holds the lower 32 bits of the 107th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address107_High on page 17-722
The MAC Address107 High register holds the upper 16 bits of the 108th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address107 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address107_Low on page 17-725
The MAC Address107 Low register holds the lower 32 bits of the 108th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address108_High on page 17-726
The MAC Address108 High register holds the upper 16 bits of the 109th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address108 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address108_Low on page 17-729
The MAC Address108 Low register holds the lower 32 bits of the 109th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address109_High on page 17-730
The MAC Address109 High register holds the upper 16 bits of the 110th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address109 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address109_Low on page 17-733
The MAC Address109 Low register holds the lower 32 bits of the 110th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address110_High on page 17-734
The MAC Address110 High register holds the upper 16 bits of the 111th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address110 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address110_Low on page 17-737
The MAC Address110 Low register holds the lower 32 bits of the 111th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address111_High on page 17-738
The MAC Address111 High register holds the upper 16 bits of the 112th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address111 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address111_Low on page 17-741
The MAC Address111 Low register holds the lower 32 bits of the 112th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address112_High on page 17-742
The MAC Address112 High register holds the upper 16 bits of the 113th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address112 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address112_Low on page 17-745
The MAC Address112 Low register holds the lower 32 bits of the 113th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address113_High on page 17-746
The MAC Address113 High register holds the upper 16 bits of the 114th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address113 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address113_Low on page 17-749
The MAC Address113 Low register holds the lower 32 bits of the 114th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address114_High on page 17-750
The MAC Address114 High register holds the upper 16 bits of the 115th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address114 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address114_Low on page 17-753
The MAC Address114 Low register holds the lower 32 bits of the 115th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address115_High on page 17-754
The MAC Address115 High register holds the upper 16 bits of the 116th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address115 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address115_Low on page 17-757
The MAC Address115 Low register holds the lower 32 bits of the 116th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address116_High on page 17-758
The MAC Address116 High register holds the upper 16 bits of the 117th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address116 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address116_Low on page 17-761
The MAC Address116 Low register holds the lower 32 bits of the 117th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address117_High on page 17-762
The MAC Address117 High register holds the upper 16 bits of the 118th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address117 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address117_Low on page 17-765
The MAC Address117 Low register holds the lower 32 bits of the 118th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address118_High on page 17-766
The MAC Address118 High register holds the upper 16 bits of the 119th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address118 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address118_Low on page 17-769
The MAC Address118 Low register holds the lower 32 bits of the 119th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address119_High on page 17-770
The MAC Address119 High register holds the upper 16 bits of the 120th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address119 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address119_Low on page 17-773
The MAC Address119 Low register holds the lower 32 bits of the 120th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address120_High on page 17-774
The MAC Address120 High register holds the upper 16 bits of the 121th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address120 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address120_Low on page 17-777
The MAC Address120 Low register holds the lower 32 bits of the 121th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address121_High on page 17-778
The MAC Address121 High register holds the upper 16 bits of the 122th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address121 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address121_Low on page 17-781
The MAC Address121 Low register holds the lower 32 bits of the 122th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address122_High on page 17-782
The MAC Address122 High register holds the upper 16 bits of the 123th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address122 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address122_Low on page 17-785
The MAC Address122 Low register holds the lower 32 bits of the 123th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address123_High on page 17-786
The MAC Address123 High register holds the upper 16 bits of the 124th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address123 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address123_Low on page 17-789
The MAC Address123 Low register holds the lower 32 bits of the 124th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address124_High on page 17-790
The MAC Address124 High register holds the upper 16 bits of the 125th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address124 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address124_Low on page 17-793
The MAC Address124 Low register holds the lower 32 bits of the 125th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address125_High on page 17-794
The MAC Address125 High register holds the upper 16 bits of the 126th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address125 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address125_Low on page 17-797
The MAC Address125 Low register holds the lower 32 bits of the 126th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address126_High on page 17-798
The MAC Address126 High register holds the upper 16 bits of the 127th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address126 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address126_Low on page 17-801
The MAC Address126 Low register holds the lower 32 bits of the 127th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Address127_High on page 17-802
The MAC Address127 High register holds the upper 16 bits of the 128th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address127 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

MAC_Address127_Low on page 17-805
The MAC Address127 Low register holds the lower 32 bits of the 128th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

MAC_Configuration

The MAC Configuration register establishes receive and transmit operating modes.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700000
emac1	0xFF702000	0xFF702000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				twokpe RW 0x0	Reserved	cst RW 0x0	tc RW 0x0	wd RW 0x0	jd RW 0x0	be RW 0x0	je RW 0x0	ifg RW 0x0			dcrs RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ps RW 0x0	fes RW 0x0	do RW 0x0	lm RW 0x0	dm RW 0x0	ipc RW 0x0	dr RW 0x0	lud RW 0x0	acs RW 0x0	bl RW 0x0		dc RW 0x0	te RW 0x0	re RW 0x0	prelen RW 0x0	

MAC_Configuration Fields

Bit	Name	Description	Access	Reset
27	twokpe	When set, the MAC considers all frames, with up to 2,000 bytes length, as normal packets. When Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 2K bytes as Giant frames. When this bit is reset and Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 1,518 bytes (1,522 bytes for tagged) as Giant frames. When Bit 20 (Jumbo Enable) is set, setting this bit has no effect on Giant Frame status.	RW	0x0
25	cst	When set, the last 4 bytes (FCS) of all frames of Ether type (type field greater than 0x0600) are stripped and dropped before forwarding the frame to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. **Value** **Description** 0x0 Strip Ether Frames Off 0x1 Strip Ether Frames On	RW	0x0
24	tc	When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII. When this bit is reset, no such information is driven to the PHY. **Value** **Description** 0x1 Enables Transmission of duplex 0x0 Disables Transmission to Phy	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>23</td><td>wd</td><td>When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive frames of up to 16,384 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the frame being received. The MAC cuts off any bytes received after 2,048 bytes.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Enable MAC cutoff > 2048Bytes</td></tr>
<tr><td>0x1</td><td>Disable Watchdog</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>22</td><td>jd</td><td>When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC cuts off TX > 2048</td></tr>
<tr><td>0x1</td><td>Jabber Timer Disabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>21</td><td>be</td><td>When this bit is set, the MAC allows frame bursting during transmission in the GMII half-duplex mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Frame Burst Enable OFF</td></tr>
<tr><td>0x1</td><td>Frame Burst Enable ON</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>20</td><td>je</td><td>When this bit is set, the MAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Report Jumbo Frame Error</td></tr>
<tr><td>0x1</td><td>Ignore Jumbo Frame Error</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>19:17</td><td>ifg</td><td>These bits control the minimum IFG between frames during transmission. In the half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 100). Lower values are not considered. In the 1000-Mbps mode, the minimum IFG supported is 80 bit times (and above).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Inter Frame Gap 96 bit times</td></tr>
<tr><td>0x1</td><td>Inter Frame Gap 88 bit times</td></tr>
<tr><td>0x2</td><td>Inter Frame Gap 80 bit times</td></tr>
<tr><td>0x3</td><td>Inter Frame Gap 72 bit times</td></tr>
<tr><td>0x4</td><td>Inter Frame Gap 64 bit times</td></tr>
<tr><td>0x5</td><td>Inter Frame Gap 56 bit times</td></tr>
<tr><td>0x6</td><td>Inter Frame Gap 48 bit times</td></tr>
<tr><td>0x7</td><td>Inter Frame Gap 40 bit times</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>16</td><td>dcrs</td><td>When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in the half-duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC Tx Gen. Err. No Carrier</td></tr>
<tr><td>0x1</td><td>MAC Tx Ignores (G)MII Crs Signal</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>15</td><td>ps</td><td>This bit selects between GMII and MII
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>GMII 1000 Mbps</td></tr>
<tr><td>0x1</td><td>MII 10/100 Mbps</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
14	fes	This bit selects the speed in the RMII/RGMII interface: * 0: 10 Mbps * 1: 100 Mbps This bit generates link speed encoding when TC (Bit 24) is set in the RGMII, SMII, or SGMII mode. **Value** **Description** 0x0 Speed = 10 Mbps 0x1 Speed = 100 Mbps	RW	0x0
13	do	When this bit is set, the MAC disables the reception of frames when the gmii_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in the full-duplex mode. **Value** **Description** 0x0 MAC Enables Reception of Frames 0x1 MAC Disables Reception of Frames	RW	0x0
12	lm	When this bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Receive clock input is required for the loopback to work properly, because the Transmit clock is not looped-back internally. **Value** **Description** 0x0 Disable Loop Back 0x1 Enable Loop Back	RW	0x0
11	dm	When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously. **Value** **Description** 0x1 MAC Full Duplex Enabled 0x0 MAC Full Duplex Disabled	RW	0x0

Bit	Name	Description	Access	Reset
10	`ipc`	When this bit is set, the MAC calculates the 16-bit ones complement of the ones complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 2526 or 2930 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The MAC also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected). When this bit is reset, this function is disabled. When Type 2 COE is selected, this bit, when set, enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared. **Value** — **Description** 0x1 — Checksum Enabled 0x0 — Checksum Disabled	`RW`	`0x0`
9	`dr`	When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current frame transmission and reports a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is applicable only in the half-duplex mode. **Value** — **Description** 0x1 — MAC attempts one transmission 0x0 — MAC attempts retries per bl Field	`RW`	`0x0`
8	`lud`	This bit indicates whether the link is up or down during the transmission of configuration in the RGMII, SGMII, or SMII interface **Value** — **Description** 0x0 — Link Down 0x1 — Link Up	`RW`	`0x0`

Bit	Name	Description	Access	Reset
7	acs	When this bit is set, the MAC strips the Pad or FCS field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming frames, without modifying them, to the Host. **Value** **Description** 0x0 Disable Automatic Pad CRC Stripping 0x1 Enable Automatic Pad CRC Stripping	RW	0x0
6:5	bl	The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only in the half-duplex mode. * 00: k = min (n, 10) * 01: k = min (n, 8) * 10: k = min (n, 4) * 11: k = min (n, 1) where <i> n </i>= retransmission attempt. The random integer <i> r </i> takes the value in the range 0 <= r < kth power of 2 **Value** **Description** 0x0 k = min (n, 10) 0x1 k = min (n, 8) 0x2 k = min (n, 4) 0x3 k = min (n, 1)	RW	0x0

Bit	Name	Description	Access	Reset
4	dc	When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status, when the transmit state machine is deferred for more than 24,288 bit times in the 10 or 100 Mbps mode. If the MAC is configured for 1000 Mbps operation, or if the Jumbo frame mode is enabled in the 10 or 100 Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier sense signal (CRS) on GMII or MII. Defer time is not cumulative. When the transmitter defers for 10,000 bit times, it transmits, collides, backs off, and then defers again after completion of back-off. The deferral timer resets to 0 and restarts. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode. **Value** **Description** 0x1 Deferral Check Enabled 0x0 Deferral Check Disabled	RW	0x0
3	te	When this bit is set, the transmit state machine of the MAC is enabled for transmission on the GMII or MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames. **Value** **Description** 0x0 MAC transmit state machine disabled 0x1 MAC transmit state machine disabled	RW	0x0
2	re	When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the GMII or MII. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and does not receive any further frames from the GMII or MII. **Value** **Description** 0x0 MAC receive state machine disabled 0x1 MAC receive state machine enabled	RW	0x0

Bit	Name	Description	Access	Reset
1:0	`prelen`	These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating **Value** — **Description** 0x0 — Preamble 7 Bytes 0x1 — Preamble 5 Bytes 0x2 — Preamble 3 Bytes	RW	0x0

MAC_Frame_Filter

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700004
emac1	0xFF702000	0xFF702004

Offset: 0x4

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ra RW 0x0	Reserved									dntu RW 0x0	ipfe RW 0x0	Reserved			vtfe RW 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					hpf RW 0x0	saf RW 0x0	saif RW 0x0	pcf RW 0x0		dbf RW 0x0	pm RW 0x0	daif RW 0x0	hmc RW 0x0	huc RW 0x0	pr RW 0x0

MAC_Frame_Filter Fields

Bit	Name	Description	Access	Reset
31	`ra`	When this bit is set, the MAC Receiver block passes all received frames, irrespective of whether they pass the address filter or not, to the Application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver block passes only those frames to the Application that pass the SA or DA address filter. **Value** — **Description** 0x0 — Receive All off 0x1 — Receive All On	RW	0x0
21	`dntu`	When set, this bit enables the MAC to drop the non-TCP or UDP over IP frames. The MAC forward only those frames that are processed by the Layer 4 filter. When reset, this bit enables the MAC to forward all non-TCP or UDP over IP frames. **Value** — **Description** 0x0 — Forward all non-TCP or UDP over IP frames 0x1 — Drop non-TCP or UDP over IP frames	RW	0x0
20	`ipfe`	When set, this bit enables the MAC to drop frames that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect. When reset, the MAC forwards all frames irrespective of the match status of the Layer 3 and Layer 4 filters. **Value** — **Description** 0x0 — Forward all frames irrespective of the match status of Layer 3 and Layer 4 filters 0x1 — Drop frames that do not match the enabled Layer 3 and Layer 4 filters	RW	0x0

Bit	Name	Description	Access	Reset
16	`vtfe`	When set, this bit enables the MAC to drop VLAN tagged frames that do not match the VLAN Tag comparison. When reset, the MAC forwards all frames irrespective of the match status of the VLAN Tag. **Value** **Description** 0x0 Forward all frames irrespective of the match status of the VLAN tag 0x1 Drop VLAN tagged frames that do not match the VLAN Tag comparison	RW	0x0
10	`hpf`	When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits. When this bit is low and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter. **Value** **Description** 0x0 Hash or Perfect Filter off 0x1 Hash or Perfect Filter on	RW	0x0
9	`saf`	When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SA Match bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the MAC drops the frame. When this bit is reset, the MAC forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison. **Value** **Description** 0x0 SA Filter Off 0x1 SA Filter On	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>8</td><td>saif</td><td>When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter. When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>SA Nomatch Fail</td></tr>
<tr><td>0x1</td><td>SA Match Fail</td></tr>
</table>
</td><td>RW</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>7:6</td><td>pcf</td><td>These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). * 00: MAC filters all control frames from reaching the application. * 01: MAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter. * 10: MAC forwards all control frames to application even if they fail the Address Filter. * 11: MAC forwards control frames that pass the Address Filter. The following conditions should be true for the PAUSE control frames processing: * Condition 1: The MAC is in the full-duplex mode and flow control is enabled by setting Bit 2 (RFE) of Register 6 (Flow Control Register) to 1. * Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 when Bit 3 (UP) of the Register 6 (Flow Control Register) is set. * Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001. This field should be set to 01 only when the Condition 1 is true, that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 10 or 11 (as required by the application).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC filters all control frames</td></tr>
<tr><td>0x1</td><td>MAC forwards all control frames except PAUSE</td></tr>
<tr><td>0x2</td><td>MAC forwards all control frames fail addrflt</td></tr>
<tr><td>0x3</td><td>MAC forwards control frames that pass addrflt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>5</td><td>dbf</td><td>When this bit is set, the AFM block filters all incoming broadcast frames. In addition, it overrides all other filter settings. When this bit is reset, the AFM block passes all received broadcast frames.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Pass All Broadcast Frames</td></tr>
<tr><td>0x1</td><td>Filter All Broadcast Frames</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
4	pm	When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit. **Value** — **Description** 0x0 — Allows Filter of MC Frames 0x1 — All Rcvd MC Frames Pass	RW	0x0
3	daif	When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed. **Value** — **Description** 0x0 — Normal Inverse Filter 0x1 — Address Check Block Inverse Filter	RW	0x0
2	hmc	When set, MAC performs destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers. **Value** — **Description** 0x0 — MAC Filters with Hash Table 0x1 — MAC Filters with Compare	RW	0x0
1	huc	When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers. **Value** — **Description** 0x1 — MAC Filters with Hash Table 0x0 — MAC Filters with Compare	RW	0x0

Bit	Name	Description	Access	Reset
0	pr	When this bit is set, the Address Filter block passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set. **Value** — **Description** 0x1 — All Incoming Frames Passed 0x0 — Clear SA DA Status Bits	RW	0x0

GMII_Address

The GMII Address register controls the management cycles to the external PHY through the management interface.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700010
emac1	0xFF702000	0xFF702010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pa RW 0x0					gr RW 0x0					cr RW 0x0				gw RW 0x0	gb RW 0x0

GMII_Address Fields

Bit	Name	Description	Access	Reset
15:11	pa	This field indicates which of the 32 possible PHY devices are being accessed. For RevMII, this field gives the PHY Address of the RevMII block.	RW	0x0
10:6	gr	These bits select the desired GMII register in the selected PHY device. For RevMII, these bits select the desired CSR register in the RevMII Registers set.	RW	0x0

Bit	Name	Description	Access	Reset
5:2	cr	The CSR Clock Range selection determines the frequency of the MDC clock according to the l3_sp_clk frequency used in your design. The suggested range of l3_sp_clk frequency applicable for each value (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz. When Bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when l3_sp_clk is of 100 MHz frequency and you program these bits as 1010, then the resultant MDC clock is of 12.5 MHz which is outside the limit of IEEE 802.3 specified range. Only use the values larger than 7 if the interfacing chips support faster MDC clocks. **Value** **Description** 0x0 l3_sp_clk 60-100Mhz and MDC clock = l3_sp_clk/42 0x1 l3_sp_clk 100-150Mhz and MDC clock = l3_sp_clk/62 0x2 l3_sp_clk 25-35Mhz and MDC clock = l3_sp_clk/16 0x3 l3_sp_clk 35-60Mhz and MDC clock = l3_sp_clk/26 0x4 l3_sp_clk 150-250Mhz and MDC clock = l3_sp_clk/102 0x5 l3_sp_clk 250-300Mhz and MDC clock = l3_sp_clk/124 0x8 l3_sp_clk/4 0x9 l3_sp_clk/6 0xa l3_sp_clk/8 0xb l3_sp_clk/10 0xc l3_sp_clk/12 0xd l3_sp_clk/14 0xe l3_sp_clk/16 0xf l3_sp_clk/18	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>1</td><td>gw</td><td>When set, this bit indicates to the PHY or RevMII that this is a Write operation using the GMII Data register. If this bit is not set, it indicates that this is a Read operation, that is, placing the data in the GMII Data register.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Gmii Write Operation</td></tr><tr><td>0x1</td><td>Gmii Read Operation</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>0</td><td>gb</td><td>This bit should read logic 0 before writing to Register 4 and Register 5. During a PHY or RevMII register access, the software sets this bit to 1'b1 to indicate that a Read or Write access is in progress. The Register 5 is invalid until this bit is cleared by the MAC. Therefore, Register 5 (GMII Data) should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of Register 5 are not valid until this bit is cleared. The subsequent read or write operation should happen only after the previous operation is complete. Because there is no acknowledgment from the PHY to MAC after a read or write operation is completed, there is no change in the functionality of this bit even when the PHY is not present.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Not Busy</td></tr><tr><td>0x1</td><td>Busy</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

GMII_Data

The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4 (GMII Address Register). This register also stores the Read data from the PHY register located at the address specified by Register 4.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700014
emac1	0xFF702000	0xFF702014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gd RW 0x0															

GMII_Data Fields

Bit	Name	Description	Access	Reset
15:0	gd	This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY or RevMII before a Management Write operation.	RW	0x0

Flow_Control

The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC's Flow control block. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700018
emac1	0xFF702000	0xFF702018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
pt RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dzpq RW 0x0	Reserved	plt RW 0x0		up RW 0x0	rfe RW 0x0	tfe RW 0x0	fca_bpa RW 0x0

Flow_Control Fields

Bit	Name	Description	Access	Reset
31:16	pt	This field holds the value to be used in the Pause Time field in the transmit control frame. Because the Pause Time bits are double-synchronized to the (G) MII clock domain, then consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.	RW	0x0
7	dzpq	When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled. **Value** **Description** 0x1 Disable Auto Gen. of Zero-Quanta Pause 0x0 Enable Auto Gen. of Zero-Quanta Pause	RW	0x0
5:4	plt	This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256 - 28) slot times after the first PAUSE frame is transmitted. The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. **Value** **Description** 0x0 Pause time - 4 slot times 0x1 Pause time - 28 slot times 0x2 Pause time - 144 slot times 0x3 Pause time - 256 slot times	RW	0x0

Bit	Name	Description	Access	Reset
3	`up`	When this bit is set, then in addition to the detecting Pause frames with the unique multicast address, the MAC detects the Pause frames with the station's unicast address specified in the MAC Address0 High Register and MAC Address0 Low Register. When this bit is reset, the MAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard. **Value** — **Description** 0x0 — MAC Detects Pause MCA 0x1 — MAC Detects Pause MCA and UCA	`RW`	`0x0`
2	`rfe`	When this bit is set, the MAC decodes the received Pause frame and disables its transmitter for a specified (Pause) time. When this bit is reset, the decode function of the Pause frame is disabled. **Value** — **Description** 0x0 — Decode Func. of Pause Frame Disabled 0x1 — MAC decodes the received Pause	`RW`	`0x0`
1	`tfe`	In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause frames. In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled. **Value** — **Description** 0x0 — Transmit Flow Control Disable 0x1 — Transmit Flow Control Enable	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	fca_bpa	This bit initiates a Pause Control frame in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. In the full-duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared. In the half-duplex mode, when this bit is set (and TFE is set) , then backpressure is asserted by the MAC. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled. **Value** / **Description** 0x0 — Pause Ctrl Frame and BPA off 0x1 — Init Pause Ctrl Frame and BPA	RW	0x0

VLAN_Tag

The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. Because the VLAN Tag register is double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70001C
emac1	0xFF702000	0xFF70201C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												vthm RW 0x0	esvl RW 0x0	vtim RW 0x0	etv RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vl RW 0x0															

VLAN_Tag Fields

Bit	Name	Description	Access	Reset
19	vthm	When set, the most significant four bits of the VLAN tag's CRC are used to index the content of Register 354 (VLAN Hash Table Register). A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the frame matched the VLAN hash table. When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison. When reset, the VLAN Hash Match operation is not performed.	RW	0x0
18	esvl	When this bit is set, the MAC transmitter and receiver also consider the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames.	RW	0x0
17	vtim	When set, this bit enables the VLAN Tag inverse matching. The frames that do not have matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The frames with matched VLAN Tag are marked as matched.	RW	0x0
16	etv	When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering. **Value** — **Description** 0x0 — Disable 12-Bit VLAN Tag Compare 0x1 — Enable 12-Bit VLAN Tag Compare	RW	0x0

Bit	Name	Description	Access	Reset
15:0	vl	This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field: * Bits [15:13]: User Priority * Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) * Bits[11:0]: VLAN tag's VLAN Identifier (VID) field When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison. If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88a8 as VLAN frames.	RW	0x0

Version

The Version registers identifies the version of the EMAC. This register contains two bytes: one specified by Synopsys to identify the core release number, and the other specified by Altera.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700020
emac1	0xFF702000	0xFF702020

Offset: 0x20

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
userver RO 0x10								snpsver RO 0x37							

Version Fields

Bit	Name	Description	Access	Reset
15:8	userver	Altera-defined Version	RO	0x10
7:0	snpsver	Synopsys-defined Version (3.7)	RO	0x37

Debug

The Debug register gives the status of all main blocks of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700024

Module Instance	Base Address	Register Address
emac1	0xFF702000	0xFF702024

Offset: 0x24

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						txstsfsts RO 0x0	txfsts RO 0x0	Reserved	twcsts RO 0x0	trcsts RO 0x0		txpaused RO 0x0	tfcfcsts RO 0x0		tpests RO 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						rxfsts RO 0x0		Reserved	rrcsts RO 0x0		rwcsts RO 0x0	Reserved	rfcfcsts RO 0x0		rpests RO 0x0

Debug Fields

Bit	Name	Description	Access	Reset
25	txstsfsts	When high, this bit indicates that the MTL TxStatus FIFO is full. Therefore, the MTL cannot accept any more frames for transmission. **Value** **Description** 0x0 MTL TxStatus FIFO Not Full Status 0x1 MTL TxStatus FIFO Full Status	RO	0x0
24	txfsts	When high, this bit indicates that the MTL Tx FIFO is not empty and some data is left for transmission. **Value** **Description** 0x0 MTL Tx FIFO Empty 0x1 MTL Tx FIFO Not Empty	RO	0x0
22	twcsts	When high, this bit indicates that the MTL Tx FIFO Write Controller is active and transferring data to the Tx FIFO. **Value** **Description** 0x0 Tx FIFO Write Ctrl Inactive 0x1 Tx FIFO Write Ctrl Active	RO	0x0

Bit	Name	Description	Access	Reset
21:20	`trcsts`	This field indicates the state of the Tx FIFO Read Controller **Value** — **Description** 0x0 — Idle State 0x1 — Read State (transferring data to the MAC transmitter) 0x2 — Waiting for TxStatus from the MAC transmitter 0x3 — Writing the received TxStatus or flushing the Tx FIFO	RO	0x0
19	`txpaused`	When high, this bit indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission. **Value** — **Description** 0x0 — MAC Transmitter Pause Disabled 0x1 — MAC Transmitter Pause Condition	RO	0x0
18:17	`tfcsts`	This field indicates the state of the MAC Transmit Frame Controller block **Value** — **Description** 0x0 — Idle State 0x1 — Waiting Prev. State or IFG 0x2 — Generating Tx Pause 0x3 — Tx Input Frame	RO	0x0
16	`tpests`	When high, this bit indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data and is not in the IDLE state. **Value** — **Description** 0x0 — Idle State 0x1 — Actively Transmitting Data	RO	0x0

Bit	Name	Description	Access	Reset
9:8	`rxfsts`	This field gives the status of the fill-level of the Rx FIFO. **Value** **Description** 0x0 Rx FIFO Empty 0x1 Rx FIFO fill-level below flow-control deactivate thres. 0x2 Rx FIFO fill-level above flow-control activate thres. 0x3 Rx FIFO Full	RO	0x0
6:5	`rrcsts`	This field gives the state of the Rx FIFO read Controller **Value** **Description** 0x0 IDLE State 0x1 Reading Frame Data 0x2 Reading Frame Status (or timestamp) 0x3 Flushing Frame Data and Status	RO	0x0
4	`rwcsts`	When high, this bit indicates that the MTL Rx FIFO Write Controller is active and is transferring a received frame to the FIFO. **Value** **Description** 0x0 MTL Rx Fifo Controller Non-Active Status 0x1 MTL Rx Fifo Controller Active Status	RO	0x0
2:1	`rfcfcsts`	When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module. **Value** **Description** 0x0 Disable Active State FIFO Read Write 0x1 Enable Active State FIFO Read Write	RO	0x0

Bit	Name	Description	Access	Reset
0	`rpests`	When high, this bit indicates that the MAC GMII or MII receive protocol engine is actively receiving data and not in IDLE state. **Value** **Description** 0x0 Idle State 0x1 Protocol Engine Active	RO	0x0

LPI_Control_Status

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700030
emac1	0xFF702000	0xFF702030

Offset: `0x30`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												lpitxa RW 0x0	plsen RW 0x0	pls RW 0x0	lpien RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						rlpist RO 0x0	tlpist RO 0x0	Reserved				rlpiex RO 0x0	rlpien RO 0x0	tlpiex RO 0x0	tlpien RO 0x0

LPI_Control_Status Fields

Bit	Name	Description	Access	Reset
19	lpitxa	This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side. This bit is not functional in the GMAC-CORE configuration in which the Tx clock gating is done during the LPI mode. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding frames (in the core) and pending frames (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any frame for transmission or the application issues a TX FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If TX FIFO Flush is set, in Bit 20 of Register 6 (Operation Mode Register), when the MAC is in the LPI mode, the MAC exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode. **Value** — **Description** 0x0 — LPI TX Automate Disabled 0x1 — LPI TX Automate Enabled	RW	0x0
18	plsen	This bit enables the link status received on the RGMII receive paths to be used for activating the LPI LS TIMER. When set, the MAC uses the link-status bits of Register 54 (SGMII/RGMII/SMII Status Register) and Bit 17 (PLS) for the LPI LS Timer trigger. When cleared, the MAC ignores the link-status bits of Register 54 and takes only the PLS bit. **Value** — **Description** 0x0 — MAC Ignores Link Status Bits 0x1 — MAC Uses Link Status Bits	RW	0x0
17	pls	This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (okay) at least for the time indicated by the LPI LS TIMER. When set, the link is considered to be okay (up) and when reset, the link is considered to be down. **Value** — **Description** 0x0 — Link Down 0x1 — Link Up (okay)	RW	0x0

Bit	Name	Description	Access	Reset
16	lpien	When set, this bit instructs the MAC Transmitter to enter the LPI state. When reset, this bit instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission. **Value** **Description** 0x0 MAC Transmitter exit LPI State 0x1 MAC Transmitter enters LPI State	RW	0x0
9	rlpist	When set, this bit indicates that the MAC is receiving the LPI pattern on the GMII or MII interface. **Value** **Description** 0x0 MAC is not receiving LPI Pattern 0x1 MAC receiving LPI Pattern	RO	0x0
8	tlpist	When set, this bit indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface. **Value** **Description** 0x0 MAC Transmitting LPI Pattern 0x1 MAC Transmitting LPI Pattern	RO	0x0
3	rlpiex	When set, this bit indicates that the MAC Receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register. Note: This bit may not get set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than 3 clock cycles of l3_sp_clk. **Value** **Description** 0x0 MAC RX receiving LPI Patterns 0x1 MAC RX Stopped receiving LPI Patterns	RO	0x0

Bit	Name	Description	Access	Reset
2	`rlpien`	When set, this bit indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register. Note: This bit may not get set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than 3 clock cycles of l3_sp_clk. **Value** **Description** 0x0 MAC Receiver Not In LPI State 0x1 MAC Receiver In LPI State	`RO`	`0x0`
1	`tlpiex`	When set, this bit indicates that the MAC transmitter has exited the LPI state after the user has cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register. **Value** **Description** 0x0 MAC Transmitter Non LPI State 0x1 MAC Transmitter Exited LPI State	`RO`	`0x0`
0	`tlpien`	When set, this bit indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register. **Value** **Description** 0x0 MAC Transmitter Not in LPI State 0x1 MAC Transmitter Entered LPI State	`RO`	`0x0`

LPI_Timers_Control

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700034`
`emac1`	`0xFF702000`	`0xFF702034`

Offset: `0x34`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						lst RW 0x3E8									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
twt RW 0x0															

LPI_Timers_Control Fields

Bit	Name	Description	Access	Reset
25:16	lst	This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.	RW	0x3E8
15:0	twt	This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.	RW	0x0

Interrupt_Status

The Interrupt Status register identifies the events in the MAC that can generate interrupt. All interrupt events are generated only when the corresponding optional feature is enabled.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700038
emac1	0xFF702000	0xFF702038

Offset: 0x38

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					lpiis RO 0x0	tsis RO 0x0	Reserved	mmcrxipis RO 0x0	mmctxis RO 0x0	mmcrxis RO 0x0	mmcis RO 0x0	Reserved	pcsancis RO 0x0	pcslchgis RO 0x0	rgsmiiis RO 0x0

Interrupt_Status Fields

Bit	Name	Description	Access	Reset
10	lpiis	This bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared on reading Bit 0 of Register 12 (LPI Control and Status Register). In all other modes, this bit is reserved. **Value** / **Description** 0x0 LPI Interrupt Status Disabled 0x1 LPI Interrupt Status Enabled	RO	0x0
9	tsis	This bit is set when any of the following conditions is true: * The system time value equals or exceeds the value specified in the Target Time High and Low registers. * There is an overflow in the seconds register. * The Auxiliary snapshot trigger is asserted. This bit is cleared on reading Bit 0 of the Register 458 (Timestamp Status Register). When set, this bit indicates that the system time value is equal to or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved. **Value** / **Description** 0x0 Timestamp Interrupt Status Disabled 0x1 Timestamp Interrupt Status Enabled	RO	0x0
7	mmcrxipis	This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. **Value** / **Description** 0x0 MMC Receive Checksum Offload Interrupt Status Disabled 0x1 MMC Receive Checksum Offload Interrupt Status Enabled	RO	0x0

Bit	Name	Description	Access	Reset
6	mmctxis	This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. **Value** — **Description** 0x0 — MMC Transmit Interrupt Status Disabled 0x1 — MMC Transmit Interrupt Status Enabled	RO	0x0
5	mmcrxis	This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. **Value** — **Description** 0x0 — MMC Receive Interrupt Status Disabled 0x1 — MMC Receive Interrupt Status Enabled	RO	0x0
4	mmcis	This bit is set high when any of the Bits [7:5] is set high and cleared only when all of these bits are low. **Value** — **Description** 0x0 — MMC Interrupt Status Disabled 0x1 — MMC Interrupt Status Enabled	RO	0x0
2	pcsancis	This bit is set when the Auto-negotiation is completed in the TBI, RTBI, or SGMII PHY interface (Bit 5 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation to the AN Status register. This bit is valid only when you select the SGMII PHY interface during operation.	RO	0x0
1	pcslchgis	This bit is set because of any change in Link Status in the TBI, RTBI, or SGMII PHY interface (Bit 2 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation on the AN Status register. This bit is valid only when you select the SGMII PHY interface during operation.	RO	0x0

Bit	Name	Description	Access	Reset
0	`rgsmiiis`	This bit is set because of any change in value of the Link Status of RGMII or SMII interface (Bit 3 in Register 54 (SGMII/RGMII/SMII Status Register)). This bit is cleared when you perform a read operation on the SGMII/RGMII/SMII Status Register. **Value** — **Description** 0x0 — Link No Change 0x1 — Link Change	RO	0x0

Interrupt_Mask

The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70003C
emac1	0xFF702000	0xFF70203C

Offset: 0x3C

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					lpiim RW 0x0	tsim RW 0x0	Reserved						pcsancim RO 0x0	pcslchgim RO 0x0	rgsmiiim RW 0x0

Interrupt_Mask Fields

Bit	Name	Description	Access	Reset
10	`lpiim`	When set, this bit disables the assertion of the interrupt signal because of the setting of the LPI Interrupt Status bit in Register 14 (Interrupt Status Register). **Value** — **Description** 0x0 — LPI Interrupt Mask Disabled 0x1 — LPI Interrupt Mask Enabled	RW	0x0

Bit	Name	Description	Access	Reset
9	`tsim`	When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit in Register 14 (Interrupt Status Register). **Value** **Description** 0x0 Timestamp Interrupt Mask Disabled 0x1 Timestamp Interrupt Mask Enabled	`RW`	`0x0`
2	`pcsancim`	When set, this bit disables the assertion of the interrupt signal because of the setting of PCS Auto-negotiation complete bit in Register 14 (Interrupt Status Register).	`RO`	`0x0`
1	`pcslchgim`	When set, this bit disables the assertion of the interrupt signal because of the setting of the PCS Link-status changed bit in Register 14 (Interrupt Status Register).	`RO`	`0x0`
0	`rgsmiiim`	When set, this bit disables the assertion of the interrupt signal because of the setting of the RGMII or SMII Interrupt Status bit in Register 14 (Interrupt Status Register). **Value** **Description** 0x0 RGMII or SMII Interrupt Mask Disable 0x1 RGMII or SMII Interrupt Mask Enable	`RW`	`0x0`

MAC_Address0_High

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211. Because the MAC address registers are double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700040`
`emac1`	`0xFF702000`	`0xFF702040`

Offset: `0x40`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RO 0x1	Reserved														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address0_High Fields

Bit	Name	Description	Access	Reset
31	ae	This bit is always set to 1.	RO	0x1
15:0	addrhi	This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.	RW	0xFFFF

MAC_Address0_Low

The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700044
emac1	0xFF702000	0xFF702044

Offset: 0x44

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address0_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.	RW	0xFFFFFFFF

MAC_Address1_High

The MAC Address1 High register holds the upper 16 bits of the 2nd 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700048
emac1	0xFF702000	0xFF702048

Offset: 0x48

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address1_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 2nd MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address1[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>25</td><td>mbc_1</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 2nd 6-byte MAC address.	RW	0xFFFF

MAC_Address1_Low

The MAC Address1 Low register holds the lower 32 bits of the 2nd 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70004C
emac1	0xFF702000	0xFF70204C

Offset: `0x4C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address1_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 2nd 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address2_High

The MAC Address2 High register holds the upper 16 bits of the 3rd 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address2 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700050
emac1	0xFF702000	0xFF702050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address2_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 3rd MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address2[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address2[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 3rd 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address2_Low

The MAC Address2 Low register holds the lower 32 bits of the 3rd 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700054`
`emac1`	`0xFF702000`	`0xFF702054`

Offset: `0x54`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address2_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 3rd 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFFF`

MAC_Address3_High

The MAC Address3 High register holds the upper 16 bits of the 4th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address3 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700058
emac1	0xFF702000	0xFF702058

Offset: 0x58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address3_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 4th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address3[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address3[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 4th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address3_Low

The MAC Address3 Low register holds the lower 32 bits of the 4th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70005C`
`emac1`	`0xFF702000`	`0xFF70205C`

Offset: `0x5C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address3_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 4th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address4_High

The MAC Address4 High register holds the upper 16 bits of the 5th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address4 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700060
emac1	0xFF702000	0xFF702060

Offset: `0x60`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address4_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 5th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address4[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address4[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 5th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address4_Low

The MAC Address4 Low register holds the lower 32 bits of the 5th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700064`
`emac1`	`0xFF702000`	`0xFF702064`

Offset: `0x64`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
</table>

MAC_Address4_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 5th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFFF`

MAC_Address5_High

The MAC Address5 High register holds the upper 16 bits of the 6th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address5 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700068
emac1	0xFF702000	0xFF702068

Offset: 0x68

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address5_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 6th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address5[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address5[47:0] is used to compare with the DA fields of the received frame.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>MAC address compare disabled</td></tr><tr><td>0x1</td><td>MAC address compare enabled</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 6th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address5_Low

The MAC Address5 Low register holds the lower 32 bits of the 6th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70006C`
`emac1`	`0xFF702000`	`0xFF70206C`

Offset: `0x6C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address5_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 6th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address6_High

The MAC Address6 High register holds the upper 16 bits of the 7th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address6 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700070
emac1	0xFF702000	0xFF702070

Offset: 0x70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address6_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 7th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	`sa`	When this bit is enabled, the MAC Address6[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address6[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 7th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address6_Low

The MAC Address6 Low register holds the lower 32 bits of the 7th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700074`
`emac1`	`0xFF702000`	`0xFF702074`

Offset: `0x74`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address6_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 7th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address7_High

The MAC Address7 High register holds the upper 16 bits of the 8th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address7 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700078
emac1	0xFF702000	0xFF702078

Offset: 0x78

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address7_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 8th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	`sa`	When this bit is enabled, the MAC Address7[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address7[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address7 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 8th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address7_Low

The MAC Address7 Low register holds the lower 32 bits of the 8th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70007C`
`emac1`	`0xFF702000`	`0xFF70207C`

Offset: `0x7C`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
</table>

MAC_Address7_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 8th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFFF`

MAC_Address8_High

The MAC Address8 High register holds the upper 16 bits of the 9th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address8 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700080`
`emac1`	`0xFF702000`	`0xFF702080`

Offset: `0x80`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address8_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 9th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address8[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address8[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 9th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address8_Low

The MAC Address8 Low register holds the lower 32 bits of the 9th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700084`
`emac1`	`0xFF702000`	`0xFF702084`

Offset: `0x84`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address8_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 9th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address9_High

The MAC Address9 High register holds the upper 16 bits of the 10th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address9 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700088`
`emac1`	`0xFF702000`	`0xFF702088`

Offset: `0x88`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address9_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 10th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address9[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address9[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 10th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address9_Low

The MAC Address9 Low register holds the lower 32 bits of the 10th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70008C`
`emac1`	`0xFF702000`	`0xFF70208C`

Offset: `0x8C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address9_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 10th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address10_High

The MAC Address10 High register holds the upper 16 bits of the 11th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address10 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700090
emac1	0xFF702000	0xFF702090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address10_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 11th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	`sa`	When this bit is enabled, the MAC Address10[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address10[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 11th 6-byte MAC address.	RW	0xFFFF

MAC_Address10_Low

The MAC Address10 Low register holds the lower 32 bits of the 11th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700094
emac1	0xFF702000	0xFF702094

Offset: 0x94

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address10_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 11th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address11_High

The MAC Address11 High register holds the upper 16 bits of the 12th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address11 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700098
emac1	0xFF702000	0xFF702098

Offset: 0x98

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address11_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 12th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address11[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address11[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 12th 6-byte MAC address.	RW	0xFFFF

MAC_Address11_Low

The MAC Address11 Low register holds the lower 32 bits of the 12th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70009C
emac1	0xFF702000	0xFF70209C

Offset: 0x9C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address11_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 12th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFF FFF

MAC_Address12_High

The MAC Address12 High register holds the upper 16 bits of the 13th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address12 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7000A0
emac1	0xFF702000	0xFF7020A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address12_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 13th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address12[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address12[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 13th 6-byte MAC address.	RW	0xFFFF

MAC_Address12_Low

The MAC Address12 Low register holds the lower 32 bits of the 13th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7000A4`
`emac1`	`0xFF702000`	`0xFF7020A4`

Offset: `0xA4`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">addrlo
RW 0xFFFFFFFF</td></tr>
</table>

MAC_Address12_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 13th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address13_High

The MAC Address13 High register holds the upper 16 bits of the 14th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address13 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7000A8`
`emac1`	`0xFF702000`	`0xFF7020A8`

Offset: `0xA8`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
addrhi RW 0xFFFF															

MAC_Address13_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 14th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address13[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address13[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 14th 6-byte MAC address.	RW	0xFFFF

MAC_Address13_Low

The MAC Address13 Low register holds the lower 32 bits of the 14th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7000AC`
`emac1`	`0xFF702000`	`0xFF7020AC`

Offset: `0xAC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address13_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 14th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFF FFF

MAC_Address14_High

The MAC Address14 High register holds the upper 16 bits of the 15th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address14 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7000B0
emac1	0xFF702000	0xFF7020B0

Offset: 0xB0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address14_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 15th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	`sa`	When this bit is enabled, the MAC Address14[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address14[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 15th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address14_Low

The MAC Address14 Low register holds the lower 32 bits of the 15th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7000B4`
`emac1`	`0xFF702000`	`0xFF7020B4`

Offset: `0xB4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`addrlo` `RW 0xFFFFFFFF`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`addrlo` `RW 0xFFFFFFFF`															

MAC_Address14_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 15th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	`RW`	`0xFFFFFFFF`

MAC_Address15_High

The MAC Address15 High register holds the upper 16 bits of the 16th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7000B8
emac1	0xFF702000	0xFF7020B8

Offset: 0xB8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address15_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 16th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0

Bit	Name	Description	Access	Reset
30	sa	When this bit is enabled, the MAC Address15[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address15[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>25</td><td>mbc_1</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 16th 6-byte MAC address.	RW	0xFFFF

MAC_Address15_Low

The MAC Address15 Low register holds the lower 32 bits of the 16th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7000BC`
`emac1`	`0xFF702000`	`0xFF7020BC`

Offset: `0xBC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address15_Low Fields

Bit	Name	Description	Access	Reset
31:0	`addrlo`	This field contains the lower 32 bits of the 16th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

SGMII_RGMII_SMII_Control_Status

The SGMII/RGMII/SMII Status register indicates the status signals received by the RGMII interface (selected at reset) from the PHY.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7000D8
emac1	0xFF702000	0xFF7020D8

Offset: 0xD8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												lnksts RO 0x0	lnkspeed RO 0x0		lnkmod RO 0x0

SGMII_RGMII_SMII_Control_Status Fields

Bit	Name	Description	Access	Reset
3	lnksts	This bit indicates whether the link is up (1'b1) or down (1'b0). **Value** / **Description** 0x0 Linkdown 0x1 Linkup	RO	0x0
2:1	lnkspeed	This bit indicates the current speed of the link. Bit 2 is reserved when the MAC is configured for the SMII PHY interface. **Value** / **Description** 0x0 Link Speed 2.5MHz 0x1 Link Speed 25MHz 0x2 Link Speed 125MHz	RO	0x0

Bit	Name	Description	Access	Reset
0	lnkmod	This bit indicates the current mode of operation of the link **Value** **Description** 0x0 Half Duplex 0x1 Full Duplex	RO	0x0

MMC_Control

The MMC Control register establishes the operating mode of the management counters. Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700100
emac1	0xFF702000	0xFF702100

Offset: 0x100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ucdbc RW 0x0	Reserved		cntprstlvl RW 0x0	cntprst RW 0x0	cntfreez RW 0x0	rstonrd RW 0x0	cntstopro RW 0x0	cntrst RW 0x0

MMC_Control Fields

Bit	Name	Description	Access	Reset
8	ucdbc	When set, this bit enables MAC to update all the related MMC Counters for Broadcast frames dropped due to setting of DBF bit (Disable Broadcast Frames) of MAC Filter Register at offset 0x0004. When reset, MMC Counters are not updated for dropped Broadcast frames.	RW	0x0

Bit	Name	Description	Access	Reset
5	`cntprstlvl`	When low and bit 4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0x7FFF_FFF0 (half - 16). When this bit is high and bit 4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0. **Value** **Description** 0x0 Preset All Counters to almost-half 0x1 Present All Counters almost-full	`RW`	`0x0`
4	`cntprst`	When this bit is set, all counters are initialized or preset to almost full or almost half according to bit 5. This bit is cleared automatically after 1 clock cycle. This bit, along with bit 5, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. **Value** **Description** 0x0 Counters not preset 0x1 Counters preset to full or almost full	`RW`	`0x0`
3	`cntfreez`	When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode. **Value** **Description** 0x0 Update MMC Counters 0x1 Freeze MMC counters to current value	`RW`	`0x0`

Bit	Name	Description	Access	Reset
2	`rstonrd`	When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read. **Value** **Description** 0x0 No reset after read 0x1 Reset after read	`RW`	`0x0`
1	`cntstopro`	When this bit is set, after reaching maximum value, the counter does not roll over to zero. **Value** **Description** 0x0 Counter Roll Over 0x1 Counter does not Roll Over	`RW`	`0x0`
0	`cntrst`	When this bit is set, all counters are reset. This bit is cleared automatically after one clock cycle. **Value** **Description** 0x0 Auto cleared after 1 clock cycle 0x1 All Counters Reset	`RW`	`0x0`

MMC_Receive_Interrupt

The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens: * Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter). * Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700104`
`emac1`	`0xFF702000`	`0xFF702104`

Offset: `0x104`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						rxctrlfis RO 0x0	rxrcverrfis RO 0x0	rxwdogfis RO 0x0	rxvlangbfis RO 0x0	rxfovfis RO 0x0	rxpausfis RO 0x0	rxorangefis RO 0x0	rxlenerfis RO 0x0	rxucgfis RO 0x0	rx1024tmaxoctgbfis RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rx512t1023octgbfis RO 0x0	rx256t511octgbfis RO 0x0	rx128t255octgbfis RO 0x0	rx65t127octgbfis RO 0x0	rx64octgbfis RO 0x0	rxosizegfis RO 0x0	rxusizegfis RO 0x0	rxjaberfis RO 0x0	rxruntfis RO 0x0	rxalgnerfis RO 0x0	rxcrcerfis RO 0x0	rxmcgfis RO 0x0	rxbcgfis RO 0x0	rxgoctis RO 0x0	rxgboctis RO 0x0	rxgbfrmis RO 0x0

MMC_Receive_Interrupt Fields

Bit	Name	Description	Access	Reset
25	rxctrlfis	This bit is set when the rxctrlframes_g counter reaches half of the maximum value or the maximum value.	RO	0x0
24	rxrcverrfis	This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value.	RO	0x0
23	rxwdogfis	This bit is set when the rxwatchdogerror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxwatchdogerror < half max 0x1 rxwatchdogerror >= half max	RO	0x0
22	rxvlangbfis	This bit is set when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxvlanframes_gb < half max 0x1 rxvlanframes_gb >= half max	RO	0x0

Bit	Name	Description	Access	Reset
21	`rxfovfis`	This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxfifooverflow < half max 0x1 rxfifooverflow >= half max	RO	0x0
20	`rxpausfis`	This bit is set when the rxpauseframe counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxpauseframe < half max 0x1 rxpauseframe >= half max	RO	0x0
19	`rxorangefis`	This bit is set when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxoutofrangetype < half max 0x1 rxoutofrangetype >= half max	RO	0x0
18	`rxlenerfis`	This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxlengtherror < half max 0x1 rxlengtherror >= half max	RO	0x0
17	`rxucgfis`	This bit is set when the rxunicastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx1024tomaxoctets_gb < half max 0x1 rx1024tomaxoctets_gb >= half max	RO	0x0

Bit	Name	Description	Access	Reset
16	rx1024tmaxoctgbfis	This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx1024tomaxoctets_gb < half max 0x1 rx1024tomaxoctets_gb >= half max	RO	0x0
15	rx512t1023octgbfis	This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx512to1023octets_gb < half max 0x1 rx512to1023octets_gb >= half max	RO	0x0
14	rx256t511octgbfis	This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx256to511octets_gb < half max 0x1 rx256to511octets_gb >= half max	RO	0x0
13	rx128t255octgbfis	This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx128to255octets_gb < half max 0x1 rx128to255octets_gb >= half max	RO	0x0
12	rx65t127octgbfis	This is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx65to127octets_gb < half max 0x1 rx65to127octets_gb >= half max	RO	0x0

Bit	Name	Description	Access	Reset
11	`rx64octgbfis`	This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rx64octets_gb < half max 0x1 rx64octets_gb >= half max	RO	0x0
10	`rxosizegfis`	This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxoversize_g < half max 0x1 rxoversize_g >= half max	RO	0x0
9	`rxusizegfis`	This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxundersize_g < half max 0x1 rxundersize_g >= half max	RO	0x0
8	`rxjaberfis`	This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxjabbererror < half max 0x1 rxjabbererror >= half max	RO	0x0
7	`rxruntfis`	This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxrunterror < half max 0x1 rxrunterror >= half max	RO	0x0

Bit	Name	Description	Access	Reset
6	`rxalgnerfis`	This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxalignmenterror < half max 0x1 rxalignmenterror >= half max	RO	0x0
5	`rxcrcerfis`	This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxcrcerror < half max 0x1 rxcrcerror >= half max	RO	0x0
4	`rxmcgfis`	This bit is set when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 rxbroadcastframes_g < half max 0x1 rxbroadcastframes_g >= half max	RO	0x0
3	`rxbcgfis`	This bit is set when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value.	RO	0x0
2	`rxgoctis`	This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 Rxoctetcount_g < half max 0x1 Rxoctetcount_g >= half max	RO	0x0
1	`rxgboctis`	This bit is set when the rxoctetcount_bg counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 Rxoctetcount_bg < half max 0x1 Rxoctetcount_bg >= half max	RO	0x0

Bit	Name	Description	Access	Reset
0	rxgbfrmis	This bit is set when the rxframecount_bg counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 Preset All Counters to almost-half 0x1 Present All Counters almost-full	RO	0x0

MMC_Transmit_Interrupt

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700108
emac1	0xFF702000	0xFF702108

Offset: 0x108

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						txosizegfis RO 0x0	txvlangfis RO 0x0	txpausfis RO 0x0	txexdeffis RO 0x0	txgfrmis RO 0x0	txgoctis RO 0x0	txcarerfis RO 0x0	txexcolfis RO 0x0	txlatcolfis RO 0x0	txdeffis RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
txmcolgfis RO 0x0	txscolgfis RO 0x0	txuflowerfis RO 0x0	txbcgbfis RO 0x0	txmcgbfis RO 0x0	txucgbfis RO 0x0	tx1024tmaxoctgbfis RO 0x0	tx512t1023octgbfis RO 0x0	tx256t511octgbfis RO 0x0	tx128t255octgbfis RO 0x0	tx65t127octgbfis RO 0x0	tx64octgbfis RO 0x0	txmcgfis RO 0x0	txbcgfis RO 0x0	txgbfrmis RO 0x0	txgboctis RO 0x0

MMC_Transmit_Interrupt Fields

Bit	Name	Description	Access	Reset
25	txosizegfis	This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value.	RO	0x0
24	txvlangfis	This bit is set when the txvlanframes_g counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 — txvlanframes_g counter < half max 0x1 — txvlanframes_g counter >= half max	RO	0x0
23	txpausfis	This bit is set when the txpauseframeserror counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 — txpauseframeserror counter < half max 0x1 — txpauseframeserror counter >= half max	RO	0x0
22	txexdeffis	This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 — txoexcessdef counter < half max 0x1 — txoexcessdef counter >= half max	RO	0x0
21	txgfrmis	This bit is set when the txframecount_g counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 — txframecount_g counter < half max 0x1 — txframecount_g counter >= half max	RO	0x0
20	txgoctis	This bit is set when the txoctetcount_g counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 — txoctetcount_g counter < half max 0x1 — txoctetcount_g counter >= half max	RO	0x0

Bit	Name	Description	Access	Reset
19	txcarerfis	This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txcarriererror counter < half max 0x1 txcarriererror counter >= half max	RO	0x0
18	txexcolfis	This bit is set when the txexcesscol counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txexesscol counter < half max 0x1 txexesscol counter >= half max	RO	0x0
17	txlatcolfis	This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txlatecol counter < half max 0x1 txlatecol counter >= half max	RO	0x0
16	txdeffis	This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txdeferred counter < half max 0x1 txdeferred counter >= half max	RO	0x0
15	txmcolgfis	This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txmulticol_g counter < half max 0x1 txmulticol_g counter >= half max	RO	0x0
14	txscolgfis	This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txsinglecol_g counter < half max 0x1 txsinglecol_g counter >= half max	RO	0x0

Bit	Name	Description	Access	Reset
13	`txuflowerfis`	This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txunderflowerror counter < half max 0x1 txunderflowerror counter >= half max	RO	0x0
12	`txbcgbfis`	This bit is set when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txbroadcastframes_gb < half max 0x1 txbroadcastframes_gb >= half max	RO	0x0
11	`txmcgbfis`	This bit is set when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txmulticastframes_gb < half max 0x1 txmulticastframes_gb >= half max	RO	0x0
10	`txucgbfis`	This bit is set when the txunicastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 txunicastframes_bb < half max 0x1 txunicastframes_bb >= half max	RO	0x0
9	`tx1024tmaxoctgbfis`	This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx1024tomaxoctets_gb < half max 0x1 tx1024tomaxoctets_gb >= half max	RO	0x0

Bit	Name	Description	Access	Reset
8	tx512t1023octgbfis	This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx512to1023octets_gb < half max 0x1 tx512to1023octets_gb >= half max	RO	0x0
7	tx256t511octgbfis	This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx256to511octets_gb < half max 0x1 tx256to511octets_gb >= half max	RO	0x0
6	tx128t255octgbfis	This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx128to255octets_gb < half max 0x1 tx128to255octets_gb >= half max	RO	0x0
5	tx65t127octgbfis	This bit is set when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx65to127octets_gb < half max 0x1 tx65to127octets_gb >= half max	RO	0x0
4	tx64octgbfis	This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 tx64octets_gb < half max 0x1 tx64octets_gb >= half max	RO	0x0

Bit	Name	Description	Access	Reset
3	`txmcgfis`	This bit is set when the txmulticastframes_g counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 txmulticastframes_g < half max 0x1 txmulticastframes_g >= half max	RO	0x0
2	`txbcgfis`	This bit is set when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 txbroadcastframes_g < half max 0x1 txbroadcastframes_g >= half max	RO	0x0
1	`txgbfrmis`	This bit is set when the txframecount_gb counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 txframecount_gb < half max 0x1 txframecount_gb >= half max	RO	0x0
0	`txgboctis`	This bit is set when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. **Value** / **Description** 0x0 txoctetcount_gb < half max 0x1 txoctetcount_gb >= half max	RO	0x0

MMC_Receive_Interrupt_Mask

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when the receive statistic counters reach half of their maximum value, or maximum value. This register is 32-bits wide.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70010C
emac1	0xFF702000	0xFF70210C

Offset: `0x10C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						rxctrlfim RW 0x0	rxrcverrfim RW 0x0	rxwdogfim RW 0x0	rxvlangbfim RW 0x0	rxfovfim RW 0x0	rxpausfim RW 0x0	rxorangefim RW 0x0	rxlenerfim RW 0x0	rxucgfim RW 0x0	rx1024tmaxoctgbfim RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rx512t1023octgbfim RW 0x0	rx256t511octgbfim RW 0x0	rx128t255octgbfim RW 0x0	rx65t127octgbfim RW 0x0	rx64octgbfim RW 0x0	rxosizegfim RW 0x0	rxusizegfim RW 0x0	rxjaberfim RW 0x0	rxruntfim RW 0x0	rxalgnerfim RW 0x0	rxcrcerfim RW 0x0	rxmcgfim RW 0x0	rxbcgfim RW 0x0	rxgoctim RW 0x0	rxgboctim RW 0x0	rxgbfrmim RW 0x0

MMC_Receive_Interrupt_Mask Fields

Bit	Name	Description	Access	Reset
25	rxctrlfim	Setting this bit masks the interrupt when the rxctrlframes counter reaches half the maximum value, and also when it reaches the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
24	rxrcverrfim	Setting this bit masks the interrupt when the rxrcverror error counter reaches half the maximum value, and also when it reaches the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
23	`rxwdogfim`	Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
22	`rxvlangbfim`	Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
21	`rxfovfim`	Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
20	`rxpausfim`	Setting this bit masks the interrupt when the rxpauseframes counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
19	`rxorangefim`	Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
18	`rxlenerfim`	Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
17	`rxucgfim`	Setting this bit masks the interrupt when the rxunicastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
16	`rx1024tmaxoctgbfim`	Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
15	`rx512t1023octgbfim`	Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
14	`rx256t511octgbfim`	Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`

Bit	Name	Description	Access	Reset
13	rx128t255octgbfim	Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
12	rx65t127octgbfim	Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
11	rx64octgbfim	Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
10	rxosizegfim	Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
9	rxusizegfim	Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
8	rxjaberfim	Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
7	rxruntfim	Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
6	rxalgnerfim	Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
5	rxcrcerfim	Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
4	rxmcgfim	Setting this bit masks the interrupt when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
3	rxbcgfim	Setting this bit masks the interrupt when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
2	rxgoctim	Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
1	rxgboctim	Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
0	rxgbfrmim	Setting this bit masks the interrupt when the rxframecount_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

MMC_Transmit_Interrupt_Mask

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700110
emac1	0xFF702000	0xFF702110

Offset: 0x110

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						txosizegfim RW 0x0	txvlangfim RW 0x0	txpausfim RW 0x0	txexdeffim RW 0x0	txgfrmim RW 0x0	txgoctim RW 0x0	txcarerfim RW 0x0	txexcolfim RW 0x0	txlatcolfim RW 0x0	txdeffim RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
txmcolgfim RW 0x0	txscolgfim RW 0x0	txuflowerfim RW 0x0	txbcgbfim RW 0x0	txmcgbfim RW 0x0	txucgbfim RW 0x0	tx1024tmaxoctgbfim RW 0x0	tx512t1023octgbfim RW 0x0	tx256t511octgbfim RW 0x0	tx128t255octgbfim RW 0x0	tx65t127octgbfim RW 0x0	tx64octgbfim RW 0x0	txmcgfim RW 0x0	txbcgfim RW 0x0	txgbfrmim RW 0x0	txgboctim RW 0x0

MMC_Transmit_Interrupt_Mask Fields

Bit	Name	Description	Access	Reset
25	txosizegfim	Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
24	txvlangfim	Setting this bit masks the interrupt when the txvlanframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
23	txpausfim	Setting this bit masks the interrupt when the txpauseframes counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
22	`txexdeffim`	Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
21	`txgfrmim`	Setting this bit masks the interrupt when the txframecount_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
20	`txgoctim`	Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
19	`txcarerfim`	Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
18	`txexcolfim`	Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	`txlatcolfim`	Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
16	`txdeffim`	Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
15	`txmcolgfim`	Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
14	`txscolgfim`	Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
13	`txuflowerfim`	Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
12	txbcgbfim	Setting this bit masks the interrupt when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
11	txmcgbfim	Setting this bit masks the interrupt when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
10	txucgbfim	Setting this bit masks the interrupt when the txunicastframes_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
9	tx1024tmaxoctgbfim	Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
8	tx512t1023octgbfim	Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
7	tx256t511octgbfim	Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
6	tx128t255octgbfim	Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
5	tx65t127octgbfim	Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
4	tx64octgbfim	Setting this bit masks the interrupt when the tx64octets_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
3	txmcgfim	Setting this bit masks the interrupt when the txmulticastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
2	`txbcgfim`	Setting this bit masks the interrupt when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
1	`txgbfrmim`	Setting this bit masks the interrupt when the txframecount_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`
0	`txgboctim`	Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RW`	`0x0`

txoctetcount_gb

Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700114`
`emac1`	`0xFF702000`	`0xFF702114`

Offset: `0x114`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`cnt` `RO 0x0`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`cnt` `RO 0x0`															

txoctetcount_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames	`RO`	`0x0`

txframecount_gb

Number of good and bad frames transmitted, exclusive of retried frames

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700118`
`emac1`	`0xFF702000`	`0xFF702118`

Offset: `0x118`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txframecount_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good and bad frames transmitted, exclusive of retried frames	`RO`	`0x0`

txbroadcastframes_g

Number of good broadcast frames transmitted

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70011C`
`emac1`	`0xFF702000`	`0xFF70211C`

Offset: `0x11C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txbroadcastframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good broadcast frames transmitted	RO	0x0

txmulticastframes_g

Number of good multicast frames transmitted

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700120
emac1	0xFF702000	0xFF702120

Offset: 0x120

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txmulticastframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good multicast frames transmitted	RO	0x0

tx64octets_gb

Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700124
emac1	0xFF702000	0xFF702124

Offset: 0x124

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx64octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames	RO	0x0

tx65to127octets_gb

Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700128
emac1	0xFF702000	0xFF702128

Offset: 0x128

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx65to127octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

tx128to255octets_gb

Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70012C

Module Instance	Base Address	Register Address
emac1	0xFF702000	0xFF70212C

Offset: 0x12C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx128to255octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

tx256to511octets_gb

Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700130
emac1	0xFF702000	0xFF702130

Offset: 0x130

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx256to511octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

tx512to1023octets_gb

Number of good and bad frames transmitted with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700134
emac1	0xFF702000	0xFF702134

Offset: 0x134

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx512to1023octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

tx1024tomaxoctets_gb

Number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700138
emac1	0xFF702000	0xFF702138

Offset: 0x138

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

tx1024tomaxoctets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

txunicastframes_gb

Number of good and bad unicast frames transmitted

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70013C
emac1	0xFF702000	0xFF70213C

Offset: 0x13C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txunicastframes_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad unicast frames transmitted	RO	0x0

txmulticastframes_gb

Number of good and bad multicast frames transmitted

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700140
emac1	0xFF702000	0xFF702140

Offset: 0x140

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txmulticastframes_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good and bad multicast frames transmitted	`RO`	`0x0`

txbroadcastframes_gb

Number of good and bad broadcast frames transmitted

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700144`
`emac1`	`0xFF702000`	`0xFF702144`

Offset: `0x144`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txbroadcastframes_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good and bad broadcast frames transmitted	`RO`	`0x0`

txunderflowerror

Number of frames aborted due to frame underflow error

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700148`
`emac1`	`0xFF702000`	`0xFF702148`

Offset: `0x148`

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txunderflowerror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames aborted due to frame underflow error	RO	0x0

txsinglecol_g

Number of successfully transmitted frames after a single collision in Half-duplex mode

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70014C
emac1	0xFF702000	0xFF70214C

Offset: 0x14C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txsinglecol_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of successfully transmitted frames after a single collision in Half-duplex mode	RO	0x0

txmulticol_g

Number of successfully transmitted frames after more than a single collision in Half-duplex mode

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700150
emac1	0xFF702000	0xFF702150

Offset: `0x150`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txmulticol_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of successfully transmitted frames after more than a single collision in Half-duplex mode	RO	0x0

txdeferred

Number of successfully transmitted frames after a deferral in Halfduplex mode

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700154
emac1	0xFF702000	0xFF702154

Offset: `0x154`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txdeferred Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of successfully transmitted frames after a deferral in Halfduplex mode	RO	0x0

txlatecol

Number of frames aborted due to late collision error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700158

Module Instance	Base Address	Register Address
emac1	0xFF702000	0xFF702158

Offset: 0x158

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txlatecol Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames aborted due to late collision error	RO	0x0

txexesscol

Number of frames aborted due to excessive (16) collision errors

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70015C
emac1	0xFF702000	0xFF70215C

Offset: 0x15C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txexesscol Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames aborted due to excessive (16) collision errors	RO	0x0

txcarriererr

Number of frames aborted due to carrier sense error (no carrier or loss of carrier)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700160
emac1	0xFF702000	0xFF702160

Offset: 0x160

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txcarriererr Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames aborted due to carrier sense error (no carrier or loss of carrier)	RO	0x0

txoctetcnt

Number of bytes transmitted, exclusive of preamble, in good frames only

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700164
emac1	0xFF702000	0xFF702164

Offset: 0x164

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
txoctetcount_g RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
txoctetcount_g RO 0x0															

txoctetcnt Fields

Bit	Name	Description	Access	Reset
31:0	txoctetcount_g	Number of bytes transmitted, exclusive of preamble, in good frames only	RO	0x0

txframecount_g

Number of good frames transmitted

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700168
emac1	0xFF702000	0xFF702168

Offset: 0x168

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txframecount_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good frames transmitted	RO	0x0

txexcessdef

Number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70016C
emac1	0xFF702000	0xFF70216C

Offset: 0x16C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txexcessdef Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times)	RO	0x0

txpauseframes

Number of good PAUSE frames transmitted

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700170
emac1	0xFF702000	0xFF702170

Offset: 0x170

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txpauseframes Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good PAUSE frames transmitted	RO	0x0

txvlanframes_g

Number of good VLAN frames transmitted, exclusive of retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700174
emac1	0xFF702000	0xFF702174

Offset: 0x174

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txvlanframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good VLAN frames transmitted, exclusive of retried frames	RO	0x0

txoversize_g

Number of good and bad frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700178
emac1	0xFF702000	0xFF702178

Offset: 0x178

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

txoversize_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received	RO	0x0

rxframecount_gb

Number of good and bad frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700180
emac1	0xFF702000	0xFF702180

Offset: 0x180

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxframecount_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received	RO	0x0

rxoctetcount_gb

Number of bytes received, exclusive of preamble, in good and bad frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700184
emac1	0xFF702000	0xFF702184

Offset: 0x184

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxoctetcount_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received, exclusive of preamble, in good and bad frames	RO	0x0

rxoctetcount_g

Number of bytes received, exclusive of preamble, only in good frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700188
emac1	0xFF702000	0xFF702188

Offset: 0x188

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxoctetcount_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received, exclusive of preamble, only in good frames	RO	0x0

rxbroadcastframes_g

Number of good broadcast frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70018C
emac1	0xFF702000	0xFF70218C

Offset: 0x18C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxbroadcastframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good broadcast frames received	RO	0x0

rxmulticastframes_g

Number of good multicast frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700190
emac1	0xFF702000	0xFF702190

Offset: 0x190

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxmulticastframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good multicast frames received	RO	0x0

rxcrcerror

Number of frames received with CRC error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700194
emac1	0xFF702000	0xFF702194

Offset: 0x194

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxcrcerror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with CRC error	RO	0x0

rxalignmenterror

Number of frames received with alignment (dribble) error. Valid only in 10/100 mode

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700198
emac1	0xFF702000	0xFF702198

Offset: 0x198

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxalignmenterror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with alignment (dribble) error. Valid only in 10/100 mode	RO	0x0

rxrunterror

Number of frames received with runt (<64 bytes and CRC error) error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70019C
emac1	0xFF702000	0xFF70219C

Offset: 0x19C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxrunterror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with runt (<64 bytes and CRC error) error	RO	0x0

rxjabbererror

Number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001A0
emac1	0xFF702000	0xFF7021A0

Offset: `0x1A0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxjabbererror Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames	RO	0x0

rxundersize_g

Number of frames received with length less than 64 bytes, without any errors

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7001A4`
`emac1`	`0xFF702000`	`0xFF7021A4`

Offset: `0x1A4`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxundersize_g Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of frames received with length less than 64 bytes, without any errors	RO	0x0

rxoversize_g

Number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames), without errors

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001A8
emac1	0xFF702000	0xFF7021A8

Offset: 0x1A8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxoversize_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames), without errors	RO	0x0

rx64octets_gb

Number of good and bad frames received with length 64 bytes, exclusive of preamble

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001AC
emac1	0xFF702000	0xFF7021AC

Offset: 0x1AC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx64octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received with length 64 bytes, exclusive of preamble	RO	0x0

rx65to127octets_gb

Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001B0
emac1	0xFF702000	0xFF7021B0

Offset: 0x1B0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx65to127octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble	RO	0x0

rx128to255octets_gb

Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001B4
emac1	0xFF702000	0xFF7021B4

Offset: 0x1B4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx128to255octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble	`RO`	`0x0`

rx256to511octets_gb

Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7001B8`
`emac1`	`0xFF702000`	`0xFF7021B8`

Offset: `0x1B8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx256to511octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble	`RO`	`0x0`

rx512to1023octets_gb

Number of good and bad frames received with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7001BC`

Module Instance	Base Address	Register Address
emac1	0xFF702000	0xFF7021BC

Offset: 0x1BC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx512to1023octets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble	RO	0x0

rx1024tomaxoctets_gb

Number of good and bad frames received with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001C0
emac1	0xFF702000	0xFF7021C0

Offset: 0x1C0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rx1024tomaxoctets_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad frames received with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames	RO	0x0

rxunicastframes_g

Number of good unicast frames received

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7001C4`
`emac1`	`0xFF702000`	`0xFF7021C4`

Offset: `0x1C4`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxunicastframes_g Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good unicast frames received	`RO`	`0x0`

rxlengtherror

Number of frames received with length error (length type field not equal to frame size), for all frames with valid length field

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7001C8`
`emac1`	`0xFF702000`	`0xFF7021C8`

Offset: `0x1C8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxlengtherror Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of frames received with length error (length type field not equal to frame size), for all frames with valid length field	RO	0x0

rxoutofrangetype

Number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001CC
emac1	0xFF702000	0xFF7021CC

Offset: `0x1CC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxoutofrangetype Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536)	RO	0x0

rxpauseframes

Number of good and valid PAUSE frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001D0
emac1	0xFF702000	0xFF7021D0

Offset: `0x1D0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxpauseframes Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and valid PAUSE frames received	RO	0x0

rxfifooverflow

Number of missed received frames due to FIFO overflow

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001D4
emac1	0xFF702000	0xFF7021D4

Offset: 0x1D4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxfifooverflow Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of missed received frames due to FIFO overflow	RO	0x0

rxvlanframes_gb

Number of good and bad VLAN frames received

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001D8
emac1	0xFF702000	0xFF7021D8

Offset: 0x1D8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxvlanframes_gb Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good and bad VLAN frames received	RO	0x0

rxwatchdogerror

Number of frames received with error due to watchdog timeout error (frames with a data load larger than 2,048 bytes)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001DC
emac1	0xFF702000	0xFF7021DC

Offset: 0x1DC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxwatchdogerror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with error due to watchdog timeout error (frames with a data load larger than 2,048 bytes)	RO	0x0

rxrcverror

Number of frames received with Receive error or Frame Extension error on the GMII or MII interface.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001E0
emac1	0xFF702000	0xFF7021E0

Offset: 0x1E0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxrcverror Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of frames received with Receive error or Frame Extension error on the GMII or MII interface.	RO	0x0

rxctrlframes_g

Number of received good control frames.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7001E4
emac1	0xFF702000	0xFF7021E4

Offset: 0x1E4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxctrlframes_g Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of received good control frames.	RO	0x0

MMC_IPC_Receive_Interrupt_Mask

This register maintains the mask for the interrupt generated from the receive IPC statistic counters.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700200
emac1	0xFF702000	0xFF702200

Offset: `0x200`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		rxicmperoim RW 0x0	rxicmpgoim RW 0x0	rxtcperoim RW 0x0	rxtcpgoim RW 0x0	rxudperoim RW 0x0	rxudpgoim RW 0x0	rxipv6nopayoim RW 0x0	rxipv6heroim RW 0x0	rxipv6goim RW 0x0	rxipv4udsbloim RW 0x0	rxipv4fragoim RW 0x0	rxipv4nopayoim RW 0x0	rxipv4heroim RW 0x0	rxipv4goim RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		rxicmperfim RW 0x0	rxicmpgfim RW 0x0	rxtcperfim RW 0x0	rxtcpgfim RW 0x0	rxudperfim RW 0x0	rxudpgfim RW 0x0	rxipv6nopayfim RW 0x0	rxipv6herfim RW 0x0	rxipv6gfim RW 0x0	rxipv4udsblfim RW 0x0	rxipv4fragfim RW 0x0	rxipv4nopayfim RW 0x0	rxipv4herfim RW 0x0	rxipv4gfim RW 0x0

MMC_IPC_Receive_Interrupt_Mask Fields

Bit	Name	Description	Access	Reset
29	rxicmperoim	Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
28	rxicmpgoim	Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
27	`rxtcperoim`	Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
26	`rxtcpgoim`	Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
25	`rxudperoim`	Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
24	`rxudpgoim`	Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
23	`rxipv6nopayoim`	Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
22	`rxipv6heroim`	Setting this bit masks interrupt when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
21	`rxipv6goim`	Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
20	`rxipv4udsbloim`	Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
19	`rxipv4fragoim`	Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
18	`rxipv4nopayoim`	Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
17	`rxipv4heroim`	Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
16	`rxipv4goim`	Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
13	`rxicmperfim`	Setting this bit masks the interrupt when the rxicmp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
12	`rxicmpgfim`	Setting this bit masks the interrupt when the rxicmp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
11	`rxtcperfim`	Setting this bit masks the interrupt when the rxtcp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
10	rxtcpgfim	Setting this bit masks the interrupt when the rxtcp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
9	rxudperfim	Setting this bit masks the interrupt when the rxudp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
8	rxudpgfim	Setting this bit masks the interrupt when the rxudp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
7	rxipv6nopayfim	Setting this bit masks the interrupt when the rxipv6_nopay_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
6	rxipv6herfim	Setting this bit masks the interrupt when the rxipv6_hdrerr_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
5	rxipv6gfim	Setting this bit masks the interrupt when the rxipv6_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
4	rxipv4udsblfim	Setting this bit masks the interrupt when the rxipv4_udsbl_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
3	rxipv4fragfim	Setting this bit masks the interrupt when the rxipv4_frag_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
2	rxipv4nopayfim	Setting this bit masks the interrupt when the rxipv4_nopay_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0
1	rxipv4herfim	Setting this bit masks the interrupt when the rxipv4_hdrerr_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

Bit	Name	Description	Access	Reset
0	rxipv4gfim	Setting this bit masks the interrupt when the rxipv4_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RW	0x0

MMC_IPC_Receive_Interrupt

This register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Checksum Offload Interrupt register is 32-bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (bits[7:0]) must be read to clear the interrupt bit.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700208
emac1	0xFF702000	0xFF702208

Offset: 0x208

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		rxicmperois RO 0x0	rxicmpgois RO 0x0	rxtcperois RO 0x0	rxtcpgois RO 0x0	rxudperois RO 0x0	rxudpgois RO 0x0	rxipv6nopayois RO 0x0	rxipv6herois RO 0x0	rxipv6gois RO 0x0	rxipv4udsblois RO 0x0	rxipv4fragois RO 0x0	rxipv4nopayois RO 0x0	rxipv4herois RO 0x0	rxipv4gois RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		rxicmperfis RO 0x0	rxicmpgfis RO 0x0	rxtcperfis RO 0x0	rxtcpgfis RO 0x0	rxudperfis RO 0x0	rxudpgfis RO 0x0	rxipv6nopayfis RO 0x0	rxipv6herfis RO 0x0	rxipv6gfis RO 0x0	rxipv4udsblfis RO 0x0	rxipv4fragfis RO 0x0	rxipv4nopayfis RO 0x0	rxipv4herfis RO 0x0	rxipv4gfis RO 0x0

MMC_IPC_Receive_Interrupt Fields

Bit	Name	Description	Access	Reset
29	rxicmperois	This bit is set when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
28	rxicmpgois	This bit is set when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
27	rxtcperois	This bit is set when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
26	rxtcpgois	This bit is set when the rxtcp_gd_octets counter reaches half the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
25	rxudperois	This bit is set when the rxudp_err_octets counter reaches half the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0

Bit	Name	Description	Access	Reset
24	`rxudpgois`	This bit is set when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
23	`rxipv6nopayois`	This bit is set when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
22	`rxipv6herois`	This bit is set when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
21	`rxipv6gois`	This bit is set when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
20	`rxipv4udsblois`	This bit is set when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0

Bit	Name	Description	Access	Reset
19	`rxipv4fragois`	This bit is set when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
18	`rxipv4nopayois`	This bit is set when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
17	`rxipv4herois`	This bit is set when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
16	`rxipv4gois`	This bit is set when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
13	`rxicmperfis`	This bit is set when the rxicmp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0

Bit	Name	Description	Access	Reset
12	`rxicmpgfis`	This bit is set when the rxicmp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RO`	`0x0`
11	`rxtcperfis`	This bit is set when the rxtcp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RO`	`0x0`
10	`rxtcpgfis`	This bit is set when the rxtcp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RO`	`0x0`
9	`rxudperfis`	This bit is set when the rxudp_err_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RO`	`0x0`
8	`rxudpgfis`	This bit is set when the rxudp_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	`RO`	`0x0`

Bit	Name	Description	Access	Reset
7	`rxipv6nopayfis`	This bit is set when the rxipv6_nopay_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
6	`rxipv6herfis`	This bit is set when the rxipv6_hdrerr_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
5	`rxipv6gfis`	This bit is set when the rxipv6_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
4	`rxipv4udsblfis`	This bit is set when the rxipv4_udsbl_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
3	`rxipv4fragfis`	This bit is set when the rxipv4_frag_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0

Bit	Name	Description	Access	Reset
2	rxipv4nopayfis	This bit is set when the rxipv4_nopay_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
1	rxipv4herfis	This bit is set when the rxipv4_hdrerr_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0
0	rxipv4gfis	This bit is set when the rxipv4_gd_frms counter reaches half of the maximum value or the maximum value. **Value** **Description** 0x0 counter < half max 0x1 counter >= half max or max	RO	0x0

rxipv4_gd_frms

Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700210
emac1	0xFF702000	0xFF702210

Offset: 0x210

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_gd_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload	RO	0x0

rxipv4_hdrerr_frms

Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700214
emac1	0xFF702000	0xFF702214

Offset: 0x214

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_hdrerr_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors	RO	0x0

rxipv4_nopay_frms

Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700218
emac1	0xFF702000	0xFF702218

Offset: 0x218

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_nopay_frms Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine	`RO`	`0x0`

rxipv4_frag_frms

Number of good IPv4 datagrams with fragmentation

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70021C`
`emac1`	`0xFF702000`	`0xFF70221C`

Offset: `0x21C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_frag_frms Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good IPv4 datagrams with fragmentation	`RO`	`0x0`

rxipv4_udsbl_frms

Number of good IPv4 datagrams received that had a UDP payload with checksum disabled

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700220`
`emac1`	`0xFF702000`	`0xFF702220`

Offset: `0x220`

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_udsbl_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IPv4 datagrams received that had a UDP payload with checksum disabled	RO	0x0

rxipv6_gd_frms

Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700224
emac1	0xFF702000	0xFF702224

Offset: 0x224

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_gd_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads	RO	0x0

rxipv6_hdrerr_frms

Number of IPv6 datagrams received with header errors (length or version mismatch)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700228
emac1	0xFF702000	0xFF702228

Offset: `0x228`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_hdrerr_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of IPv6 datagrams received with header errors (length or version mismatch)	RO	0x0

rxipv6_nopay_frms

Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70022C
emac1	0xFF702000	0xFF70222C

Offset: `0x22C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_nopay_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers	RO	0x0

rxudp_gd_frms

Number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700230
emac1	0xFF702000	0xFF702230

Offset: 0x230

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxudp_gd_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented	RO	0x0

rxudp_err_frms

Number of good IP datagrams whose UDP payload has a checksum error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700234
emac1	0xFF702000	0xFF702234

Offset: 0x234

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxudp_err_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IP datagrams whose UDP payload has a checksum error	RO	0x0

rxtcp_gd_frms

Number of good IP datagrams with a good TCP payload

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700238
emac1	0xFF702000	0xFF702238

Offset: 0x238

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxtcp_gd_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IP datagrams with a good TCP payload	RO	0x0

rxtcp_err_frms

Number of good IP datagrams whose TCP payload has a checksum error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70023C
emac1	0xFF702000	0xFF70223C

Offset: 0x23C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxtcp_err_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IP datagrams whose TCP payload has a checksum error	RO	0x0

rxicmp_gd_frms

Number of good IP datagrams with a good ICMP payload

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700240
emac1	0xFF702000	0xFF702240

Offset: 0x240

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxicmp_gd_frms Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of good IP datagrams with a good ICMP payload	RO	0x0

rxicmp_err_frms

Number of good IP datagrams whose ICMP payload has a checksum error

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700244
emac1	0xFF702000	0xFF702244

Offset: 0x244

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxicmp_err_frms Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of good IP datagrams whose ICMP payload has a checksum error	`RO`	`0x0`

rxipv4_gd_octets

Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700250`
`emac1`	`0xFF702000`	`0xFF702250`

Offset: `0x250`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_gd_octets Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data	`RO`	`0x0`

rxipv4_hdrerr_octets

Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700254`
`emac1`	`0xFF702000`	`0xFF702254`

Offset: `0x254`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_hdrerr_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter	RO	0x0

rxipv4_nopay_octets

Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700258
emac1	0xFF702000	0xFF702258

Offset: `0x258`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_nopay_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter	RO	0x0

rxipv4_frag_octets

Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70025C
emac1	0xFF702000	0xFF70225C

Offset: 0x25C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_frag_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter	RO	0x0

rxipv4_udsbl_octets

Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700260
emac1	0xFF702000	0xFF702260

Offset: 0x260

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv4_udsbl_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes	RO	0x0

rxipv6_gd_octets

Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700264
emac1	0xFF702000	0xFF702264

Offset: 0x264

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_gd_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data	RO	0x0

rxipv6_hdrerr_octets

Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 headers Length field is used to update this counter

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700268
emac1	0xFF702000	0xFF702268

Offset: 0x268

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_hdrerr_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 headers Length field is used to update this counter	RO	0x0

rxipv6_nopay_octets

Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70026C
emac1	0xFF702000	0xFF70226C

Offset: 0x26C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxipv6_nopay_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter	RO	0x0

rxudp_gd_octets

Number of bytes received in a good UDP segment. This counter does not count IP header bytes

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700270
emac1	0xFF702000	0xFF702270

Offset: 0x270

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxudp_gd_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in a good UDP segment. This counter does not count IP header bytes	RO	0x0

rxudp_err_octets

Number of bytes received in a UDP segment that had checksum errors

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700274
emac1	0xFF702000	0xFF702274

Offset: 0x274

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxudp_err_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in a UDP segment that had checksum errors	RO	0x0

rxtcp_gd_octets

Number of bytes received in a good TCP segment

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700278`
`emac1`	`0xFF702000`	`0xFF702278`

Offset: `0x278`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`cnt` `RO 0x0`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`cnt` `RO 0x0`															

rxtcp_gd_octets Fields

Bit	Name	Description	Access	Reset
31:0	`cnt`	Number of bytes received in a good TCP segment	`RO`	`0x0`

rxtcperroctets

Number of bytes received in a TCP segment with checksum errors

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70027C`
`emac1`	`0xFF702000`	`0xFF70227C`

Offset: `0x27C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`rxtcp_err_octets` `RO 0x0`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`rxtcp_err_octets` `RO 0x0`															

rxtcperroctets Fields

Bit	Name	Description	Access	Reset
31:0	`rxtcp_err_octets`	Number of bytes received in a TCP segment with checksum errors	`RO`	`0x0`

rxicmp_gd_octets

Number of bytes received in a good ICMP segment

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700280
emac1	0xFF702000	0xFF702280

Offset: 0x280

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxicmp_gd_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in a good ICMP segment	RO	0x0

rxicmp_err_octets

Number of bytes received in an ICMP segment with checksum errors

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700284
emac1	0xFF702000	0xFF702284

Offset: 0x284

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cnt RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cnt RO 0x0															

rxicmp_err_octets Fields

Bit	Name	Description	Access	Reset
31:0	cnt	Number of bytes received in an ICMP segment with checksum errors	RO	0x0

L3_L4_Control0

This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700400
emac1	0xFF702000	0xFF702400

Offset: 0x400

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="10">Reserved</td><td>l4dpim0
RW
0x0</td><td>l4dpm0
RW
0x0</td><td>l4spim0
RW
0x0</td><td>l4spm0
RW
0x0</td><td>Reserved</td><td>l4pen0
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">l3hdbm0
RW 0x0</td><td colspan="5">l3hsbm0
RW 0x0</td><td>l3daim0
RW
0x0</td><td>l3dam0
RW
0x0</td><td>l3saim0
RW
0x0</td><td>l3sam0
RW
0x0</td><td>Reserved</td><td>l3pen0
RW 0x0</td></tr>
</table>

L3_L4_Control0 Fields

Bit	Name	Description	Access	Reset
21	l4dpim0	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 20 (L4DPM0) is set high.	RW	0x0
20	l4dpm0	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Destination Port number field for matching.	RW	0x0
19	l4spim0	When set, this bit indicates that the Layer 4 Source Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 18 (L4SPM0) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
18	l4spm0	When set, this bit indicates that the Layer 4 Source Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Source Port number field for matching.	RW	0x0
16	l4pen0	When set, this bit indicates that the Source and Destination Port number fields for UDP frames are used for matching. When reset, this bit indicates that the Source and Destination Port number fields for TCP frames are used for matching. The Layer 4 matching is done only when either L4SPM0 or L4DPM0 bit is set high.	RW	0x0
15:11	l3hdbm0	IPv4 Frames: This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: Bits [12:11] of this field correspond to Bits [6:5] of L3HSBM0, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 frames. The following list describes the concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 127: All bits except MSb are masked. This field is valid and applicable only if L3DAM0 or L3SAM0 is set high.	RW	0x0
10:6	l3hsbm0	IPv4 Frames: This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: This field contains Bits [4:0] of the field that indicates the number of higher bits of IP Source or Destination Address matched in the IPv6 frames. This field is valid and applicable only if L3DAM0 or L3SAM0 is set high.	RW	0x0
5	l3daim0	When set, this bit indicates that the Layer 3 IP Destination Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 4 (L3DAM0) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
4	`l3dam0`	When set, this bit indicates that Layer 3 IP Destination Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When Bit 0 (L3PEN0) is set, you should set either this bit or Bit 2 (L3SAM0) because either IPv6 DA or SA can be checked for filtering.	RW	0x0
3	`l3saim0`	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 2 (L3SAM0) is set high.	RW	0x0
2	`l3sam0`	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When Bit 0 (L3PEN0) is set, you should set either this bit or Bit 4 (L3DAM0) because either IPv6 SA or DA can be checked for filtering.	RW	0x0
0	`l3pen0`	When set, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv6 frames. When reset, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv4 frames. The Layer 3 matching is done only when either L3SAM0 or L3DAM0 bit is set high.	RW	0x0

Layer4_Address0

Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700404`
`emac1`	`0xFF702000`	`0xFF702404`

Offset: `0x404`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l4dp0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l4sp0 RW 0x0															

Layer4_Address0 Fields

Bit	Name	Description	Access	Reset
31:16	l4dp0	When Bit 16 (L4PEN0) is reset and Bit 20 (L4DPM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN0) and Bit 20 (L4DPM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 frames.	RW	0x0
15:0	l4sp0	Layer 4 Source Port Number Field When Bit 16 (L4PEN0) is reset and Bit 20 (L4DPM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN0) and Bit 20 (L4DPM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 frames.	RW	0x0

Layer3_Addr0_Reg0

For IPv4 frames, the Layer 3 Address 0 Register 0 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700410
emac1	0xFF702000	0xFF702410

Offset: 0x410

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a00 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a00 RW 0x0															

Layer3_Addr0_Reg0 Fields

Bit	Name	Description	Access	Reset
31:0	l3a00	When Bit 0 (L3PEN0) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN0) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN0) is reset and Bit 2 (L3SAM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the IP Source Address field in the IPv4 frames.	RW	0x0

Layer3_Addr1_Reg0

For IPv4 frames, the Layer 3 Address 1 Register 0 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700414
emac1	0xFF702000	0xFF702414

Offset: 0x414

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a10 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a10 RW 0x0															

Layer3_Addr1_Reg0 Fields

Bit	Name	Description	Access	Reset
31:0	l3a10	When Bit 0 (L3PEN0) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [63:32] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN0) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [63:32] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN0) is reset and Bit 4 (L3DAM0) is set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the IP Destination Address field in the IPv4 frames.	RW	0x0

Layer3_Addr2_Reg0

For IPv4 frames, the Layer 3 Address 2 Register 0 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700418
emac1	0xFF702000	0xFF702418

Offset: `0x418`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a20 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a20 RW 0x0															

Layer3_Addr2_Reg0 Fields

Bit	Name	Description	Access	Reset
31:0	l3a20	When Bit 0 (L3PEN0) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [95:64] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN0) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains value to be matched with Bits [95:64] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN0) is reset in Register 256 (Layer 3 and Layer 4 Control Register 0), this register is not used.	RW	0x0

Layer3_Addr3_Reg0

For IPv4 frames, the Layer 3 Address 3 Register 0 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70041C
emac1	0xFF702000	0xFF70241C

Offset: 0x41C

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a30 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a30 RW 0x0															

Layer3_Addr3_Reg0 Fields

Bit	Name	Description	Access	Reset
31:0	l3a30	When Bit 0 (L3PEN0) and Bit 2 (L3SAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [127:96] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN0) and Bit 4 (L3DAM0) are set in Register 256 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with Bits [127:96] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN0) is reset in Register 256 (Layer 3 and Layer 4 Control Register 0), this register is not used.	RW	0x0

L3_L4_Control1

This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700430
emac1	0xFF702000	0xFF702430

Offset: 0x430

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										l4dpim1 RW 0x0	l4dpm1 RW 0x0	l4spim1 RW 0x0	l4spm1 RW 0x0	Reserved	l4pen1 RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3hdbm1 RW 0x0					l3hsbm1 RW 0x0					l3daim1 RW 0x0	l3dam1 RW 0x0	l3saim1 RW 0x0	l3sam1 RW 0x0	Reserved	l3pen1 RW 0x0

L3_L4_Control1 Fields

Bit	Name	Description	Access	Reset
21	l4dpim1	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 20 (L4DPM1) is set high.	RW	0x0
20	l4dpm1	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Destination Port number field for matching.	RW	0x0
19	l4spim1	When set, this bit indicates that the Layer 4 Source Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 18 (L4SPM1) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
18	l4spm1	When set, this bit indicates that the Layer 4 Source Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Source Port number field for matching.	RW	0x0
16	l4pen1	When set, this bit indicates that the Source and Destination Port number fields for UDP frames are used for matching. When reset, this bit indicates that the Source and Destination Port number fields for TCP frames are used for matching. The Layer 4 matching is done only when either L4SPM1 or L4DPM1 bit is set high.	RW	0x0
15:11	l3hdbm1	IPv4 Frames: This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: Bits [12:11] of this field correspond to Bits [6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 frames. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 127: All bits except MSb are masked. This field is valid and applicable only if L3DAM1 or L3SAM1 is set high.	RW	0x0
10:6	l3hsbm1	IPv4 Frames: This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: This field contains Bits [4:0] of the field that indicates the number of higher bits of IP Source or Destination Address matched in the IPv6 frames. This field is valid and applicable only if L3DAM1 or L3SAM1 is set high.	RW	0x0
5	l3daim1	When set, this bit indicates that the Layer 3 IP Destination Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 4 (L3DAM1) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
4	`l3dam1`	When set, this bit indicates that Layer 3 IP Destination Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When Bit 1 (L3PEN1) is set, you should set either this bit or Bit 2 (L3SAM1) because either IPv6 DA or SA can be checked for filtering.	`RW`	`0x0`
3	`l3saim1`	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 2 (L3SAM1) is set high.	`RW`	`0x0`
2	`l3sam1`	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When Bit 0 (L3PEN1) is set, you should set either this bit or Bit 4 (L3DAM1) because either IPv6 SA or DA can be checked for filtering.	`RW`	`0x0`
0	`l3pen1`	When set, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv6 frames. When reset, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv4 frames. The Layer 3 matching is done only when either L3SAM1 or L3DAM1 bit is set high.	`RW`	`0x0`

Layer4_Address1

Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700434`
`emac1`	`0xFF702000`	`0xFF702434`

Offset: `0x434`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l4dp1 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l4sp1 RW 0x0															

Layer4_Address1 Fields

Bit	Name	Description	Access	Reset
31:16	l4dp1	When Bit 16 (L4PEN1) is reset and Bit 20 (L4DPM1) is set in Register 268 (Layer 3 and Layer 4 Control Register 0), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN1) and Bit 20 (L4DPM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 frames.	RW	0x0
15:0	l4sp1	When Bit 16 (L4PEN1) is reset and Bit 20 (L4DPM1) is set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN1) and Bit 20 (L4DPM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 frames.	RW	0x0

Layer3_Addr0_Reg1

For IPv4 frames, the Layer 3 Address 0 Register 1 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700440
emac1	0xFF702000	0xFF702440

Offset: 0x440

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a01 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a01 RW 0x0															

Layer3_Addr0_Reg1 Fields

Bit	Name	Description	Access	Reset
31:0	l3a01	When Bit 0 (L3PEN1) and Bit 2 (L3SAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN1) and Bit 4 (L3DAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN1) is reset and Bit 2 (L3SAM1) is set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with the IP Source Address field in the IPv4 frames.	RW	0x0

Layer3_Addr1_Reg1

For IPv4 frames, the Layer 3 Address 1 Register 1 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700444
emac1	0xFF702000	0xFF702444

Offset: 0x444

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">l3a11
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">l3a11
RW 0x0</td></tr>
</table>

Layer3_Addr1_Reg1 Fields

Bit	Name	Description	Access	Reset
31:0	l3a11	When Bit 0 (L3PEN1) and Bit 2 (L3SAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [63:32] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN1) and Bit 4 (L3DAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [63:32] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN1) is reset and Bit 4 (L3DAM1) is set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with the IP Destination Address field in the IPv4 frames.	RW	0x0

Layer3_Addr2_Reg1

For IPv4 frames, the Layer 3 Address 2 Register 1 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700448
emac1	0xFF702000	0xFF702448

Offset: 0x448

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a21 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a21 RW 0x0															

Layer3_Addr2_Reg1 Fields

Bit	Name	Description	Access	Reset
31:0	l3a21	When Bit 0 (L3PEN1) and Bit 2 (L3SAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [95:64] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN1) and Bit 4 (L3DAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains value to be matched with Bits [95:64] of the IP Destination Address field in the IPv6 frames.	RW	0x0

Layer3_Addr3_Reg1

For IPv4 frames, the Layer 3 Address 3 Register 1 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70044C
emac1	0xFF702000	0xFF70244C

Offset: `0x44C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a31 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a31 RW 0x0															

Layer3_Addr3_Reg1 Fields

Bit	Name	Description	Access	Reset
31:0	l3a31	When Bit 1 (L3PEN1) and Bit 2 (L3SAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [127:96] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN1) and Bit 4 (L3DAM1) are set in Register 268 (Layer 3 and Layer 4 Control Register 1), this field contains the value to be matched with Bits [127:96] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN1) is reset in Register 268 (Layer 3 and Layer 4 Control Register 1), this register is not used.	RW	0x0

L3_L4_Control2

This register controls the operations of the filter 2 of Layer 3 and Layer 4.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700460
emac1	0xFF702000	0xFF702460

Offset: `0x460`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										l4dpim2 RW 0x0	l4dpm2 RW 0x0	l4spim2 RW 0x0	l4spm2 RW 0x0	Reserved	l4pen2 RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3hdbm2 RW 0x0					l3hsbm2 RW 0x0					l3daim2 RW 0x0	l3dam2 RW 0x0	l3saim2 RW 0x0	l3sam2 RW 0x0	Reserved	l3pen2 RW 0x0

L3_L4_Control2 Fields

Bit	Name	Description	Access	Reset
21	l4dpim2	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 20 (L4DPM0) is set high.	RW	0x0
20	l4dpm2	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Destination Port number field for matching.	RW	0x0
19	l4spim2	When set, this bit indicates that the Layer 4 Source Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 18 (L4SPM2) is set high.	RW	0x0
18	l4spm2	When set, this bit indicates that the Layer 4 Source Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Source Port number field for matching.	RW	0x0
16	l4pen2	When set, this bit indicates that the Source and Destination Port number fields for UDP frames are used for matching. When reset, this bit indicates that the Source and Destination Port number fields for TCP frames are used for matching. The Layer 4 matching is done only when either L4SPM2 or L4DPM2 bit is set high.	RW	0x0

Bit	Name	Description	Access	Reset
15:11	l3hdbm2	IPv4 Frames: This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: Bits [12:11] of this field correspond to Bits [6:5] of L3HSBM2, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 frames. The following list describes the concatenated values of the L3HDBM2[1:0] and L3HSBM2 bits: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 127: All bits except MSb are masked. This field is valid and applicable only if L3DAM2 or L3SAM2 is set high.	RW	0x0
10:6	l3hsbm2	Layer 3 IP SA Higher Bits Match IPv4 Frames: This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: This field contains Bits [4:0] of the field that indicates the number of higher bits of IP Source or Destination Address matched in the IPv6 frames. This field is valid and applicable only if L3DAM2 or L3SAM2 is set high.	RW	0x0
5	l3daim2	When set, this bit indicates that the Layer 3 IP Destination Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 4 (L3DAM2) is set high.	RW	0x0
4	l3dam2	When set, this bit indicates that Layer 3 IP Destination Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When Bit 0 (L3PEN2) is set, you should set either this bit or Bit 2 (L3SAM2) because either IPv6 DA or SA can be checked for filtering.	RW	0x0
3	l3saim2	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 2 (L3SAM2) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
2	l3sam2	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When Bit 0 (L3PEN2) is set, you should set either this bit or Bit 4 (L3DAM2) because either IPv6 SA or DA can be checked for filtering.	RW	0x0
0	l3pen2	When set, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv6 frames. When reset, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv4 frames. The Layer 3 matching is done only when either L3SAM2 or L3DAM2 bit is set high.	RW	0x0

Layer4_Address2

Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700464
emac1	0xFF702000	0xFF702464

Offset: 0x464

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l4dp2 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l4sp2 RW 0x0															

Layer4_Address2 Fields

Bit	Name	Description	Access	Reset
31:16	l4dp2	When Bit 16 (L4PEN2) is reset and Bit 20 (L4DPM2) is set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN2) and Bit 20 (L4DPM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 frames.	RW	0x0
15:0	l4sp2	When Bit 16 (L4PEN2) is reset and Bit 20 (L4DPM2) is set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN2) and Bit 20 (L4DPM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 frames.	RW	0x0

Layer3_Addr0_Reg2

For IPv4 frames, the Layer 3 Address 0 Register 2 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700470
emac1	0xFF702000	0xFF702470

Offset: 0x470

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a02 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a02 RW 0x0															

Layer3_Addr0_Reg2 Fields

Bit	Name	Description	Access	Reset
31:0	l3a02	When Bit 0 (L3PEN2) and Bit 2 (L3SAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [31:0] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN2) and Bit 4 (L3DAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN2) is reset and Bit 2 (L3SAM2) is set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the IP Source Address field in the IPv4 frames.	RW	0x0

Layer3_Addr1_Reg2

For IPv4 frames, the Layer 3 Address 1 Register 2 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700474
emac1	0xFF702000	0xFF702474

Offset: 0x474

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a12 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a12 RW 0x0															

Layer3_Addr1_Reg2 Fields

Bit	Name	Description	Access	Reset
31:0	l3a12	Layer 3 Address 1 Field When Bit 0 (L3PEN2) and Bit 2 (L3SAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [63:32] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN2) and Bit 4 (L3DAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [63:32] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN2) is reset and Bit 4 (L3DAM2) is set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with the IP Destination Address field in the IPv4 frames.	RW	0x0

Layer3_Addr2_Reg2

For IPv4 frames, the Layer 3 Address 2 Register 2 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700478
emac1	0xFF702000	0xFF702478

Offset: 0x478

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a22 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a22 RW 0x0															

Layer3_Addr2_Reg2 Fields

Bit	Name	Description	Access	Reset
31:0	l3a22	When Bit 0 (L3PEN2) and Bit 2 (L3SAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [95:64] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN2) and Bit 4 (L3DAM2) are set in Register 256 (Layer 3 and Layer 4 Control Register 2), this field contains value to be matched with Bits [95:64] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN2) is reset in Register 280 (Layer 3 and Layer 4 Control Register 2), this register is not used.	RW	0x0

Layer3_Addr3_Reg2

For IPv4 frames, the Layer 3 Address 3 Register 2 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70047C
emac1	0xFF702000	0xFF70247C

Offset: 0x47C

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a32 RW 0x0															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
l3a32 RW 0x0															

Layer3_Addr3_Reg2 Fields

Bit	Name	Description	Access	Reset
31:0	l3a32	When Bit 0 (L3PEN2) and Bit 2 (L3SAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [127:96] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN2) and Bit 4 (L3DAM2) are set in Register 280 (Layer 3 and Layer 4 Control Register 2), this field contains the value to be matched with Bits [127:96] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN2) is reset in Register 280 (Layer 3 and Layer 4 Control Register 2), this register is not used.	RW	0x0

L3_L4_Control3

This register controls the operations of the filter 0 of Layer 3 and Layer 4.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700490
emac1	0xFF702000	0xFF702490

Offset: 0x490

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										l4dpim3 RW 0x0	l4dpm3 RW 0x0	l4spim3 RW 0x0	l4spm3 RW 0x0	Reserved	l4pen3 RW 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3hdbm3 RW 0x0					l3hsbm3 RW 0x0					l3daim3 RW 0x0	l3dam3 RW 0x0	l3saim3 RW 0x0	l3sam3 RW 0x0	Reserved	l3pen3 RW 0x0

L3_L4_Control3 Fields

Bit	Name	Description	Access	Reset
21	l4dpim3	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Destination Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 20 (L4DPM3) is set high.	RW	0x0
20	l4dpm3	When set, this bit indicates that the Layer 4 Destination Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Destination Port number field for matching.	RW	0x0
19	l4spim3	When set, this bit indicates that the Layer 4 Source Port number field is enabled for inverse matching. When reset, this bit indicates that the Layer 4 Source Port number field is enabled for perfect matching. This bit is valid and applicable only when Bit 18 (L4SPM3) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
18	l4spm3	When set, this bit indicates that the Layer 4 Source Port number field is enabled for matching. When reset, the MAC ignores the Layer 4 Source Port number field for matching.	RW	0x0
16	l4pen3	When set, this bit indicates that the Source and Destination Port number fields for UDP frames are used for matching. When reset, this bit indicates that the Source and Destination Port number fields for TCP frames are used for matching. The Layer 4 matching is done only when either L4SPM3 or L4DPM3 bit is set high.	RW	0x0
15:11	l3hdbm3	Layer 3 IP DA Higher Bits Match IPv4 Frames: This field contains the number of higher bits of IP Destination Address that are matched in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: Bits [12:11] of this field correspond to Bits [6:5] of L3HSBM3, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 frames. The following list describes the concatenated values of the L3HDBM3[1:0] and L3HSBM3 bits: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 127: All bits except MSb are masked. This field is valid and applicable only if L3DAM3 or L3SAM3 is set high.	RW	0x0
10:6	l3hsbm3	IPv4 Frames: This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 frames. The following list describes the values of this field: * 0: No bits are masked. * 1: LSb[0] is masked. * 2: Two LSbs [1:0] are masked. * ... * 31: All bits except MSb are masked. IPv6 Frames: This field contains Bits [4:0] of the field that indicates the number of higher bits of IP Source or Destination Address matched in the IPv6 frames. This field is valid and applicable only if L3DAM3 or L3SAM3 is set high.	RW	0x0
5	l3daim3	When set, this bit indicates that the Layer 3 IP Destination Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Destination Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 4 (L3DAM3) is set high.	RW	0x0

Bit	Name	Description	Access	Reset
4	l3dam3	When set, this bit indicates that Layer 3 IP Destination Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Destination Address field for matching. Note: When Bit 0 (L3PEN3) is set, you should set either this bit or Bit 2 (L3SAM3) because either IPv6 DA or SA can be checked for filtering.	RW	0x0
3	l3saim3	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for inverse matching. When reset, this bit indicates that the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when Bit 2 (L3SAM3) is set high.	RW	0x0
2	l3sam3	When set, this bit indicates that the Layer 3 IP Source Address field is enabled for matching. When reset, the MAC ignores the Layer 3 IP Source Address field for matching. Note: When Bit 0 (L3PEN3) is set, you should set either this bit or Bit 4 (L3DAM3) because either IPv6 SA or DA can be checked for filtering.	RW	0x0
0	l3pen3	When set, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv6 frames. When reset, this bit indicates that the Layer 3 IP Source or Destination Address matching is enabled for the IPv4 frames. The Layer 3 matching is done only when either L3SAM3 or L3DAM3 bit is set high.	RW	0x0

Layer4_Address3

Because the Layer 3 and Layer 4 Address Registers are double-synchronized to the Rx clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Layer 3 and Layer 4 Address Registers are written. For proper synchronization updates, you should perform the consecutive writes to the same Layer 3 and Layer 4 Address Registers after at least four clock cycles delay of the destination clock.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700494
emac1	0xFF702000	0xFF702494

Offset: 0x494

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l4dp3 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l4sp3 RW 0x0															

Layer4_Address3 Fields

Bit	Name	Description	Access	Reset
31:16	l4dp3	When Bit 16 (L4PEN3) is reset and Bit 20 (L4DPM3) is set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN3) and Bit 20 (L4DPM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 frames.	RW	0x0
15:0	l4sp3	When Bit 16 (L4PEN3) is reset and Bit 20 (L4DPM3) is set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 frames. When Bit 16 (L4PEN3) and Bit 20 (L4DPM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 frames.	RW	0x0

Layer3_Addr0_Reg3

For IPv4 frames, the Layer 3 Address 0 Register 3 contains the 32-bit IP Source Address field. For IPv6 frames, it contains Bits [31:0] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7004A0
emac1	0xFF702000	0xFF7024A0

Offset: 0x4A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a03 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a03 RW 0x0															

Layer3_Addr0_Reg3 Fields

Bit	Name	Description	Access	Reset
31:0	l3a03	When Bit 0 (L3PEN3) and Bit 2 (L3SAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [31:0] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN3) and Bit 4 (L3DAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [31:0] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN3) is reset and Bit 2 (L3SAM3) is set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the IP Source Address field in the IPv4 frames.	RW	0x0

Layer3_Addr1_Reg3

For IPv4 frames, the Layer 3 Address 1 Register 3 contains the 32-bit IP Destination Address field. For IPv6 frames, it contains Bits [63:32] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7004A4
emac1	0xFF702000	0xFF7024A4

Offset: 0x4A4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a13 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a13 RW 0x0															

Layer3_Addr1_Reg3 Fields

Bit	Name	Description	Access	Reset
31:0	l3a13	When Bit 0 (L3PEN3) and Bit 2 (L3SAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [63:32] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN3) and Bit 4 (L3DAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [63:32] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN3) is reset and Bit 4 (L3DAM3) is set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with the IP Destination Address field in the IPv4 frames.	RW	0x0

Layer3_Addr2_Reg3

For IPv4 frames, the Layer 3 Address 2 Register 3 is reserved. For IPv6 frames, it contains Bits [95:64] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7004A8
emac1	0xFF702000	0xFF7024A8

Offset: 0x4A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a23 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a23 RW 0x0															

Layer3_Addr2_Reg3 Fields

Bit	Name	Description	Access	Reset
31:0	l3a23	When Bit 0 (L3PEN3) and Bit 2 (L3SAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [95:64] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN3) and Bit 4 (L3DAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains value to be matched with Bits [95:64] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN3) is reset in Register 292 (Layer 3 and Layer 4 Control Register 3), this register is not used.	RW	0x0

Layer3_Addr3_Reg3

For IPv4 frames, the Layer 3 Address 3 Register 3 is reserved. For IPv6 frames, it contains Bits [127:96] of the 128-bit IP Source Address or Destination Address field.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7004AC
emac1	0xFF702000	0xFF7024AC

Offset: 0x4AC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
l3a33 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
l3a33 RW 0x0															

Layer3_Addr3_Reg3 Fields

Bit	Name	Description	Access	Reset
31:0	l3a33	When Bit 0 (L3PEN3) and Bit 2 (L3SAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [127:96] of the IP Source Address field in the IPv6 frames. When Bit 0 (L3PEN3) and Bit 4 (L3DAM3) are set in Register 292 (Layer 3 and Layer 4 Control Register 3), this field contains the value to be matched with Bits [127:96] of the IP Destination Address field in the IPv6 frames. When Bit 0 (L3PEN3) is reset in Register 292 (Layer 3 and Layer 4 Control Register 3), this register is not used.	RW	0x0

Hash_Table_Reg0

This register contains the first 32 bits of the hash table. The 256-bit Hash table is used for group address filtering. For hash filtering, the content of the destination address in the incoming frame is passed through the CRC logic and the upper eight bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register X), and the least significant five bits determine the bit within the register. For example, a hash value of 8b'10111111 selects Bit 31 of the Hash Table Register 5. The hash value of the destination address is calculated in the following way: 1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32). 2. Perform bitwise reversal for the value obtained in Step 1. 3. Take the upper 8 bits from the value obtained in Step 2. If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the Bit 1 (Pass All Multicast) is set in Register 1 (MAC Frame Filter), then all multicast frames are accepted regardless of the multicast hash values. Because the Hash Table register is double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table Register X registers are written. Note: Because of double-synchronization, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700500
emac1	0xFF702000	0xFF702500

Offset: 0x500

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht31t0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht31t0 RW 0x0															

Hash_Table_Reg0 Fields

Bit	Name	Description	Access	Reset
31:0	ht31t0	This field contains the first 32 Bits (31:0) of the Hash table.	RW	0x0

Hash_Table_Reg1

This register contains the second 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700504
emac1	0xFF702000	0xFF702504

Offset: 0x504

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht63t32 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht63t32 RW 0x0															

Hash_Table_Reg1 Fields

Bit	Name	Description	Access	Reset
31:0	ht63t32	This field contains the second 32 Bits (63:32) of the Hash table.	RW	0x0

Hash_Table_Reg2

This register contains the third 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700508
emac1	0xFF702000	0xFF702508

Offset: 0x508

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht95t64 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht95t64 RW 0x0															

Hash_Table_Reg2 Fields

Bit	Name	Description	Access	Reset
31:0	ht95t64	This field contains the third 32 Bits (95:64) of the Hash table.	RW	0x0

Hash_Table_Reg3

This register contains the fourth 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70050C
emac1	0xFF702000	0xFF70250C

Offset: 0x50C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">ht127t96
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ht127t96
RW 0x0</td></tr>
</table>

Hash_Table_Reg3 Fields

Bit	Name	Description	Access	Reset
31:0	ht127t96	This field contains the fourth 32 Bits (127:96) of the Hash table.	RW	0x0

Hash_Table_Reg4

This register contains the fifth 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700510
emac1	0xFF702000	0xFF702510

Offset: 0x510

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">ht159t128
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ht159t128
RW 0x0</td></tr>
</table>

Hash_Table_Reg4 Fields

Bit	Name	Description	Access	Reset
31:0	ht159t128	This field contains the fifth 32 Bits (159:128) of the Hash table.	RW	0x0

Hash_Table_Reg5

This register contains the sixth 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700514
emac1	0xFF702000	0xFF702514

Offset: 0x514

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht191t160 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht191t160 RW 0x0															

Hash_Table_Reg5 Fields

Bit	Name	Description	Access	Reset
31:0	ht191t160	This field contains the sixth 32 Bits (191:160) of the Hash table.	RW	0x0

Hash_Table_Reg6

This register contains the seventh 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700518
emac1	0xFF702000	0xFF702518

Offset: 0x518

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht223t196 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht223t196 RW 0x0															

Hash_Table_Reg6 Fields

Bit	Name	Description	Access	Reset
31:0	ht223t196	This field contains the seventh 32 Bits (223:196) of the Hash table.	RW	0x0

Hash_Table_Reg7

This register contains the eighth 32 bits of the hash table.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70051C

Module Instance	Base Address	Register Address
emac1	0xFF702000	0xFF70251C

Offset: 0x51C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ht255t224 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ht255t224 RW 0x0															

Hash_Table_Reg7 Fields

Bit	Name	Description	Access	Reset
31:0	ht255t224	This field contains the eighth 32 Bits (255:224) of the Hash table.	RW	0x0

VLAN_Incl_Reg

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit frames.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700584
emac1	0xFF702000	0xFF702584

Offset: 0x584

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												csvl RW 0x0	vlp RW 0x0	vlc RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vlt RW 0x0															

VLAN_Incl_Reg Fields

Bit	Name	Description	Access	Reset
19	csvl	When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted frames. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the transmitted frames.	RW	0x0
18	vlp	When this bit is set, the control Bits [17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used, and Bits [17:16] are ignored.	RW	0x0
17:16	vlc	* 2'b00: No VLAN tag deletion, insertion, or replacement * 2'b01: VLAN tag deletion The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted frames with VLAN tags. * 2'b10: VLAN tag insertion The MAC inserts VLT in bytes 15 and 16 of the frame after inserting the Type value (0x8100/0x88a8) in bytes 13 and 14. This operation is performed on all transmitted frames, irrespective of whether they already have a VLAN tag. * 2'b11: VLAN tag replacement The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted frames (Bytes 13 and 14 are 0x8100/0x88a8). Note: Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value.	RW	0x0
15:0	vlt	This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority, Bit 12 is the CFI/DEI, and Bits[11:0] are the VLAN tag's VID field.	RW	0x0

VLAN_Hash_Table_Reg

The 16-bit Hash table is used for group address filtering based on VLAN tag when Bit 18 (VTHM) of Register 7 (VLAN Tag Register) is set. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on Bit 16 (ETV) of VLAN Tag Register) in the incoming frame is passed through the CRC logic and the upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table. The hash value of the destination address is calculated in the following way: 1. Calculate the 32-bit CRC for the VLAN tag or ID (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32). 2. Perform bitwise reversal for the value obtained in Step 1. 3. Take the upper four bits from the value obtained in Step 2. If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. Because the Hash Table register is double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written. Notes: * Because of double-synchronization, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700588
emac1	0xFF702000	0xFF702588

Offset: 0x588

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vlht RW 0x0															

VLAN_Hash_Table_Reg Fields

Bit	Name	Description	Access	Reset
15:0	vlht	This field contains the 16-bit VLAN Hash Table.	RW	0x0

Timestamp_Control

This register controls the operation of the System Time generator and the processing of PTP packets for timestamping in the Receiver.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700700
emac1	0xFF702000	0xFF702700

Offset: 0x700

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						atsen0 RW 0x0	atsfc RW 0x0	Reserved					tsenmacaddr RW 0x0	snaptypsel RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tsmstrena RW 0x0	tsevntena RW 0x0	tsipv4ena RW 0x1	tsipv6ena RW 0x0	tsipena RW 0x0	tsver2ena RW 0x0	tsctrlssr RW 0x0	tsenall RW 0x0	Reserved		tsaddreg RW 0x0	tstrig RW 0x0	tsupdt RW 0x0	tsinit RW 0x0	tscfupdt RW 0x0	tsena RW 0x0

Timestamp_Control Fields

Bit	Name	Description	Access	Reset
25	atsen0	This field controls capturing the Auxiliary Snapshot Trigger 0. When this bit is set, the Auxiliary snapshot of event on ptp_aux_trig_i[0] input is enabled. When this bit is reset, the events on this input are ignored. **Value** **Description** 0x0 Auxiliary snapshot of event on ptp_aux_trig_i[0] input is disabled. 0x1 Auxiliary snapshot of event on ptp_aux_trig_i[0] input is enabled.	RW	0x0
24	atsfc	When set, it resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, auxiliary snapshots get stored in the FIFO. **Value** **Description** 0x0 Don't reset Auxiliary Snapshot FIFO pointers 0x1 Reset Auxiliary Snapshot FIFO pointers	RW	0x0
18	tsenmacaddr	When set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP frames when PTP is directly sent over Ethernet. **Value** **Description** 0x0 DA MAC address doesn't filter PTP frames 0x1 DA MAC address filters PTP frames	RW	0x0

Bit	Name	Description	Access	Reset
17:16	`snaptypsel`	These bits along with Bits 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken.	RW	0x0
15	`tsmstrena`	When set, the snapshot is taken only for the messages relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node. **Value** **Description** 0x0 Timestamp snapshot taken for messages relevant to slave node 0x1 Timestamp snapshot taken for messages relevant to master node	RW	0x0
14	`tsevntena`	When set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all messages except Announce, Management, and Signaling. **Value** **Description** 0x0 Timestamp snapshot disabled for event messages 0x1 Timestamp snapshot only for event messages	RW	0x0
13	`tsipv4ena`	When set, the MAC receiver processes the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default. **Value** **Description** 0x0 Don't process PTP packets in UDP over IPv4 0x1 Process PTP packets in UDP over IPv4	RW	0x1
12	`tsipv6ena`	When set, the MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets. **Value** **Description** 0x0 Don't process PTP packets in UDP over IPv6 0x1 Process PTP packets in UDP over IPv6	RW	0x0

Bit	Name	Description	Access	Reset
11	`tsipena`	When set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores the PTP over Ethernet packets. **Value** — **Description** 0x0 — Don't process PTP packets in Ethernet frames 0x1 — Process PTP packets in Ethernet frames	RW	0x0
10	`tsver2ena`	When set, the PTP packets are processed using the 1588 version 2 format. Otherwise, the PTP packets are processed using the version 1 format. **Value** — **Description** 0x0 — PTP packets processed with 1588 version 1 format 0x1 — PTP packets processed with 1588 version 2 format	RW	0x0
9	`tsctrlssr`	When set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment has to be programmed correctly depending on the PTP reference clock frequency and the value of this bit. **Value** — **Description** 0x0 — Timestamp Low register rolls over at 0x7FFF_FFFF 0x1 — Timestamp Low register rolls over at 1ns	RW	0x0
8	`tsenall`	When set, the timestamp snapshot is enabled for all frames received by the MAC. **Value** — **Description** 0x0 — Timestamp snapshot disabled 0x1 — Timestamp snapshot enabled	RW	0x0

Bit	Name	Description	Access	Reset
5	tsaddreg	When set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it. **Value** **Description** 0x0 Timestamp Addend register is not updated 0x1 Timestamp Addend register is updated	RW	0x0
4	tstrig	When set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the generation of the Timestamp Trigger Interrupt. **Value** **Description** 0x0 Timestamp not generated 0x1 Timestamp generated	RW	0x0
3	tsupdt	When set, the system time is updated (added or subtracted) with the value specified in Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the update is completed in hardware. The Timestamp Higher Word register is not updated. **Value** **Description** 0x0 Timestamp not updated (added or subtracted) with values in Register 452 and Register 453 0x1 Timestamp updated (added or subtracted) with values in Register 452 and Register 453	RW	0x0

Bit	Name	Description	Access	Reset
2	`tsinit`	When set, the system time is initialized (overwritten) with the value specified in the Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the initialization is complete. The Timestamp Higher Word register can only be initialized. **Value** — **Description** 0x0 — Timestamp not initialized (overwritten) by values in Register 452 and Register 453 0x1 — Timestamp initialized (overwritten) by values in Register 452 and Register 453	`RW`	`0x0`
1	`tscfupdt`	When set, this bit indicates that the system times update should be done using the fine update method. When reset, it indicates the system timestamp update should be done using the Coarse method. **Value** — **Description** 0x0 — Timestamp Coarse 0x1 — Timestamp Fine	`RW`	`0x0`
0	`tsena`	When set, the timestamp is added for the transmit and receive frames. When disabled, timestamp is not added for the transmit and receive frames and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 frames only if this bit is set. **Value** — **Description** 0x0 — Timestamp not added 0x1 — Timestamp added for transmit and receive	`RW`	`0x0`

Sub_Second_Increment

In the Coarse Update mode (TSCFUPDT bit in Register 448), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In the Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700704`
`emac1`	`0xFF702000`	`0xFF702704`

Offset: 0x704

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ssinc RW 0x0							

Sub_Second_Increment Fields

Bit	Name	Description	Access	Reset
7:0	ssinc	The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time-Nanoseconds register has an accuracy of 1 ns (TSCTRLSSR bit is set). When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465ns. In this case, you should program a value of 43 (0x2B) that is derived by 20ns/0.465.	RW	0x0

System_Time_Seconds

The System Time -Seconds register, along with System-TimeNanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to l3_sp_clk).

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700708
emac1	0xFF702000	0xFF702708

Offset: 0x708

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tss RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tss RO 0x0															

System_Time_Seconds Fields

Bit	Name	Description	Access	Reset
31:0	tss	The value in this field indicates the current value in seconds of the System Time maintained by the MAC.	RO	0x0

System_Time_Nanoseconds

The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set, each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70070C
emac1	0xFF702000	0xFF70270C

Offset: 0x70C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	tsss RO 0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tsss RO 0x0															

System_Time_Nanoseconds Fields

Bit	Name	Description	Access	Reset
30:0	tsss	The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.	RO	0x0

System_Time_Seconds_Update

The System Time - Seconds Update register, along with the System Time - Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both of these registers before setting the TSINIT or TSUPDT bits in the Timestamp Control register.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700710
emac1	0xFF702000	0xFF702710

Offset: 0x710

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tss RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tss RW 0x0															

System_Time_Seconds_Update Fields

Bit	Name	Description	Access	Reset
31:0	tss	The value in this field indicates the time in seconds to be initialized or added to the system time.	RW	0x0

System_Time_Nanoseconds_Update

Update system time

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700714
emac1	0xFF702000	0xFF702714

Offset: 0x714

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addsub RW 0x0	tsss RW 0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tsss RW 0x0															

System_Time_Nanoseconds_Update Fields

Bit	Name	Description	Access	Reset
31	addsub	When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register. **Value** **Description** 0x0 Add Time Value from update reg 0x1 Subtract Time Value of update reg	RW	0x0
30:0	tsss	The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.	RW	0x0

Timestamp_Addend

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in Register 448). This register content is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700718
emac1	0xFF702000	0xFF702718

Offset: 0x718

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tsar RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tsar RW 0x0															

Timestamp_Addend Fields

Bit	Name	Description	Access	Reset
31:0	tsar	This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.	RW	0x0

Target_Time_Seconds

The Target Time Seconds register, along with Target Time Nanoseconds register, is used to schedule an interrupt event (Register 458[1] when Advanced Timestamping is enabled; otherwise, TS interrupt bit in Register14[9]) when the system time exceeds the value programmed in these registers.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70071C
emac1	0xFF702000	0xFF70271C

Offset: 0x71C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">tstr
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">tstr
RW 0x0</td></tr>
</table>

Target_Time_Seconds Fields

Bit	Name	Description	Access	Reset
31:0	tstr	This register stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, then based on Bits [6:5] of Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled).	RW	0x0

Target_Time_Nanoseconds

Target time

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700720
emac1	0xFF702000	0xFF702720

Offset: 0x720

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
trgtbusy RO 0x0	ttslo RW 0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ttslo RW 0x0															

Target_Time_Nanoseconds Fields

Bit	Name	Description	Access	Reset
31	trgtbusy	The MAC sets this bit when the PPSCMD field (Bits[3:0]) in Register 459 (PPS Control Register) is programmed to 010 or 011. Programming the PPSCMD field to 010 or 011, instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. This bit is reserved when the Enable Flexible Pulse-Per-Second Output feature is not selected.	RO	0x0
30:0	ttslo	This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the both Target Timestamp registers, then based on the TRGTMODSEL0 field (Bits [6:5]) in Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). This value should not exceed 0x3B9A_C9FF when TSCTRLSSR is set in the Timestamp control register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.	RW	0x0

System_Time_Higher_Word_Seconds

System time higher word

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700724
emac1	0xFF702000	0xFF702724

Offset: 0x724

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tshwr RW 0x0															

System_Time_Higher_Word_Seconds Fields

Bit	Name	Description	Access	Reset
15:0	tshwr	This field contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register.	RW	0x0

Timestamp_Status

Timestamp status. All bits except Bits[27:25] get cleared when the host reads this register.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700728
emac1	0xFF702000	0xFF702728

Offset: 0x728

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		atsns RO 0x0					atsstm RO 0x0	Reserved				atsstn RO 0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												tstrgterr RO 0x0	auxtstrig RO 0x0	tstargt RO 0x0	tssovf RO 0x0

Timestamp_Status Fields

Bit	Name	Description	Access	Reset
29:25	atsns	This field indicates the number of Snapshots available in the FIFO. A value of 16 (equal to the depth of the FIFO) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.	RO	0x0
24	atsstm	This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO. **Value** **Description** 0x0 Not Active 0x1 Aux timestamp snapshot full	RO	0x0
19:16	atsstn	These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list: * Bit 16: Auxiliary trigger 0 * Bit 17: Auxiliary trigger 1 * Bit 18: Auxiliary trigger 2 * Bit 19: Auxiliary trigger 3 The software can read this register to find the triggers that are set when the timestamp is taken.	RO	0x0
3	tstrgterr	This bit is set when the target time, being programmed in Target Time Registers, is already elapsed. This bit is cleared when read by the application. **Value** **Description** 0x0 When Read resets 0x1 Target Time Elapsed -Reg455 and Reg456	RO	0x0
2	auxtstrig	This bit is set high when the auxiliary snapshot is written to the FIFO. **Value** **Description** 0x0 System Time 0x1 System Time is >= Reg455 and Reg456	RO	0x0

Bit	Name	Description	Access	Reset
1	`tstargt`	When set, this bit indicates that the value of system time is greater or equal to the value specified in the Register 455 (Target Time Seconds Register) and Register 456 (Target Time Nanoseconds Register). **Value** — **Description** 0x0 — System Time 0x1 — System Time is >= Reg455 and Reg456	RO	0x0
0	`tssovf`	When set, this bit indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. **Value** — **Description** 0x0 — No Overflow 0x1 — Seconds Overflow	RO	0x0

PPS_Control

Controls timestamp Pulse-Per-Second output

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70072C`
`emac1`	`0xFF702000`	`0xFF70272C`

Offset: `0x72C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									trgtmodsel0 RW 0x0		ppsen0 RW 0x0	ppsctrl_ppscmd RW 0x0			

PPS_Control Fields

Bit	Name	Description	Access	Reset
6:5	trgtmodsel0	This field indicates the Target Time registers (register 455 and 456) mode for PPS0 output signal **Value** / **Description** 0x0 Target Time regs generate interrupt event. 0x2 Target Time gen. interr event and sig pps0 0x3 Target Time No inter just start and stop sig pps0	RW	0x0
4	ppsen0	When set low, Bits[3:0] function as PPSCTRL (backward compatible). When set high, Bits[3:0] function as PPSCMD. **Value** / **Description** 0x0 Bits[3:0] function as ppsctrl0 0x1 Bits[3:0] function as ppscmd	RW	0x0

Bit	Name	Description	Access	Reset
3:0	`ppsctrl_ppscmd`	PPSCTRL0: PPS0 Output Frequency Control This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: -0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz. -0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz. -0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz. -0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz. -... -1111: The binary rollover is 32.768 KHz, and the digital rollover is 16.384 KHz. Note: In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example: * When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms * When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of: - One clock of 50 percent duty cycle and 537 ms period - Second clock of 463 ms period (268 ms low and 195 ms high) * When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of: - Three clocks of 50 percent duty cycle and 268 ms period - Fourth clock of 195 ms period (134 ms low and 61 ms high) This behavior is because of the non-linear toggling of bits in the digital rollover mode in Register 451 (System Time - Nanoseconds Register). Flexible PPS0 Output (ptp_pps_o[0]) Control Programming these bits with a non-zero value instructs the MAC to initiate an event. Once the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The Software should ensure that these bits are programmed only when they are all-zero. The following list describes the values of PPSCMD0: * 0000: No Command * 0001: START Single Pulse This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS0 Width Register. * 0010: START Pulse Train This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands. * 0011: Cancel START This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time. * 0100: STOP Pulse train at time This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses. * 0101: STOP Pulse Train immediately This command immediately stops the train of pulses initiated by the START Pulse Train	RW	0x0

Auxiliary_Timestamp_Nanoseconds

This register, along with Register 461 (Auxiliary Timestamp Seconds Register), gives the 64-bit timestamp stored as auxiliary snapshot. The two registers together form the read port of a 64-bit wide FIFO with a depth of 16. Multiple snapshots can be stored in this FIFO. The ATSNS bits in the Timestamp Status register indicate the fill-level of this FIFO. The top of the FIFO is removed only when the last byte of Register 461 (Auxiliary Timestamp - Seconds Register) is read. In the little-endian mode, this means when Bits[31:24] are read. In big-endian mode, it corresponds to the reading of Bits[7:0] of Register 461 (Auxiliary Timestamp - Seconds Register).

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700730
emac1	0xFF702000	0xFF702730

Offset: 0x730

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	auxtslo RO 0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
auxtslo RO 0x0															

Auxiliary_Timestamp_Nanoseconds Fields

Bit	Name	Description	Access	Reset
30:0	auxtslo	Contains the lower 32 bits (nano-seconds field) of the auxiliary timestamp.	RO	0x0

Auxiliary_Timestamp_Seconds

Contains the higher 32 bits (Seconds field) of the auxiliary timestamp.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700734
emac1	0xFF702000	0xFF702734

Offset: 0x734

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
auxtshi RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
auxtshi RO 0x0															

Auxiliary_Timestamp_Seconds Fields

Bit	Name	Description	Access	Reset
31:0	auxtshi	Contains the higher 32 bits (Seconds field) of the auxiliary timestamp.	RO	0x0

PPS0_Interval

The PPS0 Interval register contains the number of units of sub-second increment value between the rising edges of PPS0 signal output (ptp_pps_o[0]).

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700760
emac1	0xFF702000	0xFF702760

Offset: 0x760

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ppsint RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ppsint RW 0x0															

PPS0_Interval Fields

Bit	Name	Description	Access	Reset
31:0	ppsint	These bits store the interval between the rising edges of PPS0 signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20ns), and desired interval between rising edges of PPS0 signal output is 100ns (that is, five units of sub-second increment value), then you should program value 4 (5 -1) in this register.	RW	0x0

PPS0_Width

The PPS0 Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (ptp_pps_o[0]).

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700764`
`emac1`	`0xFF702000`	`0xFF702764`

Offset: `0x764`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`ppswidth` `RW 0x0`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`ppswidth` `RW 0x0`															

PPS0_Width Fields

Bit	Name	Description	Access	Reset
31:0	`ppswidth`	These bits store the width between the rising edge and corresponding falling edge of the PPS0 signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20ns), and desired width between the rising and corresponding falling edges of PPS0 signal output is 80ns (that is, four units of sub-second increment value), then you should program value 3 (4-1) in this register. Note: The value programmed in this register must be lesser than the value programmed in Register 472 (PPS0 Interval Register).	`RW`	`0x0`

MAC_Address16_High

The MAC Address16 High register holds the upper 16 bits of the 17th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address16 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700800`
`emac1`	`0xFF702000`	`0xFF702800`

Offset: 0x800

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address16_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 17th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address16[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address16[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0

Bit	Name	Description	Access	Reset
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 17th 6-byte MAC address.	RW	0xFFFF

MAC_Address16_Low

The MAC Address16 Low register holds the lower 32 bits of the 17th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700804
emac1	0xFF702000	0xFF702804

Offset: 0x804

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address16_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 17th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address17_High

The MAC Address17 High register holds the upper 16 bits of the 18th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address17 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700808
emac1	0xFF702000	0xFF702808

Offset: 0x808

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address17_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 18th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address17[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address17[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 18th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address17_Low

The MAC Address17 Low register holds the lower 32 bits of the 18th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70080C`
`emac1`	`0xFF702000`	`0xFF70280C`

Offset: `0x80C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address17_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 18th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address18_High

The MAC Address18 High register holds the upper 16 bits of the 19th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address18 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700810
emac1	0xFF702000	0xFF702810

Offset: 0x810

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address18_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 19th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address18[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address18[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 19th 6-byte MAC address.	RW	0xFFFF

MAC_Address18_Low

The MAC Address18 Low register holds the lower 32 bits of the 19th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700814
emac1	0xFF702000	0xFF702814

Offset: 0x814

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address18_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 19th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address19_High

The MAC Address19 High register holds the upper 16 bits of the 20th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address19 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700818
emac1	0xFF702000	0xFF702818

Offset: 0x818

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address19_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 20th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address19[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address19[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 20th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address19_Low

The MAC Address19 Low register holds the lower 32 bits of the 20th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70081C`
`emac1`	`0xFF702000`	`0xFF70281C`

Offset: `0x81C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address19_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 20th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFFF

MAC_Address20_High

The MAC Address20 High register holds the upper 16 bits of the 21th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address20 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700820
emac1	0xFF702000	0xFF702820

Offset: 0x820

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address20_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 21th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address20[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address20[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 21th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address20_Low

The MAC Address20 Low register holds the lower 32 bits of the 21th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700824`
`emac1`	`0xFF702000`	`0xFF702824`

Offset: `0x824`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address20_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 21th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address21_High

The MAC Address21 High register holds the upper 16 bits of the 22th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address21 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700828
emac1	0xFF702000	0xFF702828

Offset: 0x828

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address21_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 22th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address21[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address21[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0: Byte is unmasked (i.e. is compared) 0x1: Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0: Byte is unmasked (i.e. is compared) 0x1: Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 22th 6-byte MAC address.	RW	0xFFFF

MAC_Address21_Low

The MAC Address21 Low register holds the lower 32 bits of the 22th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70082C
emac1	0xFF702000	0xFF70282C

Offset: `0x82C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address21_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 22th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address22_High

The MAC Address22 High register holds the upper 16 bits of the 23th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address22 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700830
emac1	0xFF702000	0xFF702830

Offset: 0x830

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address22_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 23th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address22[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address22[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 23th 6-byte MAC address.	RW	0xFFFF

MAC_Address22_Low

The MAC Address22 Low register holds the lower 32 bits of the 23th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700834`
`emac1`	`0xFF702000`	`0xFF702834`

Offset: `0x834`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address22_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 23th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address23_High

The MAC Address23 High register holds the upper 16 bits of the 24th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address23 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700838
emac1	0xFF702000	0xFF702838

Offset: 0x838

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address23_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 24th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address23[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address23[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 24th 6-byte MAC address.	RW	0xFFFF

MAC_Address23_Low

The MAC Address23 Low register holds the lower 32 bits of the 24th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70083C
emac1	0xFF702000	0xFF70283C

Offset: 0x83C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address23_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 24th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address24_High

The MAC Address24 High register holds the upper 16 bits of the 25th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address24 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700840
emac1	0xFF702000	0xFF702840

Offset: 0x840

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address24_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 25th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address24[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address24[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 25th 6-byte MAC address.	RW	0xFFFF

MAC_Address24_Low

The MAC Address24 Low register holds the lower 32 bits of the 25th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700844
emac1	0xFF702000	0xFF702844

Offset: `0x844`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address24_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 25th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address25_High

The MAC Address25 High register holds the upper 16 bits of the 26th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address25 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700848
emac1	0xFF702000	0xFF702848

Offset: 0x848

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address25_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 26th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address25[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address25[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 26th 6-byte MAC address.	RW	0xFFFF

MAC_Address25_Low

The MAC Address25 Low register holds the lower 32 bits of the 26th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70084C
emac1	0xFF702000	0xFF70284C

Offset: 0x84C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address25_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 26th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address26_High

The MAC Address26 High register holds the upper 16 bits of the 27th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address26 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700850
emac1	0xFF702000	0xFF702850

Offset: 0x850

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address26_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 27th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address26[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address26[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 27th 6-byte MAC address.	RW	0xFFFF

MAC_Address26_Low

The MAC Address26 Low register holds the lower 32 bits of the 27th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700854
emac1	0xFF702000	0xFF702854

Offset: 0x854

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address26_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 27th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address27_High

The MAC Address27 High register holds the upper 16 bits of the 28th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address27 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700858
emac1	0xFF702000	0xFF702858

Offset: 0x858

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address27_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 28th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address27[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address27[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 28th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address27_Low

The MAC Address27 Low register holds the lower 32 bits of the 28th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70085C`
`emac1`	`0xFF702000`	`0xFF70285C`

Offset: `0x85C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address27_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 28th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address28_High

The MAC Address28 High register holds the upper 16 bits of the 29th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address28 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700860
emac1	0xFF702000	0xFF702860

Offset: 0x860

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address28_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 29th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address28[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address28[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 29th 6-byte MAC address.	RW	0xFFFF

MAC_Address28_Low

The MAC Address28 Low register holds the lower 32 bits of the 29th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700864
emac1	0xFF702000	0xFF702864

Offset: 0x864

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address28_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 29th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address29_High

The MAC Address29 High register holds the upper 16 bits of the 30th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address29 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700868
emac1	0xFF702000	0xFF702868

Offset: 0x868

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address29_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 30th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address29[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address29[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 30th 6-byte MAC address.	RW	0xFFFF

MAC_Address29_Low

The MAC Address29 Low register holds the lower 32 bits of the 30th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70086C
emac1	0xFF702000	0xFF70286C

Offset: 0x86C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address29_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 30th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address30_High

The MAC Address30 High register holds the upper 16 bits of the 31th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address30 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700870
emac1	0xFF702000	0xFF702870

Offset: 0x870

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address30_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 31th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address30[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address30[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 31th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address30_Low

The MAC Address30 Low register holds the lower 32 bits of the 31th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700874`
`emac1`	`0xFF702000`	`0xFF702874`

Offset: `0x874`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address30_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 31th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address31_High

The MAC Address31 High register holds the upper 16 bits of the 32th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address31 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700878
emac1	0xFF702000	0xFF702878

Offset: 0x878

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address31_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 32th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address31[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address31[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 32th 6-byte MAC address.	RW	0xFFFF

MAC_Address31_Low

The MAC Address31 Low register holds the lower 32 bits of the 32th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70087C
emac1	0xFF702000	0xFF70287C

Offset: 0x87C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address31_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 32th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address32_High

The MAC Address32 High register holds the upper 16 bits of the 33th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address32 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700880
emac1	0xFF702000	0xFF702880

Offset: 0x880

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address32_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 33th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address32[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address32[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address32 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 33th 6-byte MAC address.	RW	0xFFFF

MAC_Address32_Low

The MAC Address32 Low register holds the lower 32 bits of the 33th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700884
emac1	0xFF702000	0xFF702884

Offset: 0x884

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address32_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 33th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address33_High

The MAC Address33 High register holds the upper 16 bits of the 34th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address33 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700888
emac1	0xFF702000	0xFF702888

Offset: 0x888

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address33_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 34th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address33[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address33[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address33 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 34th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address33_Low

The MAC Address33 Low register holds the lower 32 bits of the 34th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70088C`
`emac1`	`0xFF702000`	`0xFF70288C`

Offset: `0x88C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address33_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 34th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address34_High

The MAC Address34 High register holds the upper 16 bits of the 35th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address34 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700890
emac1	0xFF702000	0xFF702890

Offset: 0x890

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address34_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 35th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address34[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address34[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address34 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 35th 6-byte MAC address.	RW	0xFFFF

MAC_Address34_Low

The MAC Address34 Low register holds the lower 32 bits of the 35th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700894`
`emac1`	`0xFF702000`	`0xFF702894`

Offset: `0x894`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address34_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 35th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address35_High

The MAC Address35 High register holds the upper 16 bits of the 36th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address35 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700898
emac1	0xFF702000	0xFF702898

Offset: 0x898

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address35_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 36th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address35[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address35[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address35 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 36th 6-byte MAC address.	RW	0xFFFF

MAC_Address35_Low

The MAC Address35 Low register holds the lower 32 bits of the 36th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70089C
emac1	0xFF702000	0xFF70289C

Offset: 0x89C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address35_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 36th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address36_High

The MAC Address36 High register holds the upper 16 bits of the 37th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address36 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008A0
emac1	0xFF702000	0xFF7028A0

Offset: 0x8A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address36_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 37th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address36[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address36[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address36 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 37th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address36_Low

The MAC Address36 Low register holds the lower 32 bits of the 37th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008A4`
`emac1`	`0xFF702000`	`0xFF7028A4`

Offset: `0x8A4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address36_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 37th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address37_High

The MAC Address37 High register holds the upper 16 bits of the 38th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address37 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008A8
emac1	0xFF702000	0xFF7028A8

Offset: 0x8A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address37_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 38th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address37[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address37[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address37 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 38th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address37_Low

The MAC Address37 Low register holds the lower 32 bits of the 38th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008AC`
`emac1`	`0xFF702000`	`0xFF7028AC`

Offset: `0x8AC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address37_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 38th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address38_High

The MAC Address38 High register holds the upper 16 bits of the 39th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address38 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008B0
emac1	0xFF702000	0xFF7028B0

Offset: 0x8B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address38_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 39th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address38[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address38[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address38 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 39th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address38_Low

The MAC Address38 Low register holds the lower 32 bits of the 39th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008B4`
`emac1`	`0xFF702000`	`0xFF7028B4`

Offset: `0x8B4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address38_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 39th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address39_High

The MAC Address39 High register holds the upper 16 bits of the 40th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address39 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008B8
emac1	0xFF702000	0xFF7028B8

Offset: 0x8B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address39_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 40th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address39[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address39[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address39 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 40th 6-byte MAC address.	RW	0xFFFF

MAC_Address39_Low

The MAC Address39 Low register holds the lower 32 bits of the 40th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008BC`
`emac1`	`0xFF702000`	`0xFF7028BC`

Offset: `0x8BC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address39_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 40th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address40_High

The MAC Address40 High register holds the upper 16 bits of the 41th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address40 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008C0
emac1	0xFF702000	0xFF7028C0

Offset: 0x8C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address40_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 41th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address40[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address40[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address40 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 41th 6-byte MAC address.	RW	0xFFFF

MAC_Address40_Low

The MAC Address40 Low register holds the lower 32 bits of the 41th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008C4
emac1	0xFF702000	0xFF7028C4

Offset: 0x8C4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address40_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 41th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address41_High

The MAC Address41 High register holds the upper 16 bits of the 42th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address41 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008C8
emac1	0xFF702000	0xFF7028C8

Offset: 0x8C8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address41_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 42th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address41[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address41[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address41 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 42th 6-byte MAC address.	RW	0xFFFF

MAC_Address41_Low

The MAC Address41 Low register holds the lower 32 bits of the 42th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008CC
emac1	0xFF702000	0xFF7028CC

Offset: 0x8CC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address41_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 42th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address42_High

The MAC Address42 High register holds the upper 16 bits of the 43th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address42 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008D0
emac1	0xFF702000	0xFF7028D0

Offset: 0x8D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address42_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 43th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address42[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address42[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address42 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 43th 6-byte MAC address.	RW	0xFFFF

MAC_Address42_Low

The MAC Address42 Low register holds the lower 32 bits of the 43th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008D4
emac1	0xFF702000	0xFF7028D4

Offset: 0x8D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address42_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 43th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address43_High

The MAC Address43 High register holds the upper 16 bits of the 44th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address43 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008D8
emac1	0xFF702000	0xFF7028D8

Offset: 0x8D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address43_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 44th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr>
<tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address43[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address43[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address43 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 44th 6-byte MAC address.	RW	0xFFFF

MAC_Address43_Low

The MAC Address43 Low register holds the lower 32 bits of the 44th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008DC
emac1	0xFF702000	0xFF7028DC

Offset: 0x8DC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address43_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 44th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address44_High

The MAC Address44 High register holds the upper 16 bits of the 45th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address44 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008E0
emac1	0xFF702000	0xFF7028E0

Offset: 0x8E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address44_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 45th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address44[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address44[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address44 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 45th 6-byte MAC address.	RW	0xFFFF

MAC_Address44_Low

The MAC Address44 Low register holds the lower 32 bits of the 45th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008E4
emac1	0xFF702000	0xFF7028E4

Offset: 0x8E4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address44_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 45th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address45_High

The MAC Address45 High register holds the upper 16 bits of the 46th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address45 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008E8
emac1	0xFF702000	0xFF7028E8

Offset: 0x8E8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address45_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 46th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address45[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address45[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address45 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 46th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address45_Low

The MAC Address45 Low register holds the lower 32 bits of the 46th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008EC`
`emac1`	`0xFF702000`	`0xFF7028EC`

Offset: `0x8EC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address45_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 46th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address46_High

The MAC Address46 High register holds the upper 16 bits of the 47th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address46 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008F0
emac1	0xFF702000	0xFF7028F0

Offset: 0x8F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address46_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 47th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address46[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address46[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address46 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 47th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address46_Low

The MAC Address46 Low register holds the lower 32 bits of the 47th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008F4`
`emac1`	`0xFF702000`	`0xFF7028F4`

Offset: `0x8F4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address46_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 47th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address47_High

The MAC Address47 High register holds the upper 16 bits of the 48th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address47 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7008F8
emac1	0xFF702000	0xFF7028F8

Offset: 0x8F8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address47_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 48th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address47[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address47[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address47 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 48th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address47_Low

The MAC Address47 Low register holds the lower 32 bits of the 48th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7008FC`
`emac1`	`0xFF702000`	`0xFF7028FC`

Offset: `0x8FC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address47_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 48th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address48_High

The MAC Address48 High register holds the upper 16 bits of the 49th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address48 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700900
emac1	0xFF702000	0xFF702900

Offset: 0x900

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address48_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 49th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr>
<tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address48[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address48[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address48 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 49th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address48_Low

The MAC Address48 Low register holds the lower 32 bits of the 49th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700904`
`emac1`	`0xFF702000`	`0xFF702904`

Offset: `0x904`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address48_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 49th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address49_High

The MAC Address49 High register holds the upper 16 bits of the 50th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address49 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700908
emac1	0xFF702000	0xFF702908

Offset: 0x908

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address49_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 50th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address49[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address49[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address49 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 50th 6-byte MAC address.	RW	0xFFFF

MAC_Address49_Low

The MAC Address49 Low register holds the lower 32 bits of the 50th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70090C
emac1	0xFF702000	0xFF70290C

Offset: 0x90C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address49_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 50th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address50_High

The MAC Address50 High register holds the upper 16 bits of the 51th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address50 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700910
emac1	0xFF702000	0xFF702910

Offset: 0x910

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address50_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 51th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address50[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address50[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address50 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 51th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address50_Low

The MAC Address50 Low register holds the lower 32 bits of the 51th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700914`
`emac1`	`0xFF702000`	`0xFF702914`

Offset: `0x914`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address50_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 51th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address51_High

The MAC Address51 High register holds the upper 16 bits of the 52th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address51 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700918
emac1	0xFF702000	0xFF702918

Offset: 0x918

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address51_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 52th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address51[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address51[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address51 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 52th 6-byte MAC address.	RW	0xFFFF

MAC_Address51_Low

The MAC Address51 Low register holds the lower 32 bits of the 52th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70091C
emac1	0xFF702000	0xFF70291C

Offset: 0x91C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address51_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 52th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address52_High

The MAC Address52 High register holds the upper 16 bits of the 53th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address52 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700920
emac1	0xFF702000	0xFF702920

Offset: 0x920

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address52_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 53th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address52[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address52[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address52 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 53th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address52_Low

The MAC Address52 Low register holds the lower 32 bits of the 53th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700924`
`emac1`	`0xFF702000`	`0xFF702924`

Offset: `0x924`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address52_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 53th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address53_High

The MAC Address53 High register holds the upper 16 bits of the 54th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address53 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700928
emac1	0xFF702000	0xFF702928

Offset: 0x928

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address53_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 54th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address53[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address53[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address53 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 54th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address53_Low

The MAC Address53 Low register holds the lower 32 bits of the 54th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70092C`
`emac1`	`0xFF702000`	`0xFF70292C`

Offset: `0x92C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address53_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 54th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address54_High

The MAC Address54 High register holds the upper 16 bits of the 55th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address54 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700930
emac1	0xFF702000	0xFF702930

Offset: 0x930

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address54_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 55th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address54[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address54[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address54 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 55th 6-byte MAC address.	RW	0xFFFF

MAC_Address54_Low

The MAC Address54 Low register holds the lower 32 bits of the 55th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700934`
`emac1`	`0xFF702000`	`0xFF702934`

Offset: `0x934`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address54_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 55th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address55_High

The MAC Address55 High register holds the upper 16 bits of the 56th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address55 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700938
emac1	0xFF702000	0xFF702938

Offset: 0x938

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address55_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 56th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address55[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address55[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address55 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 56th 6-byte MAC address.	RW	0xFFFF

MAC_Address55_Low

The MAC Address55 Low register holds the lower 32 bits of the 56th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70093C
emac1	0xFF702000	0xFF70293C

Offset: 0x93C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address55_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 56th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address56_High

The MAC Address56 High register holds the upper 16 bits of the 57th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address56 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700940
emac1	0xFF702000	0xFF702940

Offset: 0x940

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address56_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 57th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address56[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address56[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address56 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 57th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address56_Low

The MAC Address56 Low register holds the lower 32 bits of the 57th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700944`
`emac1`	`0xFF702000`	`0xFF702944`

Offset: `0x944`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address56_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 57th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address57_High

The MAC Address57 High register holds the upper 16 bits of the 58th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address57 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700948
emac1	0xFF702000	0xFF702948

Offset: 0x948

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address57_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 58th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address57[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address57[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address57 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 58th 6-byte MAC address.	RW	0xFFFF

MAC_Address57_Low

The MAC Address57 Low register holds the lower 32 bits of the 58th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70094C
emac1	0xFF702000	0xFF70294C

Offset: 0x94C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address57_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 58th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address58_High

The MAC Address58 High register holds the upper 16 bits of the 59th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address58 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700950
emac1	0xFF702000	0xFF702950

Offset: 0x950

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address58_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 59th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address58[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address58[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address58 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 59th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address58_Low

The MAC Address58 Low register holds the lower 32 bits of the 59th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700954`
`emac1`	`0xFF702000`	`0xFF702954`

Offset: `0x954`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address58_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 59th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address59_High

The MAC Address59 High register holds the upper 16 bits of the 60th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address59 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700958
emac1	0xFF702000	0xFF702958

Offset: 0x958

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address59_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 60th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address59[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address59[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address59 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 60th 6-byte MAC address.	RW	0xFFFF

MAC_Address59_Low

The MAC Address59 Low register holds the lower 32 bits of the 60th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70095C
emac1	0xFF702000	0xFF70295C

Offset: 0x95C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address59_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 60th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address60_High

The MAC Address60 High register holds the upper 16 bits of the 61th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address60 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700960
emac1	0xFF702000	0xFF702960

Offset: 0x960

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address60_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 61th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address60[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address60[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address60 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 61th 6-byte MAC address.	RW	0xFFFF

MAC_Address60_Low

The MAC Address60 Low register holds the lower 32 bits of the 61th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700964
emac1	0xFF702000	0xFF702964

Offset: 0x964

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address60_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 61th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address61_High

The MAC Address61 High register holds the upper 16 bits of the 62th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address61 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700968
emac1	0xFF702000	0xFF702968

Offset: 0x968

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address61_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 62th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address61[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address61[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address61 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 62th 6-byte MAC address.	RW	0xFFFF

MAC_Address61_Low

The MAC Address61 Low register holds the lower 32 bits of the 62th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70096C`
`emac1`	`0xFF702000`	`0xFF70296C`

Offset: `0x96C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address61_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 62th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address62_High

The MAC Address62 High register holds the upper 16 bits of the 63th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address62 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700970
emac1	0xFF702000	0xFF702970

Offset: 0x970

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address62_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 63th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address62[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address62[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address62 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 63th 6-byte MAC address.	RW	0xFFFF

MAC_Address62_Low

The MAC Address62 Low register holds the lower 32 bits of the 63th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700974
emac1	0xFF702000	0xFF702974

Offset: 0x974

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address62_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 63th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address63_High

The MAC Address63 High register holds the upper 16 bits of the 64th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address63 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700978
emac1	0xFF702000	0xFF702978

Offset: 0x978

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address63_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 64th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address63[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address63[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address63 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 64th 6-byte MAC address.	RW	0xFFFF

MAC_Address63_Low

The MAC Address63 Low register holds the lower 32 bits of the 64th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70097C
emac1	0xFF702000	0xFF70297C

Offset: 0x97C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address63_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 64th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address64_High

The MAC Address64 High register holds the upper 16 bits of the 65th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address64 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700980
emac1	0xFF702000	0xFF702980

Offset: 0x980

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address64_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 65th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address64[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address64[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address64 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 65th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address64_Low

The MAC Address64 Low register holds the lower 32 bits of the 65th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700984`
`emac1`	`0xFF702000`	`0xFF702984`

Offset: `0x984`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address64_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 65th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address65_High

The MAC Address65 High register holds the upper 16 bits of the 66th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address65 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700988
emac1	0xFF702000	0xFF702988

Offset: 0x988

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address65_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 66th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address65[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address65[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address65 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 66th 6-byte MAC address.	RW	0xFFFF

MAC_Address65_Low

The MAC Address65 Low register holds the lower 32 bits of the 66th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70098C
emac1	0xFF702000	0xFF70298C

Offset: 0x98C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address65_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 66th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address66_High

The MAC Address66 High register holds the upper 16 bits of the 67th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address66 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700990
emac1	0xFF702000	0xFF702990

Offset: 0x990

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address66_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 67th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address66[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address66[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address66 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 67th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address66_Low

The MAC Address66 Low register holds the lower 32 bits of the 67th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700994`
`emac1`	`0xFF702000`	`0xFF702994`

Offset: `0x994`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address66_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 67th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address67_High

The MAC Address67 High register holds the upper 16 bits of the 68th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address67 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700998
emac1	0xFF702000	0xFF702998

Offset: 0x998

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address67_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 68th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr>
<tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address67[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address67[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address67 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 68th 6-byte MAC address.	RW	0xFFFF

MAC_Address67_Low

The MAC Address67 Low register holds the lower 32 bits of the 68th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70099C
emac1	0xFF702000	0xFF70299C

Offset: 0x99C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address67_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 68th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address68_High

The MAC Address68 High register holds the upper 16 bits of the 69th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address68 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009A0
emac1	0xFF702000	0xFF7029A0

Offset: 0x9A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address68_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 69th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address68[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address68[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address68 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 69th 6-byte MAC address.	RW	0xFFFF

MAC_Address68_Low

The MAC Address68 Low register holds the lower 32 bits of the 69th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009A4
emac1	0xFF702000	0xFF7029A4

Offset: 0x9A4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address68_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 69th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address69_High

The MAC Address69 High register holds the upper 16 bits of the 70th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address69 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009A8
emac1	0xFF702000	0xFF7029A8

Offset: 0x9A8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address69_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 70th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address69[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address69[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address69 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 70th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address69_Low

The MAC Address69 Low register holds the lower 32 bits of the 70th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009AC`
`emac1`	`0xFF702000`	`0xFF7029AC`

Offset: `0x9AC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address69_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 70th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address70_High

The MAC Address70 High register holds the upper 16 bits of the 71th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address70 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009B0
emac1	0xFF702000	0xFF7029B0

Offset: 0x9B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address70_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 71th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address70[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address70[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address70 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 71th 6-byte MAC address.	RW	0xFFFF

MAC_Address70_Low

The MAC Address70 Low register holds the lower 32 bits of the 71th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009B4
emac1	0xFF702000	0xFF7029B4

Offset: `0x9B4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address70_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 71th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address71_High

The MAC Address71 High register holds the upper 16 bits of the 72th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address71 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009B8
emac1	0xFF702000	0xFF7029B8

Offset: 0x9B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address71_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 72th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address71[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address71[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address71 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 72th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address71_Low

The MAC Address71 Low register holds the lower 32 bits of the 72th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009BC`
`emac1`	`0xFF702000`	`0xFF7029BC`

Offset: `0x9BC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address71_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 72th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address72_High

The MAC Address72 High register holds the upper 16 bits of the 73th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address72 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009C0
emac1	0xFF702000	0xFF7029C0

Offset: 0x9C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address72_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 73th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address72[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address72[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>25</td><td><code>mbc_1</code></td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td><code>RW</code></td><td><code>0x0</code></td></tr>
<tr><td>24</td><td><code>mbc_0</code></td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address72 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td><code>RW</code></td><td><code>0x0</code></td></tr>
<tr><td>15:0</td><td><code>addrhi</code></td><td>This field contains the upper 16 bits (47:32) of the 73th 6-byte MAC address.</td><td><code>RW</code></td><td><code>0xFFFF</code></td></tr>
</table>

MAC_Address72_Low

The MAC Address72 Low register holds the lower 32 bits of the 73th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009C4`
`emac1`	`0xFF702000`	`0xFF7029C4`

Offset: `0x9C4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address72_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 73th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address73_High

The MAC Address73 High register holds the upper 16 bits of the 74th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address73 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009C8
emac1	0xFF702000	0xFF7029C8

Offset: 0x9C8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address73_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 74th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address73[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address73[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address73 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** – **Description** 0x0 – Byte is unmasked (i.e. is compared) 0x1 – Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 74th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address73_Low

The MAC Address73 Low register holds the lower 32 bits of the 74th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009CC`
`emac1`	`0xFF702000`	`0xFF7029CC`

Offset: `0x9CC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address73_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 74th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address74_High

The MAC Address74 High register holds the upper 16 bits of the 75th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address74 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009D0
emac1	0xFF702000	0xFF7029D0

Offset: 0x9D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address74_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 75th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address74[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address74[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address74 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 75th 6-byte MAC address.	RW	0xFFFF

MAC_Address74_Low

The MAC Address74 Low register holds the lower 32 bits of the 75th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009D4
emac1	0xFF702000	0xFF7029D4

Offset: 0x9D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address74_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 75th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address75_High

The MAC Address75 High register holds the upper 16 bits of the 76th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address75 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009D8
emac1	0xFF702000	0xFF7029D8

Offset: 0x9D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address75_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 76th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address75[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address75[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address75 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 76th 6-byte MAC address.	RW	0xFFFF

MAC_Address75_Low

The MAC Address75 Low register holds the lower 32 bits of the 76th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009DC
emac1	0xFF702000	0xFF7029DC

Offset: 0x9DC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address75_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 76th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address76_High

The MAC Address76 High register holds the upper 16 bits of the 77th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address76 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009E0
emac1	0xFF702000	0xFF7029E0

Offset: 0x9E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address76_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 77th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address76[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address76[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address76 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 77th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address76_Low

The MAC Address76 Low register holds the lower 32 bits of the 77th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009E4`
`emac1`	`0xFF702000`	`0xFF7029E4`

Offset: `0x9E4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address76_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 77th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address77_High

The MAC Address77 High register holds the upper 16 bits of the 78th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address77 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009E8
emac1	0xFF702000	0xFF7029E8

Offset: 0x9E8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address77_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 78th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr><tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address77[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address77[47:0] is used to compare with the DA fields of the received frame.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>MAC address compare disabled</td></tr><tr><td>0x1</td><td>MAC address compare enabled</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address77 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 78th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address77_Low

The MAC Address77 Low register holds the lower 32 bits of the 78th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF7009EC`
`emac1`	`0xFF702000`	`0xFF7029EC`

Offset: `0x9EC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address77_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 78th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address78_High

The MAC Address78 High register holds the upper 16 bits of the 79th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address78 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009F0
emac1	0xFF702000	0xFF7029F0

Offset: 0x9F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address78_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 79th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address78[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address78[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address78 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 79th 6-byte MAC address.	RW	0xFFFF

MAC_Address78_Low

The MAC Address78 Low register holds the lower 32 bits of the 79th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009F4
emac1	0xFF702000	0xFF7029F4

Offset: 0x9F4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address78_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 79th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address79_High

The MAC Address79 High register holds the upper 16 bits of the 80th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address79 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009F8
emac1	0xFF702000	0xFF7029F8

Offset: 0x9F8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address79_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 80th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address79[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address79[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address79 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 80th 6-byte MAC address.	RW	0xFFFF

MAC_Address79_Low

The MAC Address79 Low register holds the lower 32 bits of the 80th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF7009FC
emac1	0xFF702000	0xFF7029FC

Offset: 0x9FC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address79_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 80th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address80_High

The MAC Address80 High register holds the upper 16 bits of the 81th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address80 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A00
emac1	0xFF702000	0xFF702A00

Offset: 0xA00

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address80_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 81th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address80[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address80[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address80 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 81th 6-byte MAC address.	RW	0xFFFF

MAC_Address80_Low

The MAC Address80 Low register holds the lower 32 bits of the 81th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A04
emac1	0xFF702000	0xFF702A04

Offset: 0xA04

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address80_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 81th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address81_High

The MAC Address81 High register holds the upper 16 bits of the 82th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address81 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A08
emac1	0xFF702000	0xFF702A08

Offset: 0xA08

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address81_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 82th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address81[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address81[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address81 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 82th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address81_Low

The MAC Address81 Low register holds the lower 32 bits of the 82th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A0C`
`emac1`	`0xFF702000`	`0xFF702A0C`

Offset: `0xA0C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address81_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 82th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address82_High

The MAC Address82 High register holds the upper 16 bits of the 83th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address82 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A10
emac1	0xFF702000	0xFF702A10

Offset: 0xA10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address82_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 83th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address82[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address82[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address82 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 83th 6-byte MAC address.	RW	0xFFFF

MAC_Address82_Low

The MAC Address82 Low register holds the lower 32 bits of the 83th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A14
emac1	0xFF702000	0xFF702A14

Offset: 0xA14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address82_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 83th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address83_High

The MAC Address83 High register holds the upper 16 bits of the 84th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address83 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A18
emac1	0xFF702000	0xFF702A18

Offset: 0xA18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address83_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 84th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address83[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address83[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address83 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 84th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address83_Low

The MAC Address83 Low register holds the lower 32 bits of the 84th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A1C`
`emac1`	`0xFF702000`	`0xFF702A1C`

Offset: `0xA1C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address83_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 84th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address84_High

The MAC Address84 High register holds the upper 16 bits of the 85th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address84 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A20
emac1	0xFF702000	0xFF702A20

Offset: 0xA20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address84_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 85th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address84[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address84[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address84 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 85th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address84_Low

The MAC Address84 Low register holds the lower 32 bits of the 85th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A24`
`emac1`	`0xFF702000`	`0xFF702A24`

Offset: `0xA24`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address84_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 85th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address85_High

The MAC Address85 High register holds the upper 16 bits of the 86th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address85 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A28
emac1	0xFF702000	0xFF702A28

Offset: 0xA28

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address85_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 86th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address85[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address85[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address85 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 86th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address85_Low

The MAC Address85 Low register holds the lower 32 bits of the 86th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A2C`
`emac1`	`0xFF702000`	`0xFF702A2C`

Offset: `0xA2C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address85_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 86th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address86_High

The MAC Address86 High register holds the upper 16 bits of the 87th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address86 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A30
emac1	0xFF702000	0xFF702A30

Offset: 0xA30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address86_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 87th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr>
<tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address86[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address86[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address86 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 87th 6-byte MAC address.	RW	0xFFFF

MAC_Address86_Low

The MAC Address86 Low register holds the lower 32 bits of the 87th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A34`
`emac1`	`0xFF702000`	`0xFF702A34`

Offset: `0xA34`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address86_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 87th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address87_High

The MAC Address87 High register holds the upper 16 bits of the 88th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address87 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A38
emac1	0xFF702000	0xFF702A38

Offset: 0xA38

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address87_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 88th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address87[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address87[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address87 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 88th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address87_Low

The MAC Address87 Low register holds the lower 32 bits of the 88th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A3C`
`emac1`	`0xFF702000`	`0xFF702A3C`

Offset: `0xA3C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address87_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 88th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address88_High

The MAC Address88 High register holds the upper 16 bits of the 89th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address88 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A40
emac1	0xFF702000	0xFF702A40

Offset: 0xA40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address88_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 89th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address88[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address88[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address88 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 89th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address88_Low

The MAC Address88 Low register holds the lower 32 bits of the 89th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A44`
`emac1`	`0xFF702000`	`0xFF702A44`

Offset: `0xA44`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address88_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 89th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address89_High

The MAC Address89 High register holds the upper 16 bits of the 90th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address89 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A48
emac1	0xFF702000	0xFF702A48

Offset: 0xA48

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address89_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 90th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address89[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address89[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address89 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 90th 6-byte MAC address.	RW	0xFFFF

MAC_Address89_Low

The MAC Address89 Low register holds the lower 32 bits of the 90th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A4C
emac1	0xFF702000	0xFF702A4C

Offset: 0xA4C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address89_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 90th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address90_High

The MAC Address90 High register holds the upper 16 bits of the 91th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address90 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A50
emac1	0xFF702000	0xFF702A50

Offset: 0xA50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address90_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 91th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address90[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address90[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address90 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 91th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address90_Low

The MAC Address90 Low register holds the lower 32 bits of the 91th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A54`
`emac1`	`0xFF702000`	`0xFF702A54`

Offset: `0xA54`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address90_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 91th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address91_High

The MAC Address91 High register holds the upper 16 bits of the 92th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address91 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A58
emac1	0xFF702000	0xFF702A58

Offset: 0xA58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address91_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 92th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address91[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address91[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address91 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 92th 6-byte MAC address.	RW	0xFFFF

MAC_Address91_Low

The MAC Address91 Low register holds the lower 32 bits of the 92th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A5C
emac1	0xFF702000	0xFF702A5C

Offset: 0xA5C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address91_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 92th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address92_High

The MAC Address92 High register holds the upper 16 bits of the 93th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address92 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A60
emac1	0xFF702000	0xFF702A60

Offset: 0xA60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address92_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 93th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address92[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address92[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address92 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 93th 6-byte MAC address.	RW	0xFFFF

MAC_Address92_Low

The MAC Address92 Low register holds the lower 32 bits of the 93th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A64
emac1	0xFF702000	0xFF702A64

Offset: `0xA64`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address92_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 93th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address93_High

The MAC Address93 High register holds the upper 16 bits of the 94th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address93 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A68
emac1	0xFF702000	0xFF702A68

Offset: 0xA68

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address93_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 94th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address93[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address93[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address93 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 94th 6-byte MAC address.	RW	0xFFFF

MAC_Address93_Low

The MAC Address93 Low register holds the lower 32 bits of the 94th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A6C
emac1	0xFF702000	0xFF702A6C

Offset: 0xA6C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address93_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 94th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address94_High

The MAC Address94 High register holds the upper 16 bits of the 95th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address94 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A70
emac1	0xFF702000	0xFF702A70

Offset: 0xA70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address94_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 95th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address94[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address94[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address94 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 95th 6-byte MAC address.	RW	0xFFFF

MAC_Address94_Low

The MAC Address94 Low register holds the lower 32 bits of the 95th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A74`
`emac1`	`0xFF702000`	`0xFF702A74`

Offset: `0xA74`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address94_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 95th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address95_High

The MAC Address95 High register holds the upper 16 bits of the 96th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address95 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A78
emac1	0xFF702000	0xFF702A78

Offset: 0xA78

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address95_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 96th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address95[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address95[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address95 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 96th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address95_Low

The MAC Address95 Low register holds the lower 32 bits of the 96th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A7C`
`emac1`	`0xFF702000`	`0xFF702A7C`

Offset: `0xA7C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address95_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 96th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address96_High

The MAC Address96 High register holds the upper 16 bits of the 97th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address96 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A80
emac1	0xFF702000	0xFF702A80

Offset: 0xA80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address96_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 97th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address96[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address96[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address96 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 97th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address96_Low

The MAC Address96 Low register holds the lower 32 bits of the 97th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A84`
`emac1`	`0xFF702000`	`0xFF702A84`

Offset: `0xA84`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address96_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 97th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address97_High

The MAC Address97 High register holds the upper 16 bits of the 98th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address97 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A88
emac1	0xFF702000	0xFF702A88

Offset: 0xA88

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address97_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 98th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address97[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address97[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address97 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 98th 6-byte MAC address.	RW	0xFFFF

MAC_Address97_Low

The MAC Address97 Low register holds the lower 32 bits of the 98th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A8C`
`emac1`	`0xFF702000`	`0xFF702A8C`

Offset: `0xA8C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address97_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 98th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address98_High

The MAC Address98 High register holds the upper 16 bits of the 99th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address98 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A90
emac1	0xFF702000	0xFF702A90

Offset: 0xA90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address98_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 99th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address98[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address98[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address98 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 99th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address98_Low

The MAC Address98 Low register holds the lower 32 bits of the 99th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A94`
`emac1`	`0xFF702000`	`0xFF702A94`

Offset: `0xA94`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address98_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 99th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address99_High

The MAC Address99 High register holds the upper 16 bits of the 100th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address99 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700A98
emac1	0xFF702000	0xFF702A98

Offset: 0xA98

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address99_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 100th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address99[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address99[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address99 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 100th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address99_Low

The MAC Address99 Low register holds the lower 32 bits of the 100th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700A9C`
`emac1`	`0xFF702000`	`0xFF702A9C`

Offset: `0xA9C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address99_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 100th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address100_High

The MAC Address100 High register holds the upper 16 bits of the 101th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address100 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AA0
emac1	0xFF702000	0xFF702AA0

Offset: 0xAA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address100_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 101th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address100[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address100[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address100 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 101th 6-byte MAC address.	RW	0xFFFF

MAC_Address100_Low

The MAC Address100 Low register holds the lower 32 bits of the 101th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700AA4`
`emac1`	`0xFF702000`	`0xFF702AA4`

Offset: `0xAA4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address100_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 101th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address101_High

The MAC Address101 High register holds the upper 16 bits of the 102th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address101 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AA8
emac1	0xFF702000	0xFF702AA8

Offset: 0xAA8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address101_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 102th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address101[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address101[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address101 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 102th 6-byte MAC address.	RW	0xFFFF

MAC_Address101_Low

The MAC Address101 Low register holds the lower 32 bits of the 102th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AAC
emac1	0xFF702000	0xFF702AAC

Offset: 0xAAC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address101_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 102th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address102_High

The MAC Address102 High register holds the upper 16 bits of the 103th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address102 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AB0
emac1	0xFF702000	0xFF702AB0

Offset: 0xAB0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address102_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 103th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address102[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address102[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address102 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 103th 6-byte MAC address.	RW	0xFFFF

MAC_Address102_Low

The MAC Address102 Low register holds the lower 32 bits of the 103th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AB4
emac1	0xFF702000	0xFF702AB4

Offset: `0xAB4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address102_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 103th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address103_High

The MAC Address103 High register holds the upper 16 bits of the 104th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address103 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AB8
emac1	0xFF702000	0xFF702AB8

Offset: 0xAB8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address103_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 104th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address103[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address103[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address103 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 104th 6-byte MAC address.	RW	0xFFFF

MAC_Address103_Low

The MAC Address103 Low register holds the lower 32 bits of the 104th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700ABC
emac1	0xFF702000	0xFF702ABC

Offset: 0xABC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address103_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 104th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address104_High

The MAC Address104 High register holds the upper 16 bits of the 105th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address104 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AC0
emac1	0xFF702000	0xFF702AC0

Offset: 0xAC0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address104_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 105th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address104[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address104[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address104 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 105th 6-byte MAC address.	RW	0xFFFF

MAC_Address104_Low

The MAC Address104 Low register holds the lower 32 bits of the 105th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AC4
emac1	0xFF702000	0xFF702AC4

Offset: 0xAC4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address104_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 105th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address105_High

The MAC Address105 High register holds the upper 16 bits of the 106th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address105 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AC8
emac1	0xFF702000	0xFF702AC8

Offset: 0xAC8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address105_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 106th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address105[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address105[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address105 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 106th 6-byte MAC address.	RW	0xFFFF

MAC_Address105_Low

The MAC Address105 Low register holds the lower 32 bits of the 106th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700ACC
emac1	0xFF702000	0xFF702ACC

Offset: 0xACC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address105_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 106th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address106_High

The MAC Address106 High register holds the upper 16 bits of the 107th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address106 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AD0
emac1	0xFF702000	0xFF702AD0

Offset: 0xAD0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address106_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 107th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** / **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address106[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address106[47:0] is used to compare with the DA fields of the received frame. **Value** / **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** / **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address106 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 107th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address106_Low

The MAC Address106 Low register holds the lower 32 bits of the 107th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700AD4`
`emac1`	`0xFF702000`	`0xFF702AD4`

Offset: `0xAD4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address106_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 107th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address107_High

The MAC Address107 High register holds the upper 16 bits of the 108th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address107 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AD8
emac1	0xFF702000	0xFF702AD8

Offset: 0xAD8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address107_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 108th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address107[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address107[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address107 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 108th 6-byte MAC address.	RW	0xFFFF

MAC_Address107_Low

The MAC Address107 Low register holds the lower 32 bits of the 108th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700ADC
emac1	0xFF702000	0xFF702ADC

Offset: `0xADC`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address107_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 108th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address108_High

The MAC Address108 High register holds the upper 16 bits of the 109th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address108 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AE0
emac1	0xFF702000	0xFF702AE0

Offset: 0xAE0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address108_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 109th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address108[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address108[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address108 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 109th 6-byte MAC address.	RW	0xFFFF

MAC_Address108_Low

The MAC Address108 Low register holds the lower 32 bits of the 109th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AE4
emac1	0xFF702000	0xFF702AE4

Offset: 0xAE4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address108_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 109th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address109_High

The MAC Address109 High register holds the upper 16 bits of the 110th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address109 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AE8
emac1	0xFF702000	0xFF702AE8

Offset: 0xAE8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address109_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 110th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address109[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address109[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>mbc_4</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>27</td><td>mbc_3</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>26</td><td>mbc_2</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr><tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address109 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 110th 6-byte MAC address.	RW	0xFFFF

MAC_Address109_Low

The MAC Address109 Low register holds the lower 32 bits of the 110th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AEC
emac1	0xFF702000	0xFF702AEC

Offset: 0xAEC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address109_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 110th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address110_High

The MAC Address110 High register holds the upper 16 bits of the 111th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address110 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AF0
emac1	0xFF702000	0xFF702AF0

Offset: 0xAF0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address110_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 111th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address110[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address110[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address110 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 111th 6-byte MAC address.	RW	0xFFFF

MAC_Address110_Low

The MAC Address110 Low register holds the lower 32 bits of the 111th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AF4
emac1	0xFF702000	0xFF702AF4

Offset: 0xAF4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address110_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 111th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address111_High

The MAC Address111 High register holds the upper 16 bits of the 112th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address111 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AF8
emac1	0xFF702000	0xFF702AF8

Offset: 0xAF8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address111_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 112th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address111[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address111[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address111 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 112th 6-byte MAC address.	RW	0xFFFF

MAC_Address111_Low

The MAC Address111 Low register holds the lower 32 bits of the 112th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700AFC
emac1	0xFF702000	0xFF702AFC

Offset: 0xAFC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address111_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 112th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address112_High

The MAC Address112 High register holds the upper 16 bits of the 113th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address112 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B00
emac1	0xFF702000	0xFF702B00

Offset: 0xB00

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address112_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 113th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address112[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address112[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address112 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 113th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address112_Low

The MAC Address112 Low register holds the lower 32 bits of the 113th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B04`
`emac1`	`0xFF702000`	`0xFF702B04`

Offset: `0xB04`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address112_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 113th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address113_High

The MAC Address113 High register holds the upper 16 bits of the 114th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address113 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B08
emac1	0xFF702000	0xFF702B08

Offset: 0xB08

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address113_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 114th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address113[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address113[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address113 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 114th 6-byte MAC address.	RW	0xFFFF

MAC_Address113_Low

The MAC Address113 Low register holds the lower 32 bits of the 114th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B0C
emac1	0xFF702000	0xFF702B0C

Offset: 0xB0C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address113_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 114th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address114_High

The MAC Address114 High register holds the upper 16 bits of the 115th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address114 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B10
emac1	0xFF702000	0xFF702B10

Offset: 0xB10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address114_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 115th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address114[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address114[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address114 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 115th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address114_Low

The MAC Address114 Low register holds the lower 32 bits of the 115th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B14`
`emac1`	`0xFF702000`	`0xFF702B14`

Offset: `0xB14`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address114_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 115th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address115_High

The MAC Address115 High register holds the upper 16 bits of the 116th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address115 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B18
emac1	0xFF702000	0xFF702B18

Offset: 0xB18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address115_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 116th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address115[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address115[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address115 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 116th 6-byte MAC address.	RW	0xFFFF

MAC_Address115_Low

The MAC Address115 Low register holds the lower 32 bits of the 116th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B1C
emac1	0xFF702000	0xFF702B1C

Offset: 0xB1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address115_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 116th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address116_High

The MAC Address116 High register holds the upper 16 bits of the 117th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address116 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B20
emac1	0xFF702000	0xFF702B20

Offset: 0xB20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address116_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 117th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	`RW`	`0x0`
30	`sa`	When this bit is enabled, the MAC Address116[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address116[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	`RW`	`0x0`
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address116 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 117th 6-byte MAC address.	RW	0xFFFF

MAC_Address116_Low

The MAC Address116 Low register holds the lower 32 bits of the 117th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B24
emac1	0xFF702000	0xFF702B24

Offset: 0xB24

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address116_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 117th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address117_High

The MAC Address117 High register holds the upper 16 bits of the 118th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address117 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B28
emac1	0xFF702000	0xFF702B28

Offset: 0xB28

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address117_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 118th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address117[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address117[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address117 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 118th 6-byte MAC address.	RW	0xFFFF

MAC_Address117_Low

The MAC Address117 Low register holds the lower 32 bits of the 118th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B2C
emac1	0xFF702000	0xFF702B2C

Offset: 0xB2C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address117_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 118th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address118_High

The MAC Address118 High register holds the upper 16 bits of the 119th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address118 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B30
emac1	0xFF702000	0xFF702B30

Offset: 0xB30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address118_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 119th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address118[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address118[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address118 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 119th 6-byte MAC address.	RW	0xFFFF

MAC_Address118_Low

The MAC Address118 Low register holds the lower 32 bits of the 119th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B34
emac1	0xFF702000	0xFF702B34

Offset: 0xB34

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address118_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 119th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address119_High

The MAC Address119 High register holds the upper 16 bits of the 120th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address119 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B38
emac1	0xFF702000	0xFF702B38

Offset: 0xB38

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address119_High Fields

Bit	Name	Description	Access	Reset
31	`ae`	When this bit is enabled, the address filter block uses the 120th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	`sa`	When this bit is enabled, the MAC Address119[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address119[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	`mbc_5`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address119 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 120th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address119_Low

The MAC Address119 Low register holds the lower 32 bits of the 120th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B3C`
`emac1`	`0xFF702000`	`0xFF702B3C`

Offset: `0xB3C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address119_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 120th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address120_High

The MAC Address120 High register holds the upper 16 bits of the 121th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address120 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B40
emac1	0xFF702000	0xFF702B40

Offset: 0xB40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address120_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 121th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address120[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address120[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address120 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 121th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address120_Low

The MAC Address120 Low register holds the lower 32 bits of the 121th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B44`
`emac1`	`0xFF702000`	`0xFF702B44`

Offset: `0xB44`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address120_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 121th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address121_High

The MAC Address121 High register holds the upper 16 bits of the 122th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address121 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B48
emac1	0xFF702000	0xFF702B48

Offset: 0xB48

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address121_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 122th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address121[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address121[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address121 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 122th 6-byte MAC address.	RW	0xFFFF

MAC_Address121_Low

The MAC Address121 Low register holds the lower 32 bits of the 122th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B4C
emac1	0xFF702000	0xFF702B4C

Offset: `0xB4C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address121_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 122th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address122_High

The MAC Address122 High register holds the upper 16 bits of the 123th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address122 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B50
emac1	0xFF702000	0xFF702B50

Offset: 0xB50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address122_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 123th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address122[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address122[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address122 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 123th 6-byte MAC address.	RW	0xFFFF

MAC_Address122_Low

The MAC Address122 Low register holds the lower 32 bits of the 123th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B54
emac1	0xFF702000	0xFF702B54

Offset: `0xB54`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address122_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 123th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address123_High

The MAC Address123 High register holds the upper 16 bits of the 124th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address123 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B58
emac1	0xFF702000	0xFF702B58

Offset: 0xB58

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address123_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 124th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** — **Description** 0x0 — Second MAC address filtering disabled 0x1 — Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address123[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address123[47:0] is used to compare with the DA fields of the received frame. **Value** — **Description** 0x0 — MAC address compare disabled 0x1 — MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address123 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 124th 6-byte MAC address.	RW	0xFFFF

MAC_Address123_Low

The MAC Address123 Low register holds the lower 32 bits of the 124th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B5C
emac1	0xFF702000	0xFF702B5C

Offset: 0xB5C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address123_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 124th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address124_High

The MAC Address124 High register holds the upper 16 bits of the 125th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address124 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B60
emac1	0xFF702000	0xFF702B60

Offset: 0xB60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address124_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 125th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address124[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address124[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address124 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 125th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address124_Low

The MAC Address124 Low register holds the lower 32 bits of the 125th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B64`
`emac1`	`0xFF702000`	`0xFF702B64`

Offset: `0xB64`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address124_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 125th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address125_High

The MAC Address125 High register holds the upper 16 bits of the 126th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address125 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B68
emac1	0xFF702000	0xFF702B68

Offset: 0xB68

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address125_High Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>ae</td><td>When this bit is enabled, the address filter block uses the 126th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Second MAC address filtering disabled</td></tr>
<tr><td>0x1</td><td>Second MAC address filtering enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>30</td><td>sa</td><td>When this bit is enabled, the MAC Address125[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address125[47:0] is used to compare with the DA fields of the received frame.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>MAC address compare disabled</td></tr>
<tr><td>0x1</td><td>MAC address compare enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>29</td><td>mbc_5</td><td>This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0).
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Byte is unmasked (i.e. is compared)</td></tr>
<tr><td>0x1</td><td>Byte is masked (i.e. not compared)</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address125 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 126th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address125_Low

The MAC Address125 Low register holds the lower 32 bits of the 126th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B6C`
`emac1`	`0xFF702000`	`0xFF702B6C`

Offset: `0xB6C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address125_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 126th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address126_High

The MAC Address126 High register holds the upper 16 bits of the 127th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address126 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B70
emac1	0xFF702000	0xFF702B70

Offset: 0xB70

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address126_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 127th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address126[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address126[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	mbc_4	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
27	mbc_3	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0
26	mbc_2	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
25	mbc_1	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
24	mbc_0	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address126 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** — **Description** 0x0 — Byte is unmasked (i.e. is compared) 0x1 — Byte is masked (i.e. not compared)	RW	0x0
15:0	addrhi	This field contains the upper 16 bits (47:32) of the 127th 6-byte MAC address.	RW	0xFFFF

MAC_Address126_Low

The MAC Address126 Low register holds the lower 32 bits of the 127th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B74
emac1	0xFF702000	0xFF702B74

Offset: 0xB74

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address126_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 127th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

MAC_Address127_High

The MAC Address127 High register holds the upper 16 bits of the 128th 6-byte MAC address of the station. Because the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, the synchronization is triggered only when bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address127 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain. Note that all MAC Address High registers (except MAC Address0 High) have the same format.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF700B78
emac1	0xFF702000	0xFF702B78

Offset: 0xB78

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ae RW 0x0	sa RW 0x0	mbc_5 RW 0x0	mbc_4 RW 0x0	mbc_3 RW 0x0	mbc_2 RW 0x0	mbc_1 RW 0x0	mbc_0 RW 0x0	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrhi RW 0xFFFF															

MAC_Address127_High Fields

Bit	Name	Description	Access	Reset
31	ae	When this bit is enabled, the address filter block uses the 128th MAC address for perfect filtering. When this bit is disabled, the address filter block ignores the address for filtering. **Value** **Description** 0x0 Second MAC address filtering disabled 0x1 Second MAC address filtering enabled	RW	0x0
30	sa	When this bit is enabled, the MAC Address127[47:0] is used to compare with the SA fields of the received frame. When this bit is disabled, the MAC Address127[47:0] is used to compare with the DA fields of the received frame. **Value** **Description** 0x0 MAC address compare disabled 0x1 MAC address compare enabled	RW	0x0
29	mbc_5	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	RW	0x0

Bit	Name	Description	Access	Reset
28	`mbc_4`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
27	`mbc_3`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
26	`mbc_2`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
25	`mbc_1`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
24	`mbc_0`	This array of bits are mask control bits for comparison of each of the MAC Address bytes. When masked, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address127 high and low registers. Each bit controls the masking of the bytes. You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. The array index corresponds to the byte (e.g. index 0 is for bits 7:0). **Value** **Description** 0x0 Byte is unmasked (i.e. is compared) 0x1 Byte is masked (i.e. not compared)	`RW`	`0x0`
15:0	`addrhi`	This field contains the upper 16 bits (47:32) of the 128th 6-byte MAC address.	`RW`	`0xFFFF`

MAC_Address127_Low

The MAC Address127 Low register holds the lower 32 bits of the 128th 6-byte MAC address of the station. Note that all MAC Address Low registers (except MAC Address0 Low) have the same format.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF700B7C`
`emac1`	`0xFF702000`	`0xFF702B7C`

Offset: `0xB7C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
addrlo RW 0xFFFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
addrlo RW 0xFFFFFFFF															

MAC_Address127_Low Fields

Bit	Name	Description	Access	Reset
31:0	addrlo	This field contains the lower 32 bits of the 128th 6-byte MAC address. The content of this field is undefined until loaded by software after the initialization process.	RW	0xFFFFFFFF

DMA Register Group Register Descriptions

DMA Register Group

Offset: 0x1000

Bus_Mode on page 17-808
The Bus Mode register establishes the bus operating modes for the DMA.

Transmit_Poll_Demand on page 17-812
The Transmit Poll Demand register enables the Tx DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the Tx DMA if it is in the Suspend mode. The Tx DMA can go into the Suspend mode because of an Underflow error in a transmitted frame or the unavailability of descriptors owned by it. You can give this command anytime and the Tx DMA resets this command when it again starts fetching the current descriptor from host memory.

Receive_Poll_Demand on page 17-813
The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

Receive_Descriptor_List_Address on page 17-813
The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Transmit_Descriptor_List_Address on page 17-814
The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. You can write to this register only when the Tx DMA has stopped, that is, Bit 13 (ST) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Status on page 17-815
The Status register contains all status bits that the DMA reports to the host. The software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing 1'b1 to (unreserved) Bits[16:0] of this register clears these bits and writing 1'b0 has no effect. Each field (Bits[16:0]) can be masked by masking the appropriate bit in Register 7 (Interrupt Enable Register).

Operation_Mode on page 17-820
The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization.

Interrupt_Enable on page 17-826
The Interrupt Enable register enables the interrupts reported by Register 5 (Status Register). Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Missed_Frame_And_Buffer_Overflow_Counter on page 17-830
The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

Receive_Interrupt_Watchdog_Timer on page 17-831
This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of Register 5 (Status Register)

AXI_Bus_Mode on page 17-832
The AXI Bus Mode Register controls the behavior of the AXI master. It is mainly used to control the burst splitting and the number of outstanding requests.

AHB_or_AXI_Status on page 17-835
This register provides the active status of the AXI interface's read and write channels. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

Current_Host_Transmit_Descriptor on page 17-836
The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

Current_Host_Receive_Descriptor on page 17-836
The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

Current_Host_Transmit_Buffer_Address on page 17-837
The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

Current_Host_Receive_Buffer_Address on page 17-837
The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

HW_Feature on page 17-838
This register indicates the presence of the optional features or functions of the gmac. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Bus_Mode

The Bus Mode register establishes the bus operating modes for the DMA.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701000
emac1	0xFF702000	0xFF703000

Offset: `0x1000`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						aal RW 0x0	eightxpbl RW 0x0	usp RW 0x0	rpbl RW 0x1						fb RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		pbl RW 0x1						atds RW 0x0	dsl RW 0x0					Reserved	swr RW 0x1

Bus_Mode Fields

Bit	Name	Description	Access	Reset
25	aal	When this bit is set high and the FB bit is equal to 1, the AHB or AXI interface generates all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address. **Value** **Description** 0x0 No Address-Aligned Beats 0x1 Address-Aligned Beats (dependent on FB)	RW	0x0

Bit	Name	Description	Access	Reset
24	eightxpbl	When set high, this bit multiplies the programmed PBL value (Bits[22:17] and Bits[13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. **Value** **Description** 0x0 Non Multiply Mode 0x1 Multiplies PBL value by 8	RW	0x0
23	usp	When set high, this bit configures the Rx DMA to use the value configured in Bits[22:17] as PBL. The PBL value in Bits[13:8] is applicable only to the Tx DMA operations. When reset to low, the PBL value in Bits[13:8] is applicable for both DMA engines. **Value** **Description** 0x0 Configures TX RX DMA to PBL 0x1 Configures TX DMA to PBL	RW	0x0
22:17	rpbl	This field indicates the maximum number of beats to be transferred in one Rx DMA transaction. This is the maximum value that is used in a single block Read or Write. The Rx DMA always attempts to burst as specified in the RPBL bit each time it starts a Burst transfer on the host bus. You can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high. **Value** **Description** 0x1 Beats Trans. in one Rx DMA Transaction 0x2 Beats Trans. in one Rx DMA Transaction 0x4 Beats Trans. in one Rx DMA Transaction 0x8 Beats Trans. in one Rx DMA Transaction 0x10 Beats Trans. in one Rx DMA Transaction 0x20 Beats Trans. in one Rx DMA Transaction	RW	0x1

Bit	Name	Description	Access	Reset
16	`fb`	This bit controls whether the AXI Master interface performs fixed burst transfers or not. When set, the AXI interface uses FIXED bursts during the start of the normal burst transfers. When reset, the AXI interface uses SINGLE and INCR burst transfer operations. For more information, see Bit 0 (UNDEFINED) of the AXI Bus Mode register. **Value** — **Description** 0x0 — SINGLE or INCR Burst 0x1 — FIXED Burst (1, 4, 8, or 16)	RW	0x0
13:8	`pbl`	These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for Tx DMA transactions. If the number of beats to be transferred is more than 32, then perform the following steps: 1. Set the 8xPBL mode. 2. Set the PBL. For example, if the maximum number of beats to be transferred is 64, then first set 8xPBL to 1 and then set PBL to 8. The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified.	RW	0x1

Bit	Name	Description	Access	Reset
7	`atds`	When set, the size of the alternate descriptor increases to 32 bytes (8 DWORDS). This is required when the Advanced Timestamp feature or the IPC Full Offload Engine (Type 2) is enabled in the receiver. The enhanced descriptor is not required if the Advanced Timestamp and IPC Full Checksum Offload (Type 2) features are not enabled. In such cases, you can use the 16 bytes descriptor to save 4 bytes of memory. When reset, the descriptor size reverts back to 4 DWORDs (16 bytes). This bit preserves the backward compatibility for the descriptor size. In versions prior to 3.50a, the descriptor size is 16 bytes for both normal and enhanced descriptor. In version 3.50a, descriptor size is increased to 32 bytes because of the Advanced Timestamp and IPC Full Checksum Offload Engine (Type 2) features. **Value** — **Description** 0x0 — MAC DMA Controller Clears Logic 0x1 — MAC DMA Controller Resets Logic	`RW`	`0x0`
6:2	`dsl`	This bit specifies the number of Word, Dword, or Lword (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA in Ring mode.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	swr	When this bit is set, the MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the EMAC clock domains. Before reprogramming any register of the EMAC, you should read a zero (0) value in this bit . Note: * The Software reset function is driven only by this bit. Bit 0 of Register 64 (Channel 1 Bus Mode Register) or Register 128 (Channel 2 Bus Mode Register) has no impact on the Software reset function. * The reset operation is completed only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for the software reset completion. **Value** **Description** 0x0 MAC DMA Controller Clears Logic 0x1 MAC DMA Controller Resets Logic	RW	0x1

Transmit_Poll_Demand

The Transmit Poll Demand register enables the Tx DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the Tx DMA if it is in the Suspend mode. The Tx DMA can go into the Suspend mode because of an Underflow error in a transmitted frame or the unavailability of descriptors owned by it. You can give this command anytime and the Tx DMA resets this command when it again starts fetching the current descriptor from host memory.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701004
emac1	0xFF702000	0xFF703004

Offset: 0x1004

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">tpd
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">tpd
RW 0x0</td></tr>
</table>

Transmit_Poll_Demand Fields

Bit	Name	Description	Access	Reset
31:0	tpd	When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 18 (Current Host Transmit Descriptor Register). If that descriptor is not available (owned by the Host), the transmission returns to the Suspend state and the Bit 2 (TU) of Register 5 (Status Register) is asserted. If the descriptor is available, the transmission resumes.	RW	0x0

Receive_Poll_Demand

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701008
emac1	0xFF702000	0xFF703008

Offset: 0x1008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rpd RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rpd RW 0x0															

Receive_Poll_Demand Fields

Bit	Name	Description	Access	Reset
31:0	rpd	When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 19 (Current Host Receive Descriptor Register). If that descriptor is not available (owned by the Host), the reception returns to the Suspended state and the Bit 7 (RU) of Register 5 (Status Register) is not asserted. If the descriptor is available, the Rx DMA returns to the active state.	RW	0x0

Receive_Descriptor_List_Address

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70100C
emac1	0xFF702000	0xFF70300C

Offset: 0x100C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
rdesla_32bit RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rdesla_32bit RW 0x0														Reserved	

Receive_Descriptor_List_Address Fields

Bit	Name	Description	Access	Reset
31:2	rdesla_32bit	This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits (1:0) are ignored (32-bit wide bus) and internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).	RW	0x0

Transmit_Descriptor_List_Address

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. You can write to this register only when the Tx DMA has stopped, that is, Bit 13 (ST) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701010
emac1	0xFF702000	0xFF703010

Offset: `0x1010`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
tdesla_32bit RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
tdesla_32bit RW 0x0														Reserved	

Transmit_Descriptor_List_Address Fields

Bit	Name	Description	Access	Reset
31:2	tdesla_32bit	This field contains the base address of the first descriptor in the Transmit Descriptor list. The LSB bits (1:0) are ignored (32-bit wide bus) and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).	RW	0x0

Status

The Status register contains all status bits that the DMA reports to the host. The software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing 1'b1 to (unreserved) Bits[16:0] of this register clears these bits and writing 1'b0 has no effect. Each field (Bits[16:0]) can be masked by masking the appropriate bit in Register 7 (Interrupt Enable Register).

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701014
emac1	0xFF702000	0xFF703014

Offset: `0x1014`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	glpii RO 0x0	tti RO 0x0	Reserved	gmi RO 0x0	gli RO 0x0	eb RO 0x0			ts RO 0x0			rs RO 0x0			nis RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ais RW 0x0	eri RW 0x0	fbi RW 0x0	Reserved		eti RW 0x0	rwt RW 0x0	rps RW 0x0	ru RW 0x0	ri RW 0x0	unf RW 0x0	ovf RW 0x0	tjt RW 0x0	tu RW 0x0	tps RW 0x0	ti RW 0x0

Status Fields

Bit	Name	Description	Access	Reset
30	`glpii`	This bit indicates an interrupt event in the LPI logic of the EMAC. To reset this bit to 1'b0, the software must read the corresponding registers in the EMAC to get the exact cause of the interrupt and clear its source. When this bit is high, the interrupt signal from the MAC (sbd_intr_o) is high. **Value** — **Description** 0x0 — No Interrupt 0x1 — GMAC LPI Interrupt	RO	0x0
29	`tti`	This bit indicates an interrupt event in the Timestamp Generator block of EMAC. The software must read the corresponding registers in the EMAC to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the EMAC subsystem (sbd_intr_o) is high. **Value** — **Description** 0x0 — No Interrupt 0x1 — Timestamp Trigger Interrupt	RO	0x0
27	`gmi`	This bit reflects an interrupt event in the MMC block of the EMAC. The software must read the corresponding registers in the EMAC to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the EMAC subsystem (sbd_intr_o) is high when this bit is high. **Value** — **Description** 0x0 — No Interrupt 0x1 — GMAC MMC Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
26	gli	This bit reflects an interrupt event in the PCS (link change and AN complete), SMII (link change), or RGMII (link change) interface block of the EMAC. The software must read the corresponding registers (Register 49 for PCS or Register 54 for SMII or RGMII) in the EMAC to get the exact cause of the interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the EMAC subsystem (sbd_intr_o) is high when this bit is high. **Value** — **Description** 0x0 — No Interrupt 0x1 — GMAC Line Interrupt	RO	0x0
25:23	eb	This field indicates the type of error that caused a Bus Error, for example, error response on the AHB or AXI interface. This field is valid only when Bit 13 (FBI) is set. This field does not generate an interrupt. * Bit 23 - 1'b1: Error during data transfer by the Tx DMA - 1'b0: Error during data transfer by the Rx DMA * Bit 24 - 1'b1: Error during read transfer - 1'b0: Error during write transfer * Bit 25 - 1'b1: Error during descriptor access - 1'b0: Error during data buffer access	RO	0x0
22:20	ts	This field indicates the Transmit DMA FSM state. This field does not generate an interrupt. **Value** — **Description** 0x0 — Stopped Reset or Stop Transmit Command 0x1 — Running: Fetching Tranmit Transfer Descriptor 0x2 — Running; Waiting for status 0x3 — Running; Reading Data host memory buffer and queuing it to transmit buffer (Tx FIFO) 0x4 — TIME_STAMP write state 0x5 — Reserved for future use 0x6 — Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow 0x7 — Running; Closing Transmit Descriptor	RO	0x0

Bit	Name	Description	Access	Reset
19:17	`rs`	This field indicates the Receive DMA FSM state. This field does not generate an interrupt. **Value** **Description** 0x0 Stopped Reset or Stop Receive Command issued 0x1 Running: Fetching Receive Transfer Descriptor 0x2 Reserved for future use 0x3 Running: Waiting for receive packet 0x4 Suspended: Receive Descriptor Unavailable 0x5 Running: Closing Receive Descriptor 0x6 TIME_STAMP write state 0x7 Transferring rcv packet data from receive buffer to host memory	RO	0x0
16	`nis`	Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register): * Register 5[0]: Transmit Interrupt * Register 5[2]: Transmit Buffer Unavailable * Register 5[6]: Receive Interrupt * Register 5[14]: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in Register 7) affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.	RW	0x0
15	`ais`	Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register): * Register 5[1]: Transmit Process Stopped * Register 5[3]: Transmit Jabber Timeout * Register 5[4]: Receive FIFO Overflow * Register 5[5]: Transmit Underflow * Register 5[7]: Receive Buffer Unavailable * Register 5[8]: Receive Process Stopped * Register 5[9]: Receive Watchdog Timeout * Register 5[10]: Early Transmit Interrupt * Register 5[13]: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.	RW	0x0

Bit	Name	Description	Access	Reset
14	eri	This bit indicates that the DMA had filled the first data buffer of the packet. Bit 6 (RI) of this register automatically clears this bit.	RW	0x0
13	fbi	This bit indicates that a bus error occurred, as described in Bits[25:23]. When this bit is set, the corresponding DMA engine disables all of its bus accesses.	RW	0x0
10	eti	This bit indicates that the frame to be transmitted is fully transferred to the MTL Transmit FIFO.	RW	0x0
9	rwt	This bit is asserted when a frame with length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled).	RW	0x0
8	rps	This bit is asserted when the Receive Process enters the Stopped state.	RW	0x0
7	ru	This bit indicates that the host owns the Next Descriptor in the Receive List and the DMA cannot acquire it. The Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor is owned by the DMA.	RW	0x0
6	ri	This bit indicates that the frame reception is complete. When reception is complete, the Bit 31 of RDES1 (Disable Interrupt on Completion) is reset in the last Descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.	RW	0x0
5	unf	This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.	RW	0x0
4	ovf	This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].	RW	0x0

Bit	Name	Description	Access	Reset
3	tjt	This bit indicates that the Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when the Jumbo frame is enabled) . When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.	RW	0x0
2	tu	This bit indicates that the host owns the Next Descriptor in the Transmit List and the DMA cannot acquire it. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command.	RW	0x0
1	tps	This bit is set when the transmission is stopped.	RW	0x0
0	ti	This bit indicates that the frame transmission is complete. When transmission is complete, the Bit 31 (Interrupt on Completion) of TDES1 is reset in the first descriptor, and the specific frame status information is updated in the descriptor.	RW	0x0

Operation_Mode

The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701018
emac1	0xFF702000	0xFF703018

Offset: 0x1018

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="5">Reserved</td><td>dt
RW
0x0</td><td>rsf
RW
0x0</td><td>dff
RW
0x0</td><td colspan="2">Reserved</td><td>tsf
RW
0x0</td><td>ftf
RW
0x0</td><td colspan="3">Reserved</td><td>ttc
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">ttc
RW 0x0</td><td>st
RW
0x0</td><td colspan="2">rfd
RW 0x0</td><td colspan="2">rfa
RW 0x0</td><td>efc
RW
0x0</td><td>fef
RW
0x0</td><td>fuf
RW
0x0</td><td>Reserved</td><td colspan="2">rtc
RW 0x0</td><td>osf
RW
0x0</td><td>sr
RW
0x0</td><td>Reserved</td></tr>
</table>

Operation_Mode Fields

Bit	Name	Description	Access	Reset
26	dt	When this bit is set, the MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. When this bit is reset, all error frames are dropped if the FEF bit is reset. **Value** / **Description** 0x0 All Error Frames Dropped 0x1 MAC does not drop frame with errors	RW	0x0
25	rsf	When this bit is set, the MTL reads a frame from the Rx FIFO only after the complete frame has been written to it, ignoring the RTC bits. When this bit is reset, the Rx FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits. **Value** / **Description** 0x0 Rx Fifo cut-through mode 0x1 Read Rx FIFO only after complete frame	RW	0x0
24	dff	When this bit is set, the Rx DMA does not flush any frames because of the unavailability of receive descriptors or buffers as it does normally when this bit is reset. **Value** / **Description** 0x0 Rx DMA Flushed 0x1 Rx DMA not Flushed	RW	0x0
21	tsf	When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Bits[16:14] are ignored. This bit should be changed only when the transmission is stopped. **Value** / **Description** 0x0 Tx Does not Start with Full Frame 0x1 Tx Start with Full Frame	RW	0x0

Bit	Name	Description	Access	Reset
20	ftf	When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost or flushed. This bit is cleared internally when the flushing operation is completed. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. Note: The flush operation is complete only when the Tx FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. To complete this flush operation, the PHY transmit clock is required to be active. **Value** — **Description** 0x0 — Tx FIFO Data not Flushed 0x1 — TX FIFO Data Flushed	RW	0x0
16:14	ttc	These bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset. **Value** — **Description** 0x0 — MTL Transmit FIFO Threshold 64 0x1 — MTL Transmit FIFO Threshold 128 0x2 — MTL Transmit FIFO Threshold 192 0x3 — MTL Transmit FIFO Threshold 256 0x4 — MTL Transmit FIFO Threshold 40 0x5 — MTL Transmit FIFO Threshold 32 0x6 — MTL Transmit FIFO Threshold 24 0x7 — MTL Transmit FIFO Threshold 16	RW	0x0

Bit	Name	Description	Access	Reset
13	st	When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 4 (Transmit Descriptor List Address Register), or from the position retained when transmission was stopped previously. If the DMA does not own the current descriptor, transmission enters the Suspended state and Bit 2 (Transmit Buffer Unavailable) of Register 5 (Status Register) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Register 4 (Transmit Descriptor List Address Register), then the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program Register 4 (Transmit Descriptor List Address Register) with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state. **Value** — **Description** 0x0 — Transmission Stopped State 0x1 — Transmission in Run State	RW	0x0
12:11	rfd	These bits control the threshold (Fill-level of Rx FIFO) at which the flow control is de-asserted after activation. The de-assertion is effective only after flow control is asserted. **Value** — **Description** 0x0 — Full minus 1 KB 0x1 — Full minus 2 KB 0x2 — Full minus 3 KB 0x3 — Full minus 4 KB	RW	0x0

Bit	Name	Description	Access	Reset
10:9	`rfa`	These bits control the threshold (Fill level of Rx FIFO) at which the flow control is activated. These values only apply to the Rx FIFO when the EFC bit is set high. **Value** **Description** 0x0 Full minus 1 KB 0x1 Full minus 2 KB 0x2 Full minus 3 KB 0x3 Full minus 4 KB	`RW`	`0x0`
8	`efc`	When this bit is set, the flow control signal operation based on the fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled. **Value** **Description** 0x0 Rx FIFO Fill Level Disabled 0x1 Rx FIFO Fill Level Enabled Ctrl	`RW`	`0x0`
7	`fef`	When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the read controller side (in Threshold mode), then the frame is not dropped. When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If the Bit 25 (RSF) is set and the Rx FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the Bit 25 (RSF) is reset and the Rx FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA. **Value** **Description** 0x0 Drops Frames with error status 0x1 Forward all Frames(except runt)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
6	fuf	When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC. When reset, the Rx FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold, for example, RTC = 01. **Value** **Description** 0x0 Drops Frames less than 64Bytes 0x1 Forward Frames with no errors	RW	0x0
4:3	rtc	These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. **Value** **Description** 0x0 MTL Rcv Fifo threshold level 64 0x1 MTL Rcv Fifo threshold level 32 0x2 MTL Rcv Fifo threshold level 96 0x3 MTL Rcv Fifo threshold level 128	RW	0x0
2	osf	When this bit is set, it instructs the DMA to process the second frame of the Transmit data even before the status for the first frame is obtained. **Value** **Description** 0x0 DMA Does Not Process second frame 0x1 DMA Processes second frame	RW	0x0

Bit	Name	Description	Access	Reset
1	sr	When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by Register 3 (Receive Descriptor List Address Register) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting Register 3 (Receive Descriptor List Address Register), the DMA behavior is unpredictable. When this bit is cleared, the Rx DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state. **Value** — **Description** 0x0 — Rx DMA operation is stopped 0x1 — Rx DMA operation is started	RW	0x0

Interrupt_Enable

The Interrupt Enable register enables the interrupts reported by Register 5 (Status Register). Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70101C
emac1	0xFF702000	0xFF70301C

Offset: 0x101C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															nie RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
aie RW 0x0	ere RW 0x0	fbe RW 0x0	Reserved		ete RW 0x0	rwe RW 0x0	rse RW 0x0	rue RW 0x0	rie RW 0x0	une RW 0x0	ove RW 0x0	tje RW 0x0	tue RW 0x0	tse RW 0x0	tie RW 0x0

Interrupt_Enable Fields

Bit	Name	Description	Access	Reset
16	`nie`	When this bit is set, normal interrupt summary is enabled. When this bit is reset, normal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): * Register 5[0]: Transmit Interrupt * Register 5[2]: Transmit Buffer Unavailable * Register 5[6]: Receive Interrupt * Register 5[14]: Early Receive Interrupt **Value** **Description** 0x0 Normal Interrupt Summary Disabled 0x1 Normal Interrupt Summary Enabled	`RW`	`0x0`
15	`aie`	When this bit is set, abnormal interrupt summary is enabled. When this bit is reset, the abnormal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): * Register 5[1]: Transmit Process Stopped * Register 5[3]: Transmit Jabber Timeout * Register 5[4]: Receive Overflow * Register 5[5]: Transmit Underflow * Register 5[7]: Receive Buffer Unavailable * Register 5[8]: Receive Process Stopped * Register 5[9]: Receive Watchdog Timeout * Register 5[10]: Early Transmit Interrupt * Register 5[13]: Fatal Bus Error **Value** **Description** 0x0 Abnormal Interrupt Summary Interrupt Disabled 0x1 Abnormal Interrupt Summary Interrupt Enabled	`RW`	`0x0`
14	`ere`	When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Early Receive Interrupt is enabled. When this bit is reset, the Early Receive Interrupt is disabled. **Value** **Description** 0x0 Early Receive Interrupt Disabled 0x1 Early Receive Interrupt Enabled	`RW`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>13</td><td>fbe</td><td>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, the Fatal Bus Error Enable Interrupt is disabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Fatal Bus Error Interrupt Disabled</td></tr>
<tr><td>0x1</td><td>Fatal Bus Error Interrupt Enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>10</td><td>ete</td><td>When this bit is set with an Abnormal Interrupt Summary Enable (Bit 15), the Early Transmit Interrupt is enabled. When this bit is reset, the Early Transmit Interrupt is disabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Early Transmit Interrupt Disabled</td></tr>
<tr><td>0x1</td><td>Early Transmit Interrupt Enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>9</td><td>rwe</td><td>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout Interrupt is disabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Receive Watchdog Timeout Interrupt Disabled</td></tr>
<tr><td>0x1</td><td>Receive Watchdog Timeout Interrupt Enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>8</td><td>rse</td><td>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped Interrupt is disabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Receive Stopped Interrupt Disabled</td></tr>
<tr><td>0x1</td><td>Receive Stopped Interrupt Enabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>7</td><td>rue</td><td>When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.</td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
6	rie	When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled. **Value** **Description** 0x0 Receive Interrupt Disabled 0x1 Receive Interrupt Enabled	RW	0x0
5	une	When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Underflow Interrupt is enabled. When this bit is reset, the Underflow Interrupt is disabled. **Value** **Description** 0x0 Underflow Interrupt Disabled 0x1 Underflow Interrupt Enabled	RW	0x0
4	ove	When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Overflow Interrupt is enabled. When this bit is reset, the Overflow Interrupt is disabled. **Value** **Description** 0x0 Transmit Overflow Interrupt Disabled 0x1 Transmit Overflow Interrupt Enabled	RW	0x0
3	tje	When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, the Transmit Jabber Timeout Interrupt is disabled. **Value** **Description** 0x0 Transmit Jabber Timeout Interrupt Disabled 0x1 Transmit Jabber Timeout Interrupt Enabled	RW	0x0

Bit	Name	Description	Access	Reset
2	tue	When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable Interrupt is disabled. **Value** **Description** 0x0 Transmit Buffer Unavailable Interrupt Disabled 0x1 Transmit Buffer Unavailable Interrupt Enabled	RW	0x0
1	tse	When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmission Stopped Interrupt is enabled. When this bit is reset, the Transmission Stopped Interrupt is disabled. **Value** **Description** 0x0 Transmit Stopped Interrupt Disabled 0x1 Transmit Stopped Interrupt Enabled	RW	0x0
0	tie	When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled. **Value** **Description** 0x0 Transmit Interrupt Disabled 0x1 Transmit Interrupt Enabled	RW	0x0

Missed_Frame_And_Buffer_Overflow_Counter

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701020
emac1	0xFF702000	0xFF703020

Offset: 0x1020

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			ovfcntovf RO 0x0	ovffrmcnt RO 0x0											miscntovf RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
misfrmcnt RO 0x0															

Missed_Frame_And_Buffer_Overflow_Counter Fields

Bit	Name	Description	Access	Reset
28	ovfcntovf	Overflow bit for FIFO Overflow Counter	RO	0x0
27:17	ovffrmcnt	This field indicates the number of frames missed by the application. This counter is incremented each time the MTL asserts the sideband signal mtl_rxoverflow_o. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.	RO	0x0
16	miscntovf	Overflow bit for Missed Frame Counter	RO	0x0
15:0	misfrmcnt	This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.	RO	0x0

Receive_Interrupt_Watchdog_Timer

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of Register 5 (Status Register)

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701024
emac1	0xFF702000	0xFF703024

Offset: 0x1024

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								riwt RW 0x0							

Receive_Interrupt_Watchdog_Timer Fields

Bit	Name	Description	Access	Reset
7:0	riwt	This bit indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the Rx DMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.	RW	0x0

AXI_Bus_Mode

The AXI Bus Mode Register controls the behavior of the AXI master. It is mainly used to control the burst splitting and the number of outstanding requests.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701028
emac1	0xFF702000	0xFF703028

Offset: 0x1028

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
en_lpi RW 0x0	lpi_xit_frm RW 0x0	Reserved						wr_osr_lmt RW 0x1				rd_osr_lmt RW 0x1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		onekbbe RW 0x0	axi_aal RO 0x0	Reserved								blen16 RW 0x0	blen8 RW 0x0	blen4 RW 0x0	undefined RO 0x1

AXI_Bus_Mode Fields

Bit	Name	Description	Access	Reset
31	en_lpi	When set to 1, this bit enables the LPI mode supported by the AXI master and accepts the LPI request from the AXI System Clock controller. When set to 0, this bit disables the LPI mode and always denies the LPI request from the AXI System Clock controller. **Value** **Description** 0x0 Disable LPI Mode 0x1 Enable LPI Mode	RW	0x0
30	lpi_xit_frm	When set to 1, this bit enables the GMAC-AXI to come out of the LPI mode only when the Magic Packet or Remote Wake Up Packet is received. When set to 0, this bit enables the GMAC-AXI to come out of LPI mode when any frame is received. **Value** **Description** 0x0 Do Not exit LPI Mode with Magic Packet 0x1 Exit LPI Mode with Magic Packet	RW	0x0
23:20	wr_osr_lmt	AXI Maximum Write OutStanding Request Limit	RW	0x1
19:16	rd_osr_lmt	This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT+1	RW	0x1
13	onekbbe	1 KB Boundary Crossing Enable for the GMAC-AXI Master When set, the GMAC-AXI Master performs burst transfers that do not cross 1 KB boundary. When reset, the GMAC-AXI Master performs burst transfers that do not cross 4 KB boundary. **Value** **Description** 0x0 4K boundary 0x1 1K boundary	RW	0x0

Bit	Name	Description	Access	Reset
12	`axi_aal`	This bit is read-only bit and reflects the Bit 25 (AAL) of Register 0 (Bus Mode Register). When this bit is set to 1, the GMAC-AXI performs address-aligned burst transfers on both read and write channels. **Value** — **Description** 0x0 — No Address-Alignment Bursts 0x1 — Address-Alignmnet Bursts	RO	0x0
3	`blen16`	When this bit is set to 1 or UNDEFINED is set to 1, the GMAC-AXI is allowed to select a burst length of 16 on the AXI Master interface. **Value** — **Description** 0x0 — AXI No Fixed Busrts 0x1 — AXI Fixed Burst BLEN = 16	RW	0x0
2	`blen8`	When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 8 on the AXI Master interface. Setting this bit has no effect when UNDEFINED is set to 1. **Value** — **Description** 0x0 — AXI No Fixed Busrts 0x1 — AXI Fixed Burst BLEN = 8	RW	0x0
1	`blen4`	When this bit is set to 1, the GMAC-AXI is allowed to select a burst length of 4 on the AXI Master interface. Setting this bit has no effect when UNDEFINED is set to 1. **Value** — **Description** 0x0 — AXI No Fixed Busrts 0x1 — AXI Fixed Burst BLEN = 4	RW	0x0

Bit	Name	Description	Access	Reset
0	`undefined`	This bit is read-only bit and indicates the complement (invert) value of Bit 16 (FB) in Register 0 (Bus Mode Register[16]). * When this bit is set to 1, the GMAC-AXI is allowed to perform any burst length equal to or below the maximum allowed burst length programmed in Bits[7:1]. * When this bit is set to 0, the GMAC-AXI is allowed to perform only fixed burst lengths as indicated by BLEN16, BLEN8, or BLEN4, or a burst length of 1. **Value** / **Description** 0x0 Fixed Burst Lengths 4 to 32 0x1 Any Burst Length up to max	RO	0x1

AHB_or_AXI_Status

This register provides the active status of the AXI interface's read and write channels. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

Module Instance	Base Address	Register Address
`emac0`	`0xFF700000`	`0xFF70102C`
`emac1`	`0xFF702000`	`0xFF70302C`

Offset: `0x102C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`														`axirdsts` `RO` `0x0`	`axwhsts` `RO 0x0`

AHB_or_AXI_Status Fields

Bit	Name	Description	Access	Reset
1	`axirdsts`	When high, it indicates that AXI Master's read channel is active and transferring data.	RO	0x0
0	`axwhsts`	When high, it indicates that AXI Master's write channel is active and transferring data	RO	0x0

Current_Host_Transmit_Descriptor

The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701048
emac1	0xFF702000	0xFF703048

Offset: 0x1048

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
curtdesaptr RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
curtdesaptr RO 0x0															

Current_Host_Transmit_Descriptor Fields

Bit	Name	Description	Access	Reset
31:0	curtdesaptr	Cleared on Reset. Pointer updated by the DMA during operation.	RO	0x0

Current_Host_Receive_Descriptor

The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF70104C
emac1	0xFF702000	0xFF70304C

Offset: 0x104C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
currdesaptr RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
currdesaptr RO 0x0															

Current_Host_Receive_Descriptor Fields

Bit	Name	Description	Access	Reset
31:0	currdesaptr	Cleared on Reset. Pointer updated by the DMA during operation.	RO	0x0

Current_Host_Transmit_Buffer_Address

The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701050
emac1	0xFF702000	0xFF703050

Offset: 0x1050

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
curtbufaptr RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
curtbufaptr RO 0x0															

Current_Host_Transmit_Buffer_Address Fields

Bit	Name	Description	Access	Reset
31:0	curtbufaptr	Cleared on Reset. Pointer updated by the DMA during operation.	RO	0x0

Current_Host_Receive_Buffer_Address

The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701054
emac1	0xFF702000	0xFF703054

Offset: 0x1054

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
currbufaptr RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
currbufaptr RO 0x0															

Current_Host_Receive_Buffer_Address Fields

Bit	Name	Description	Access	Reset
31:0	currbufaptr	Cleared on Reset. Pointer updated by the DMA during operation.	RO	0x0

HW_Feature

This register indicates the presence of the optional features or functions of the gmac. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

Module Instance	Base Address	Register Address
emac0	0xFF700000	0xFF701058
emac1	0xFF702000	0xFF703058

Offset: 0x1058

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	actphyif RO 0x0			Reserved			enhdessel RO 0x1	txchcnt RO 0x0		rxchcnt RO 0x0		rxfifosize RO 0x1	rxtyp2coe RO 0x1	rxtyp1coe RO 0x0	txoesel RO 0x1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
avsel RO 0x0	eeesel RO 0x1	tsver2sel RO 0x1	tsver1sel RO 0x1	mmcsel RO 0x1	mgksel RO 0x1	rwksel RO 0x1	smasel RO 0x1	Reserved	pcssel RO 0x0	addmacadrsel RO 0x1	hashsel RO 0x1	Reserved	hdsel RO 0x1	gmiisel RO 0x1	miisel RO 0x1

HW_Feature Fields

Bit	Name	Description	Access	Reset
30:28	actphyif	When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of emacx_phy_if_selduring reset de-assertion. **Value** — **Description** 0x0 — Sampled Value GMII or MII 0x1 — Sampled Value RGMII 0x2 — Sampled Value SGMII 0x3 — Sampled Value TBI 0x4 — Sampled Value RMII 0x5 — Sampled Value RTBI 0x6 — Sampled Value SMII 0x7 — Sampled Value RevMII	RO	0x0
24	enhdessel	Alternate (Enhanced Descriptor) **Value** — **Description** 0x0 — Enhanced Descriptor Select disabled 0x1 — Enhanced Descriptor Select enabled	RO	0x1
23:22	txchcnt	Number of additional Tx channels **Value** — **Description** 0x0 — Tx Channel Count disabled 0x1 — Tx Channel Count enabled	RO	0x0
21:20	rxchcnt	Number of additional Rx channels **Value** — **Description** 0x0 — Rx Channel Count disabled 0x1 — Rx Channel Count enabled	RO	0x0
19	rxfifosize	RxFIFO > 2048 Bytes **Value** — **Description** 0x0 — RxFIFO > 2048 bytes disabled 0x1 — RxFIFO > 2048 bytes enabled	RO	0x1

Bit	Name	Description	Access	Reset
18	`rxtyp2coe`	IP Checksum Offload (Type 2) in Rx **Value** **Description** 0x0 Rx Type 2 Checksum Offload disabled 0x1 Rx Type 2 Checksum Offload enabled	RO	0x1
17	`rxtyp1coe`	IP Checksum Offload (Type 1) in Rx **Value** **Description** 0x0 Rx Type 1 Checksum Offload disabled 0x1 Rx Type 1 Checksum Offload enabled	RO	0x0
16	`txoesel`	Checksum Offload in Tx **Value** **Description** 0x0 Tx Offload Checksum disabled 0x1 Tx Offload Checksum enabled	RO	0x1
15	`avsel`	AV Feature **Value** **Description** 0x0 AV Select disabled 0x1 AV Select enabled	RO	0x0
14	`eeesel`	Energy Efficient Ethernet Feature **Value** **Description** 0x0 Energy Efficient Ethernet disabled 0x1 Energy Efficient Ethernet enabled	RO	0x1
13	`tsver2sel`	IEEE 1588-2008 Advanced Timestamp **Value** **Description** 0x0 TS Version2 Select disabled 0x1 TS Version2 Select enabled	RO	0x1

Bit	Name	Description	Access	Reset
12	`tsver1sel`	Only IEEE 1588-2002 Timestamp **Value** **Description** 0x0 TS Version1 Select disabled 0x1 TS Version1 Select enabled	RO	0x1
11	`mmcsel`	RMON block **Value** **Description** 0x0 Rmon block disabled 0x1 Rmon block enabled	RO	0x1
10	`mgksel`	PMT Magic Packet **Value** **Description** 0x0 PMT Magic Packet disabled 0x1 PMT Magic Packet enabled	RO	0x1
9	`rwksel`	PMT Remote Wakeup support **Value** **Description** 0x0 PMT Remote Wake Up disabled 0x1 PMT Remote Wake Up enabled	RO	0x1
8	`smasel`	SMA (MDIO) Interface support **Value** **Description** 0x0 SMA Interface Support disabled 0x1 SMA Interface Support enabled	RO	0x1
6	`pcssel`	TBI/SGMII/RTBI PHY interface support **Value** **Description** 0x0 PCS Support disabled 0x1 PCS Support enabled	RO	0x0

Bit	Name	Description	Access	Reset
5	addmacadrsel	Multiple MAC Address Registers support **Value** **Description** 0x0 Multiple MAC Address registers disabled 0x1 Multiple MAC Address registers enabled	RO	0x1
4	hashsel	HASH Filter support **Value** **Description** 0x0 Hash Filter disabled 0x1 Hash Filter enabled	RO	0x1
2	hdsel	Half-Duplex support **Value** **Description** 0x0 Half Duplex disabled 0x1 Half Duplex enabled	RO	0x1
1	gmiisel	1000 Mbps support **Value** **Description** 0x0 1000 Mbps disabled 0x1 1000 Mbps enabled	RO	0x1
0	miisel	10/100 Mbps support **Value** **Description** 0x0 10 Mbps or 100 Mbps disabled 0x1 10 Mbps or 100 Mbps enabled	RO	0x1

Document Revision History

Table 17-21: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Updated EMAC to RGMII Interface table with EMAC Port names Updated EMAC to FPGA PHY Interface table with Signal names Updated EMAC to FPGA IEEE1588 Timestamp Interface with Signal names Added Address Map and Register Descriptions
February 2014	2014.02.28	ECC updates.
December 2013	1.4	Maintenance release.
November 2012	1.3	• Expanded shared memory block table. • Added CSEL tables. • Additional minor updates.
June 2012	1.2	Updated the HPS boot and FPGA configuration sections.

USB 2.0 OTG Controller 18

2014.07.31

cv_54018 Subscribe Send Feedback

The hard processor system (HPS) provides two instances of a USB On-The-Go (OTG) controller that supports both device and host functions. The controller supports all high-speed, full-speed, and low-speed transfers in both device and host modes. The controller is fully compliant with the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*. The controller can be programmed for both device and host functions to support data movement over the USB protocol.

The controllers are operationally independent of each other. Each USB OTG controller supports a single USB port connected through a USB 2.0 Transceiver Macrocell Interface Plus (UTMI+) Low Pin Interface (ULPI) compliant PHY. The USB OTG controllers are instances of the Synopsys® [†]DesignWare® Cores USB 2.0 Hi-Speed On-The-Go (DWC_otg) controller.

The USB OTG controller is optimized for the following applications and systems: †

- Portable electronic devices †
- Point-to-point applications (no hub, direct connection to HS, FS, or LS device) †
- Multi-point applications (as an embedded USB host) to devices (hub and split support) †

Each of the two USB OTG ports supports both host and device modes, as described in the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*. The USB OTG ports support connections for all types of USB peripherals, including the following peripherals:

- Mouse
- Keyboard
- Digital cameras
- Network adapters
- Hard drives
- Generic hubs

[†] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non infringement, and any warranties arising out of a course of dealing or usage of trade.

† Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

Related Information

http://www.usb.org/home
Additional information is available in the On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification, which you can download from the USB Implementers Forum website

Features of the USB OTG Controller

The USB OTG controller has the following USB-specific features:

- Complies with both Revision 1.3 and Revision 2.0 of the *On The Go and Embedded Host Supplement to the USB Revision 2.0 Specification*
- Supports software-configurable modes of operation between OTG 1.3 and OTG 2.0
- Supports all USB 2.0 speeds:
  - High speed (HS, 480-Mbps)
  - Full speed (FS, 12-Mbps)
  - Low speed (LS, 1.5-Mbps)

  Note: In host mode, all speeds are supported. However, in device mode, only high speed and full speed are supported.
- Supports all USB transaction types:
  - Control transfers
  - Bulk transfers
  - Isochronous transfers
  - Interrupts
- Supports automatic ping capability
- Supports Session Request Protocol (SRP) and Host Negotiation Protocol (HNP)
- Supports suspend, resume, and remote wake
- Supports up to 16 host channels

  Note: In host mode, when the number of device endpoints is greater than the number of host channels, software can reprogram the channels to support up to 127 devices, each having 32 endpoints (IN + OUT), for a maximum of 4,064 endpoints.
- Supports up to 16 bidirectional endpoints, including control endpoint 0

  Note: Only seven periodic device IN endpoints are supported.
- Supports a generic root hub
- Performs transaction scheduling in hardware

On the USB PHY layer, the USB OTG controller supports the following features:

- A single USB port connected to each OTG instance
- A ULPI connection to an off-chip USB transceiver
- Software-controlled access, supporting vendor-specific or optional PHY registers access to ease debug
- The OTG 2.0 support for Attach Detection Protocol (ADP) only through an external (off-chip) ADP controller

On the integration side, the USB OTG controller supports the following features:

- Different clocks for system and PHY interfaces
- Dedicated TX FIFO buffer for each device IN endpoint in direct memory access (DMA) mode
- Packet-based, dynamic FIFO memory allocation for endpoints for small FIFO buffers and flexible, efficient use of RAM that can be dynamically sized by software
- Ability to change an endpoint's FIFO memory size during transfers
- Clock gating support during USB suspend and session-off modes
    - PHY clock gating support
    - System clock gating support
- Data FIFO RAM clock gating support
- Local buffering with error correction code (ECC) support

Note: The USB OTG controller does not support the following protocols:

- Enhanced Host Controller Interface (EHCI)
- Open Host Controller Interface (OHCI)
- Universal Host Controller Interface (UHCI)

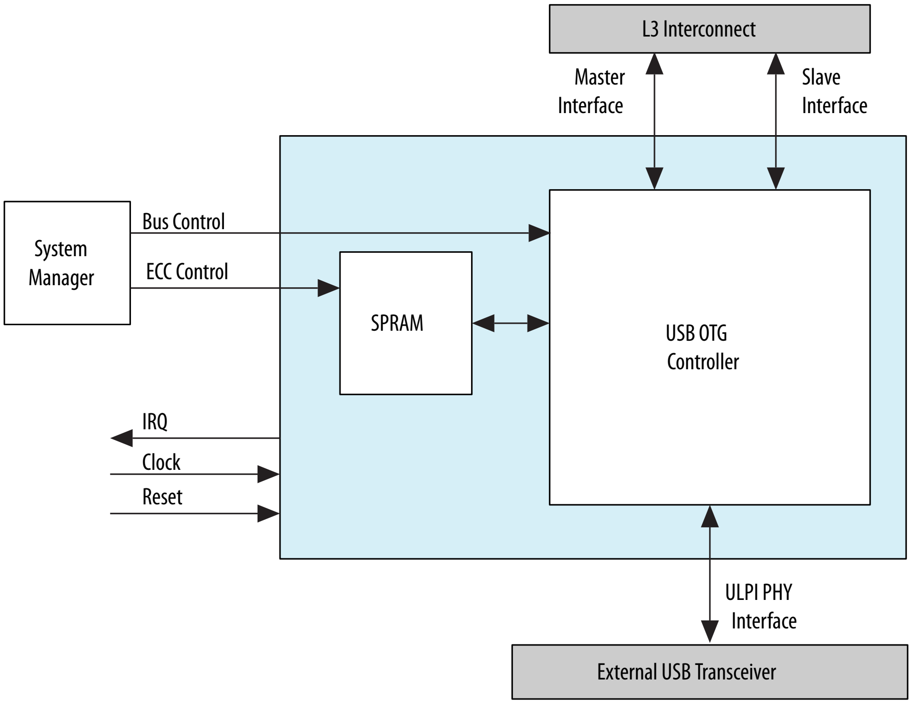
Supported PHYS

The USB OTG controller supports USB 2.0 ULPI PHYs. Only the single data rate (SDR) mode is supported. Refer to the *Cyclone V Device Datasheet* for timing characteristics on the USB PHY interface. The Altera *Cyclone V* SoC Development Board currently uses a SMSC USB3300 PHY.

USB OTG Controller Block Diagram and System Integration

Figure 18-1: USB OTG Controller System Integration

Two subsystems are included in the HPS.



The USB OTG controller connects to the level 3 (L3) interconnect through a slave interface, allowing other masters to access the control and status registers (CSRs) in the controller. The controller also connects to the L3 interconnect through a master interface, allowing the DMA engine in the controller to move data between external memory and the controller.

A single-port RAM (SPRAM) connected to the USB OTG controller is used to store USB data packets for both host and device modes. It is configured as FIFO buffers for receive and transmit data packets on the USB link.

Through the system manager, the USB OTG controller has control to use and test error correction codes (ECCs) in the SPRAM. Through the system manager, the USB OTG controller can also control the behavior of the master interface to the L3 interconnect.

The USB OTG controller connects to the external USB transceiver through a ULPI PHY interface. This interface also connects through pin multiplexers within the HPS. The pin multiplexers are controlled by the system manager.

Additional connections on the USB OTG controller include:

- Clock input from the clock manager to the USB OTG controller
- Reset input from the reset manager to the USB OTG controller
- Interrupt line from the USB OTG controller to the microprocessor unit (MPU) global interrupt controller (GIC).

Related Information

System Manager

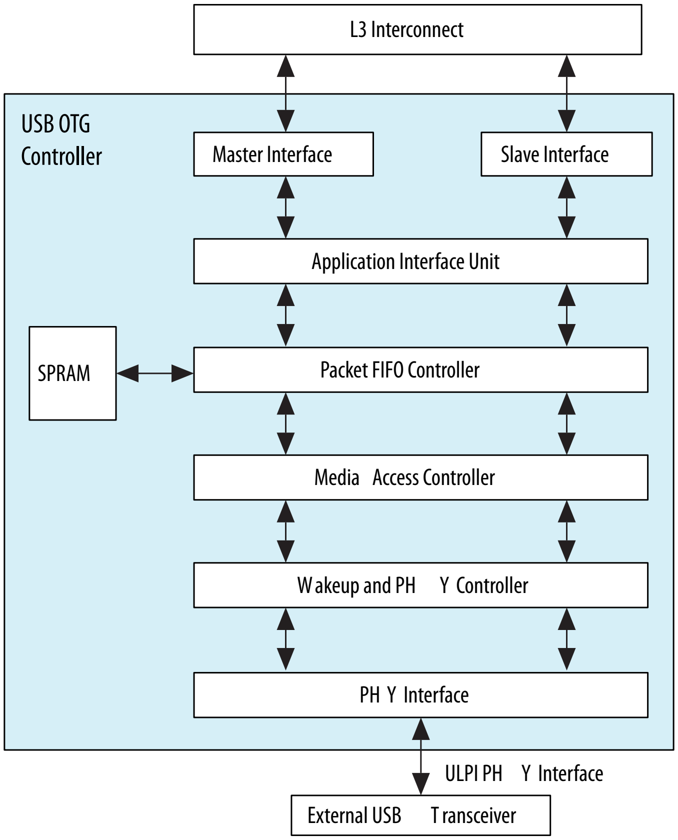
Details available in the System Manager chapter.

Functional Description of the USB OTG Controller

USB OTG Controller Block Description

Figure 18-2: USB OTG Controller Block Description

Details about each of the units that comprise the USB OTG controller are shown below.



Master Interface

The master interface includes a built-in DMA controller. The DMA controller moves data between external memory and the media access controller (MAC).

Properties of the master interface are controlled through the USB L3 Master HPROT Register (`l3master`) in the system manager. These bits provide access information to the L3 interconnect, including whether or not transactions are cacheable, bufferable, or privileged.

Note: Bits in the `l3master` register can be updated only when the master interface is guaranteed to be in an inactive state.

Slave Interface

The slave interface allows other masters in the system to access the USB OTG controller's CSRs. For testing purposes, other masters can also access the SPRAM.

Slave Interface CSR Unit

The slave interface can read from and write to all the CSRs in the USB OTG controllers. All register accesses are 32 bits.

The CSR is divided into the following groups of registers:

- Global
- Host
- Device
- Power and clock gating

Some registers are shared between host and device modes, because the controller can only be in one mode at a time. The controller generates a mode mismatch interrupt if a master attempts to access device registers when the controller is in host mode, or attempts to access host registers when the controller is in device mode. Writing to unimplemented registers is ignored. Reading from unimplemented registers returns indeterminate values.

Application Interface Unit

The application interface unit (AIU) generates DMA requests based on programmable FIFO buffer thresholds. The AIU generates interrupts to the GIC for both host and device modes. A DMA scheduler is included in the AIU to arbitrate and control the data transfer between packets in system memory and their respective USB endpoints.

Packet FIFO Controller

The Packet FIFO Controller (PFC) connects the AIU with the MAC through data FIFO buffers located in the SPRAM. In device mode, one FIFO buffer is implemented for each IN endpoint. In host mode, a single FIFO buffer stores data for all periodic (isochronous and interrupt) OUT endpoints, and a single FIFO buffer is used for nonperiodic (control and bulk) OUT endpoints. Host and device mode share a single receive data FIFO buffer.

SPRAM

An SPRAM implements the data FIFO buffers for host and device modes. The size of the FIFO buffers can be programmed dynamically.

The SPRAM supports ECCs. ECCs can be enabled through the system manager, by setting the RAM ECC Enable (`en`) bit in the USB0 or USB1 RAM ECC Enable Register (`usb0` or `usb1`), in the ECC Management Register Group (`eccgrp`). Single-bit and double-bit errors in each USB instance can be injected using this register.

The SPRAM provides outputs to notify the system manager when single-bit correctable errors are detected (and corrected), and when double-bit (uncorrectable) errors are detected. The system manager generates an interrupt to the GIC when an ECC error is detected.

MAC

The MAC module implements the following functionality:

- USB transaction support
- Host protocol support
- Device protocol support
- OTG protocol support
- Link power management (LPM) functions

USB Transactions

In device mode, the MAC decodes and checks the integrity of all token packets. For valid OUT or SETUP tokens, the following DATA packet is also checked. If the data packet is valid, the MAC performs the following steps:

1. Writes the data to the receive FIFO buffer
2. Sends the appropriate handshake when required to the USB host

If a receive FIFO buffer is not available, the MAC sends a NAK response to the host. The MAC also supports ping protocol.

For IN tokens, if data is available in the transmit FIFO buffer, the MAC performs the following steps:

1. Reads the data from the FIFO buffer
2. Forms the data packet
3. Transmits the packet to the host
4. Receives the response from the host
5. Sends the updated status to the PFC

In host mode, the MAC receives a token request from the AIU. The MAC performs the following steps:

1. Builds the token packet
2. Sends the packet to the device

For OUT or SETUP transactions, the MAC also performs the following steps:

1. Reads the data from the transmit FIFO buffer
2. Assembles the data packet
3. Sends the packet to the device
4. Waits for a response

The response from the device causes the MAC to send a status update to the AIU.

For IN or PING transactions, the MAC waits for the data or handshake response from the device. For data responses, the MAC performs the following steps:

1. Validates the data
2. Writes the data to the receive FIFO buffer
3. Sends a status update to the AIU
4. Sends a handshake to the device, if appropriate

Host Protocol

In host mode, the MAC performs the following functions:

- Detects connect, disconnect, and remote wakeup events on the USB link
- Initiates reset
- Initiates speed enumeration processes
- Generates Start of Frame (SOF) packets.

Device Protocol

In device mode, the MAC performs the following functions:

- Handles USB reset sequence
- Handles speed enumeration
- Detects USB suspend and resume activity on the USB link
- Initiates remote wakeup
- Decodes SOF packets

OTG Protocol

The MAC handles HNP and SRP for OTG operation. HNP provides a mechanism for swapping host and device roles. SRP provides mechanisms for the host to turn off V_{BUS} to save power, and for a device to request a new USB session.

LPM Functions

The USB OTG controller supports LPM in both host and device modes. With this feature, the USB OTG controller can enter a sleep state when a successful LPM transaction occurs on the USB link.

Wakeup and Power Control

To reduce power, the USB OTG controller supports a power-down mode. In power-down mode, the controller and the PHY can shut down their clocks. The controller supports wakeup on the detection of the following events:

- Resume
- Remote wakeup
- Session request protocol
- New session start

PHY Interface Unit

The USB OTG controller supports synchronous SDR data transmission to a ULPI PHY. The SDR mode implements an eight-bit data bus.

ULPI PHY Interface

Table 18-1: ULPI PHY Interfaces

The ULPI PHY interface is synchronous to the `ulpi_clk` signal coming from the PHY.

Port Name	Bit Width	Direction	Description
`ulpi_clk`	1	Input	ULPI Clock Receives the 60-MHz clock supplied by the high-speed ULPI PHY. All signals are synchronous to the positive edge of the clock.
`ulpi_dir`	1	Input	ULPI Data Bus Control 1—The PHY has data to transfer to the USB OTG controller. 0—The PHY does not have data to transfer.
`ulpi_nxt`	1	Input	ULPI Next Data Control Indicates that the PHY has accepted the current byte from the USB OTG controller. When the PHY is transmitting, this signal indicates that a new byte is available for the controller.
`ulpi_stp`	1	Output	ULPI Stop Data Control The controller drives this signal high to indicate the end of its data stream. The controller can also drive this signal high to request data from the PHY.
`ulpi_data[7:0]`	8	Bidirectional	Bidirectional data bus. Driven low by the controller during idle.

Clocks

Table 18-2: USB OTG Controller Clock Inputs

All clocks must be operational when reset is released. No special handling is required on the clocks.

Clock Signal	Frequency	Functional Usage
`usb_mp_clk`	60 – 200 MHz	Drives the master and slave interfaces, DMA controller, and internal FIFO buffers
`usb0_ulpi_clk`	60 MHz	ULPI reference clock for usb0 from external ULPI PHY I/O pin
`usb1_ulpi_clk`	60 MHz	ULPI reference clock for usb1 from external ULPI PHY I/O pin

Resets

The USB OTG controller can be reset either through the hardware reset input or through software.

Reset Requirements

There must be a minimum of 12 cycles on the `ulpi_clk` clock before the controller is taken out of reset. During reset, the USB OTG controller asserts the `ulpi_stp` signal. The PHY outputs a clock when it sees the `ulpi_stp` signal asserted. However, if the pin multiplexers are not programmed, the PHY does not see the `ulpi_stp` signal. As a result, the `ulpi_clk` clock signal does not arrive at the USB OTG controller.

Software must ensure that the reset is active for a minimum of two `usb_mp_c lk` cycles. There is no maximum assertion time.

Hardware Reset

Each of the USB OTG controllers has one reset input from the reset manager. The reset signal is asserted during a cold or warm reset event. The reset manager holds the controllers in reset until software releases the resets. Software releases resets by clearing the appropriate USB bits in the Peripheral Module Reset Register (`permodrst`) in the HPS reset manager.

The reset input resets the following blocks:

- The master and slave interface logic
- The integrated DMA controller
- The internal FIFO buffers
- The CSR

The reset input is synchronized to the `usb_mp_clk` domain. The reset input is also synchronized to the ULPI clock within the USB OTG controller and is used to reset the ULPI PHY domain logic.

Software Reset

Software can reset the controller by setting the Core Soft Reset (`csftrst`) bit in the Reset Register (`grstctl`) in the Global Registers (`globgrp`) group of the USB OTG controller.

Software resets are useful in the following situations:

- A PHY selection bit is changed by software. Resetting the USB OTG controller is part of clean-up to ensure that the PHY can operate with the new configuration or clock.
- During software development and debugging.

Interrupts

Table 18-3: USB OTG Interrupt Conditions

Each USB OTG controller has a single interrupt output. Interrupts are asserted on the conditions shown in the following table.

Condition	Mode
Device-initiated remote wakeup is detected.	Host mode
Session request is detected from the device.	Host mode

Condition	Mode
Device disconnect is detected.	Host mode
Host LPM entry retry has expired or LPM transaction(s) are complete.	Host mode
Host periodic TX FIFO buffer is empty (can be further programmed to indicate half-empty).	Host mode
Host channels interrupt received.	Host mode
Incomplete periodic transfer is pending at the end of the microframe.	Host mode
Host port status interrupt received.	Host mode
External host initiated resume is detected.	Device mode
LPM handshake is sent.	Device mode
Reset is detected when in suspend or normal mode.	Device mode
USB suspend mode is detected.	Device mode
Data fetch is suspended due to TX FIFO buffer full or request queue full.	Device mode
At least one isochronous OUT endpoint is pending at the end of the microframe.	Device mode
At least one isochronous IN endpoint is pending at the end of the microframe.	Device mode
At least one IN or OUT endpoint interrupt is pending at the end of the microframe.	Device mode
The end of the periodic frame is reached.	Device mode
Failure to write an isochronous OUT packet to the RX FIFO buffer. The RX FIFO buffer does not have enough space to accommodate the maximum packet size for the isochronous OUT endpoint.	Device mode
Enumeration has completed.	Device mode
Connector ID change.	Common modes
Mode mismatch. Software accesses registers belonging to an incorrect mode.	Common modes
Nonperiodic TX FIFO buffer is empty.	Common modes

Condition	Mode
RX FIFO buffer is not empty.	Common modes
Start of microframe.	Common modes
Device connection debounce is complete in host mode.	OTG interrupts
A-Device timeout while waiting for B-Device connection.	OTG interrupts
Host negotiation is complete.	OTG interrupts
Session request is complete.	OTG interrupts
Session end is detected in device mode.	OTG interrupts

USB OTG Controller Programming Model

For detailed information about using the USB OTG controller, consult your operating system (OS) driver documentation. The OS vendor provides application programming interfaces (APIs) to control USB host, device and OTG operation. This section provides a brief overview of the following software operations:

- Enabling SPRAM ECCs
- Host operation
- Device operation

Enabling SPRAM ECCs

To avoid false ECC errors, you must initialize the ECC bits in the SPRAM before using ECCs. To initialize the ECC bits, software writes data to all locations in the SPRAM.

The L3 interconnect has access to the SPRAM and is accessible through the USB OTG L3 slave interface. Software accesses the SPRAM through the `directfifo` memory space, in the USB OTG controller address space.

The SPRAM contains 8192 (32 KB) locations. The L3 slave provides 32-bit access to the SPRAM. Physically, the SPRAM is implemented as a 35-bit memory, with the highest three bits reserved for the USB OTG controller's internal use. When a write is performed to the SPRAM through the L3 slave interface, bits 32 through 34 of the internal data bus are tied to 1, to enable the ECC bits to be initialized.

Note: Software cannot access the SPRAM beyond the 32-KB range. Out-of-range read transactions return indeterminate data. Out-of-range write transactions are ignored.

Related Information

USB OTG Controller Address Map and Register Definitions on page 18-15
The directfifo memory space is described in the controller address map.

Host Operation

Host Initialization

After power up, the USB port is in its default mode. No VBUS is applied to the USB cable. The following process sets up the USB OTG controller as a USB host.

1. To enable power to the USB port, the software driver sets the Port Power (`prtpwr`) bit to 1 in the Host Port Control and Status Register (`hprt`) of the Host Mode Registers (`hostgrp`) group. This action drives the V_{BUS} signal on the USB link.

   The controller waits for a connection to be detected on the USB link.
2. When a USB device connects, an interrupt is generated. The Port Connect Detected (`PrtConnDe t`) bit in `hprt` is set to 1.
3. Upon detecting a port connection, the software driver initiates a port reset by setting the Port Reset (`prtrst`) bit to 1 in `hprt`.
4. The software driver must wait a minimum of 10 ms so that speed enumeration can complete on the USB link.
5. After the 10 ms, the software driver sets `prtrst` back to 0 to release the port reset.
6. The USB OTG controller generates an interrupt. The Port Enable Disable Change (`prtenchng`) and Port Speed (`prtspd`) bits, in `hprt`, are set to reflect the enumerated speed of the device that attached.

   At this point the port is enabled for communication. Keep alive or SOF packets are sent on the port. If a USB 2.0-capable device fails to initialize correctly, it is reported as a USB 1.1 device.

   The Host Frame Interval Register (`hfir`) is updated with the corresponding PHY clock settings. The `hfir`, used for sending SOF packets, is in the Host Mode Registers (`host grp`) group.
7. The software driver must program the following registers in the Global Registers (`globgrp`) group, in the order listed:

   a. Receive FIFO Size Register (`grxfsiz`)—selects the size of the receive FIFO buffer
   b. Non-periodic Transmit FIFO Size Register (`gnptxfsiz`)—selects the size and the start address of the non-periodic transmit FIFO buffer for nonperiodic transactions
   c. Host Periodic Transmit FIFO Size Register (`hptxfsiz`)—selects the size and start address of the periodic transmit FIFO buffer for periodic transactions
8. System software initializes and enables at least one channel to communicate with the USB device.

Host Transaction

When configured as a host, the USB OTG controller pipes the USB transactions through one of two request queues (one for periodic transactions and one for nonperiodic transactions). Each entry in the request queue holds the SETUP, IN, or OUT channel number along with other information required to perform a transaction on the USB link. The sequence in which the requests are written to the queue determines the sequence of transactions on the USB link.

The host processes the requests in the following order at the beginning of each frame or microframe:

1. Periodic request queue, including isochronous and interrupt transactions
2. Nonperiodic request queue (bulk or control transfers)

The host schedules transactions for each enabled channel in round-robin fashion. When the host controller completes the transfer for a channel, the controller updates the DMA descriptor status in the system memory.

For OUT transactions, the host controller uses two transmit FIFO buffers to hold the packet payload to be transmitted. One transmit FIFO buffer is used for all nonperiodic OUT transactions and the other is used for all periodic OUT transactions.

For IN transactions, the USB host controller uses one receive FIFO buffer for all periodic and nonperiodic transactions. The controller holds the packet payload from the USB device in the receive FIFO buffer until the packet is transferred to the system memory. The receive FIFO buffer also holds the status of each packet received. The status entry holds the IN channel number along with other information, including received byte count and validity status.

For generic hub operations, the USB OTG controller uses SPLIT transfers to communicate with slower-speed devices downstream of the hub. For these transfers, the transaction accumulation or buffering is performed in the generic hub, and is scheduled accordingly. The USB OTG controller ensures that enough transmit and receive buffers are allocated when the downstream transactions are completed or when accumulated data is ready to be sent upstream.

Device Operation

Device Initialization

The following process sets up the USB OTG controller as a USB device:

1. After power up, the USB OTG controller must be set to the desired device speed by writing to the Device Speed (`devspd`) bits in the Device Configuration Register (`dcfg`) in the Device Mode Registers (`devgrp`) group. After the device speed is set, the controller waits for a USB host to detect the USB port as a device port.
2. When an external host detects the USB port, the host performs a port reset, which generates an interrupt to the USB device software. The USB Reset (`usbrst`) bit in the Interrupt (`port reset`) register in the Global Registers (`globgrp`) group is set. The device software then sets up the data FIFO buffer to receive a SETUP packet from the external host. Endpoint 0 is not enabled yet.
3. After completion of the port reset, the operation speed required by the external host is known. Software reads the device speed status and sets up all the remaining required transaction fields to enable control endpoint 0.

After completion of this process, the device is receiving SOF packets, and is ready for the USB host to set up the device’s control endpoint.

Device Transaction

When configured as a device, the USB OTG controller uses a single FIFO buffer to receive the data for all the OUT endpoints. The receive FIFO buffer holds the status of the received data packet, including the byte count, the data packet ID (PID), and the validity of the received data. The DMA controller reads the data out of the FIFO buffer as the data are received. If a FIFO buffer overflow condition occurs, the controller responds to the OUT packet with a NAK, and internally rewinds the pointers.

For IN endpoints, the controller uses dedicated transmit buffers for each endpoint. The application does not need to predict the order in which the USB host will access the nonperiodic endpoints. If a FIFO buffer underrun condition occurs during transmit, the controller inverts the cyclic redundancy code (CRC) to mark the packet as corrupt on the USB link.

The application handles one data packet at a time per endpoint in transaction-level operations. The software receives an interrupt on completion of every packet. Based on the handshake response received on the USB link, the application determines whether to retry the transaction or proceed with the next transaction, until all packets in the transfer are completed.

IN Transactions

For an IN transaction, the application performs the following steps:

1. Enables the endpoint
2. Triggers the DMA engine to write the associated data packet to the corresponding transmit FIFO buffer
3. Waits for the packet completion interrupt from the controller

When an IN token is received on an endpoint when the associated transmit FIFO buffer does not contain sufficient data, the controller performs the following steps:

1. Generates an interrupt
2. Returns a NAK handshake to the USB host

If sufficient data is available, the controller transmits the data to the USB host.

OUT Transactions

For an OUT transaction, the application performs the following steps:

1. Enables the endpoint
2. Waits for the packet received interrupt from the USB OTG controller
3. Retrieves the packet from the receive FIFO buffer

When an OUT token or PING token is received on an endpoint where the receive FIFO buffer does not have sufficient space, the controller performs the following steps:

1. Generates an interrupt
2. Returns a NAK handshake to USB host

If sufficient space is available, the controller stores the data in the receive FIFO buffer and returns an ACK handshake to the USB link.

Control Transfers

For control transfers, the application performs the following steps:

1. Waits for the packet received interrupt from the controller
2. Retrieves the packet from the receive buffer

Because the control transfer is governed by USB protocol, the controller always responds with an ACK handshake.

USB OTG Controller Address Map and Register Definitions

The address map and register definitions for the USB OTG Controller consist of the following regions:

- USB OTG Controller Module 0 Registers
- USB OTG Controller Module 1 Registers

USB OTG Controller Module Registers Address Map on page 18-16
Registers in the USB OTG Controller Module. Only the Core Global, Power and Clock Gating, Data FIFO Access, and Host Port registers can be accessedin both Host and Device modes. When the USB OTG Controller is operating in one mode, either Device or Host, the application must not access registers from the other mode. If an illegal access occurs, a Mode Mismatch interrupt is generated and reflected in the Core Interrupt register (GINTSTS.ModeMis). When the core switches from one mode to another, the registers in the new mode must be reprogrammed as they would be after a power-on reset. The register address map is fixed and does not depend on the module configuration (for example, how many endpoints are implemented). Host and Device mode registers occupy different addresses.

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

USB OTG Controller Module Registers Address Map

Registers in the USB OTG Controller Module. Only the Core Global, Power and Clock Gating, Data FIFO Access, and Host Port registers can be accessedin both Host and Device modes. When the USB OTG Controller is operating in one mode, either Device or Host, the application must not access registers from the other mode. If an illegal access occurs, a Mode Mismatch interrupt is generated and reflected in the Core Interrupt register (GINTSTS.ModeMis). When the core switches from one mode to another, the registers in the new mode must be reprogrammed as they would be after a power-on reset. The register address map is fixed and does not depend on the module configuration (for example, how many endpoints are implemented). Host and Device mode registers occupy different addresses.

Module Instance	Base Address
usb0	0xFFB00000
usb1	0xFFB40000

Global Registers

Register	Offset	Width	Access	Reset Value	Description
gotgctl on page 18-39	0x0	32	RW	0x10000	OTG Control and Status Register
gotgint on page 18-45	0x4	32	RO	0x0	OTG Interrupt Register
gahbcfg on page 18-47	0x8	32	RW	0x0	AHB Configuration Register
gusbcfg on page 18-51	0xC	32	RW	0x1410	USB Configuration Registe
grstctl on page 18-56	0x10	32	RW	0x80000000	Reset Register
gintsts on page 18-60	0x14	32	RO	0x14000000	Interrupt Register
gintmsk on page 18-69	0x18	32	RW	0x0	Interrupt Mask Register
grxstsr on page 18-74	0x1C	32	RO	0x0	Receive Status Debug Read Register
grxstsp on page 18-75	0x20	32	RO	0x0	Receive Status Read Pop Register
grxfsiz on page 18-77	0x24	32	RW	0x2000	Receive FIFO Size Register

Register	Offset	Width	Access	Reset Value	Description
gnptxfsiz on page 18-78	0x28	32	RW	0x20002000	Non-periodic Transmit FIFO Size Register
gnptxsts on page 18-78	0x2C	32	RO	0x80400	Non-periodic Transmit FIFO Queue Status Register
gpvndctl on page 18-80	0x34	32	RW	0x0	PHY Vendor Control Register
ggpio on page 18-81	0x38	32	RW	0x0	General Purpose Input Output Register
guid on page 18-82	0x3C	32	RW	0x12345678	User ID Register
gsnpsid on page 18-83	0x40	32	RO	0x4F54293A	Synopsys ID Register
ghwcfg1 on page 18-83	0x44	32	RO	0x0	User HW Config1 Register
ghwcfg2 on page 18-84	0x48	32	RO	0x208FFC90	User HW Config2 Register
ghwcfg3 on page 18-88	0x4C	32	RO	0x1F8002E8	User HW Config3 Register
ghwcfg4 on page 18-90	0x50	32	RO	0xFE0F0020	User HW Config4 Register
gdfifocfg on page 18-95	0x5C	32	RW	0x1F802000	DFIFO Software Config Register
hptxfsiz on page 18-95	0x100	32	RW	0x20004000	Host Periodic Transmit FIFO Size Register
dieptxf1 on page 18-96	0x104	32	RW	0x20004000	Device IN Endpoint Transmit FIFO Size Register 1
dieptxf2 on page 18-97	0x108	32	RW	0x20006000	Device IN Endpoint Transmit FIFO Size Register 2
dieptxf3 on page 18-98	0x10C	32	RW	0x20008000	Device IN Endpoint Transmit FIFO Size Register 3
dieptxf4 on page 18-98	0x110	32	RW	0x2000A000	Device IN Endpoint Transmit FIFO Size Register 4
dieptxf5 on page 18-99	0x114	32	RW	0x2000C000	Device IN Endpoint Transmit FIFO Size Register 5
dieptxf6 on page 18-100	0x118	32	RW	0x2000E000	Device IN Endpoint Transmit FIFO Size Register 6
dieptxf7 on page 18-101	0x11C	32	RW	0x20000000	Device IN Endpoint Transmit FIFO Size Register 7
dieptxf8 on page 18-101	0x120	32	RW	0x20002000	Device IN Endpoint Transmit FIFO Size Register 8
dieptxf9 on page 18-102	0x124	32	RW	0x20004000	Device IN Endpoint Transmit FIFO Size Register 9
dieptxf10 on page 18-103	0x128	32	RW	0x20006000	Device IN Endpoint Transmit FIFO Size Register 10

Register	Offset	Width	Acces s	Reset Value	Description
dieptxf11 on page 18-104	0x12C	32	RW	0x20008000	Device IN Endpoint Transmit FIFO Size Register 11
dieptxf12 on page 18-104	0x130	32	RW	0x2000A000	Device IN Endpoint Transmit FIFO Size Register 12
dieptxf13 on page 18-105	0x134	32	RW	0x2000C000	Device IN Endpoint Transmit FIFO Size Register 13
dieptxf14 on page 18-106	0x138	32	RW	0x2000E000	Device IN Endpoint Transmit FIFO Size Register 14
dieptxf15 on page 18-107	0x13C	32	RW	0x20000000	Device IN Endpoint Transmit FIFO Size Register 15

Host Mode Registers

Register	Offset	Width	Acces s	Reset Value	Description
hcfg on page 18-117	0x400	32	RW	0x200	Host Configuration Register
hfir on page 18-120	0x404	32	RW	0xEA60	Host Frame Interval Register
hfnum on page 18-121	0x408	32	RO	0x3FFF	Host Frame Number Frame Time Remaining Register
hptxsts on page 18-122	0x410	32	RO	0x102000	Host Periodic Transmit FIFO Queue Status Register
haint on page 18-125	0x414	32	RO	0x0	Host All Channels Interrupt Register
haintmsk on page 18-126	0x418	32	RW	0x0	Host All Channels Interrupt Mask Register
hflbaddr on page 18-126	0x41C	32	RW	0x0	Host Frame List Base Address Register
hprt on page 18-127	0x440	32	RW	0x0	Host Port Control and Status Register
hcchar0 on page 18-131	0x500	32	RW	0x0	Host Channel 0 Characteristics Register
hcsplt0 on page 18-134	0x504	32	RW	0x0	Host Channel 0 Split Control Register
hcint0 on page 18-136	0x508	32	RO	0x0	Host Channel 0 Interrupt Register
hcintmsk0 on page 18-140	0x50C	32	RW	0x0	Host Channel 0 Interrupt Mask Register
hctsiz0 on page 18-141	0x510	32	RW	0x0	Host Channel 0 Transfer Size Register
hcdma0 on page 18-143	0x514	32	RW	0x0	Host Channel 0 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab0 on page 18-144	0x518	32	RW	0x0	Host Channel 0 DMA Buffer Address Register
hcchar1 on page 18-145	0x520	32	RW	0x0	Host Channel 1 Characteristics Register
hcsplt1 on page 18-148	0x524	32	RW	0x0	Host Channel 1 Split Control Register
hcint1 on page 18-150	0x528	32	RO	0x0	Host Channel 1 Interrupt Register
hcintmsk1 on page 18-154	0x52C	32	RW	0x0	Host Channel 1 Interrupt Mask Register
hctsiz1 on page 18-155	0x530	32	RW	0x0	Host Channel 1 Transfer Size Register
hcdma1 on page 18-157	0x534	32	RW	0x0	Host Channel 1 DMA Address Register
hcdmab1 on page 18-158	0x538	32	RW	0x0	Host Channel 1 DMA Buffer Address Register
hcchar2 on page 18-159	0x540	32	RW	0x0	Host Channel 2 Characteristics Register
hcsplt2 on page 18-162	0x544	32	RW	0x0	Host Channel 2 Split Control Register
hcint2 on page 18-164	0x548	32	RO	0x0	Host Channel 2 Interrupt Register
hcintmsk2 on page 18-168	0x54C	32	RW	0x0	Host Channel 2 Interrupt Mask Register
hctsiz2 on page 18-169	0x550	32	RW	0x0	Host Channel 2 Transfer Size Register
hcdma2 on page 18-171	0x554	32	RW	0x0	Host Channel 2 DMA Address Register
hcdmab2 on page 18-172	0x558	32	RW	0x0	Host Channel 2 DMA Buffer Address Register
hcchar3 on page 18-173	0x560	32	RW	0x0	Host Channel 3 Characteristics Register
hcsplt3 on page 18-176	0x564	32	RW	0x0	Host Channel 3 Split Control Register
hcint3 on page 18-178	0x568	32	RO	0x0	Host Channel 3 Interrupt Register
hcintmsk3 on page 18-182	0x56C	32	RW	0x0	Host Channel 3 Interrupt Mask Registe
hctsiz3 on page 18-183	0x570	32	RW	0x0	Host Channel 3 Transfer Size Registe
hcdma3 on page 18-185	0x574	32	RW	0x0	Host Channel 3 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab3 on page 18-186	0x578	32	RW	0x0	Host Channel 3 DMA Buffer Address Register
hcchar4 on page 18-187	0x580	32	RW	0x0	Host Channel 4 Characteristics Register
hcsplt4 on page 18-187	0x584	32	RW	0x0	Host Channel 4 Split Control Register
hcint4 on page 18-189	0x588	32	RO	0x0	Host Channel 4 Interrupt Register
hcintmsk4 on page 18-193	0x58C	32	RW	0x0	Host Channel 4 Interrupt Mask Register
hctsiz4 on page 18-195	0x590	32	RW	0x0	Host Channel 4 Transfer Size Register
hcdma4 on page 18-196	0x594	32	RW	0x0	Host Channel 4 DMA Address Register
hcdmab4 on page 18-197	0x598	32	RW	0x0	Host Channel 4 DMA Buffer Address Register
hcchar5 on page 18-198	0x5A0	32	RW	0x0	Host Channel 5 Characteristics Register
hcsplt5 on page 18-202	0x5A4	32	RW	0x0	Host Channel 5 Split Control Register
hcint5 on page 18-204	0x5A8	32	RO	0x0	Host Channel 5 Interrupt Register
hcintmsk5 on page 18-208	0x5AC	32	RW	0x0	Host Channel 5 Interrupt Mask Register
hctsiz5 on page 18-209	0x5B0	32	RW	0x0	Host Channel 5 Transfer Size Register
hcdma5 on page 18-211	0x5B4	32	RW	0x0	Host Channel 5 DMA Address Register
hcdmab5 on page 18-212	0x5B8	32	RW	0x0	Host Channel 5 DMA Buffer Address Register
hcchar6 on page 18-213	0x5C0	32	RW	0x0	Host Channel 6 Characteristics Register
hcsplt6 on page 18-216	0x5C4	32	RW	0x0	Host Channel 6 Split Control Register
hcint6 on page 18-218	0x5C8	32	RO	0x0	Host Channel 6 Interrupt Register
hcintmsk6 on page 18-222	0x5CC	32	RW	0x0	Host Channel 6 Interrupt Mask Register
hctsiz6 on page 18-223	0x5D0	32	RW	0x0	Host Channel 6 Transfer Size Register
hcdma6 on page 18-225	0x5D4	32	RW	0x0	Host Channel DMA Address Registe

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab6 on page 18-226	0x5D8	32	RW	0x0	Host Channel 6 DMA Buffer Address Register
hcchar7 on page 18-227	0x5E0	32	RW	0x0	Host Channel 7 Characteristics Register
hcsplt7 on page 18-230	0x5E4	32	RW	0x0	Host Channel 7 Split Control Register
hcint7 on page 18-232	0x5E8	32	RO	0x0	Host Channel 7 Interrupt Register
hcintmsk7 on page 18-236	0x5EC	32	RW	0x0	Host Channel 7 Interrupt Mask Register
hctsiz7 on page 18-237	0x5F0	32	RW	0x0	Host Channel 7 Transfer Size Register
hcdma7 on page 18-239	0x5F4	32	RW	0x0	Host Channel 7 DMA Address Register
hcdmab7 on page 18-240	0x5F8	32	RW	0x0	Host Channel 7 DMA Buffer Address Register
hcchar8 on page 18-241	0x600	32	RW	0x0	Host Channel 8 Characteristics Register
hcsplt8 on page 18-244	0x604	32	RW	0x0	Host Channel 8 Split Control Register
hcint8 on page 18-246	0x608	32	RO	0x0	Host Channel 8 Interrupt Register
hcintmsk8 on page 18-250	0x60C	32	RW	0x0	Host Channel 8 Interrupt Mask Register
hctsiz8 on page 18-251	0x610	32	RW	0x0	Host Channel 8 Transfer Size Register
hcdma8 on page 18-253	0x614	32	RW	0x0	Host Channel 8 DMA Address Register
hcdmab8 on page 18-254	0x618	32	RW	0x0	Host Channel 8 DMA Buffer Address Register
hcchar9 on page 18-255	0x620	32	RW	0x0	Host Channel 9 Characteristics Register
hcsplt9 on page 18-258	0x624	32	RW	0x0	Host Channel 9 Split Control Register
hcint9 on page 18-260	0x628	32	RO	0x0	Host Channel 9 Interrupt Register
hcintmsk9 on page 18-264	0x62C	32	RW	0x0	Host Channel 9 Interrupt Mask Register
hctsiz9 on page 18-265	0x630	32	RW	0x0	Host Channel 9 Transfer Size Register
hcdma9 on page 18-267	0x634	32	RW	0x0	Host Channel DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab9 on page 18-268	0x638	32	RW	0x0	Host Channel 9 DMA Buffer Address Register
hcchar10 on page 18-269	0x640	32	RW	0x0	Host Channel 10 Characteristics Register
hcsplt10 on page 18-272	0x644	32	RW	0x0	Host Channel 10 Split Control Register
hcint10 on page 18-274	0x648	32	RO	0x0	Host Channel 10 Interrupt Register
hcintmsk10 on page 18-278	0x64C	32	RW	0x0	Host Channel 10 Interrupt Mask Register
hctsiz10 on page 18-279	0x650	32	RW	0x0	Host Channel 10 Transfer Size Register
hcdma10 on page 18-281	0x654	32	RW	0x0	Host Channel 10 DMA Address Register
hcdmab10 on page 18-282	0x658	32	RW	0x0	Host Channel 10 DMA Buffer Address Register
hcchar11 on page 18-283	0x660	32	RW	0x0	Host Channel 11 Characteristics Register
HCSPLT11 on page 18-286	0x664	32	RW	0x0	Host Channel 11 Split Control Register
hcint11 on page 18-288	0x668	32	RO	0x0	Host Channel 11 Interrupt Register
hcintmsk11 on page 18-292	0x66C	32	RW	0x0	Channel 11 Interrupt Mask Register
hctsiz11 on page 18-293	0x670	32	RW	0x0	Host Channel 11 Transfer Size Register
hcdma11 on page 18-295	0x674	32	RW	0x0	Host Channel 11 DMA Address Register
hcdmab11 on page 18-296	0x678	32	RW	0x0	Host Channel 11 DMA Buffer Address Register
hcchar12 on page 18-297	0x680	32	RW	0x0	Host Channel 12 Characteristics Register
hcsplt12 on page 18-300	0x684	32	RW	0x0	Host Channel 12 Split Control Register
hcint12 on page 18-302	0x688	32	RO	0x0	Host Channel 12 Interrupt Register
hcintmsk12 on page 18-306	0x68C	32	RW	0x0	Host Channel 12 Interrupt Mask Register
hctsiz12 on page 18-307	0x690	32	RW	0x0	Host Channel 12 Transfer Size Register
hcdma12 on page 18-309	0x694	32	RW	0x0	Host Channel 12 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab12 on page 18-310	0x698	32	RW	0x0	Host Channel 12 DMA Buffer Address Register
hcchar13 on page 18-311	0x6A0	32	RW	0x0	Host Channel 13 Characteristics Register
hcsplt13 on page 18-314	0x6A4	32	RW	0x0	Host Channel 13 Split Control Register
hcint13 on page 18-316	0x6A8	32	RO	0x0	Host Channel 13 Interrupt Register
hcintmsk13 on page 18-320	0x6AC	32	RW	0x0	Host Channel 13 Interrupt Mask Registe
hctsiz13 on page 18-321	0x6B0	32	RW	0x0	Host Channel 13 Transfer Size Register
hcdma13 on page 18-323	0x6B4	32	RW	0x0	Host Channel 13 DMA Address Register
hcdmab13 on page 18-324	0x6B8	32	RW	0x0	Host Channel 13 DMA Buffer Address Register
hcchar14 on page 18-325	0x6C0	32	RW	0x0	Host Channel 14 Characteristics Register
hcsplt14 on page 18-328	0x6C4	32	RW	0x0	Host Channel 14 Split Control Register
hcint14 on page 18-330	0x6C8	32	RO	0x0	Host Channel 14 Interrupt Register
hcintmsk14 on page 18-334	0x6CC	32	RW	0x0	Host Channel 14 Interrupt Mask Register
hctsiz14 on page 18-335	0x6D0	32	RW	0x0	Host Channel 14 Transfer Size Register
hcdma14 on page 18-337	0x6D4	32	RW	0x0	Host Channel 14 DMA Address Register
hcdmab14 on page 18-338	0x6D8	32	RW	0x0	Host Channel 14 DMA Buffer Address Register
hcchar15 on page 18-339	0x6E0	32	RW	0x0	Host Channel 15 Characteristics Register
hcsplt15 on page 18-342	0x6E4	32	RW	0x0	Host Channel 15 Split Control Register
hcint15 on page 18-344	0x6E8	32	RO	0x0	Host Channel 15 Interrupt Register
hcintmsk15 on page 18-348	0x6EC	32	RW	0x0	Host Channel 15 Interrupt Mask Register
hctsiz15 on page 18-349	0x6F0	32	RW	0x0	Host Channel 15 Transfer Size Register
hcdma15 on page 18-351	0x6F4	32	RW	0x0	Host Channel 15 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
hcdmab15 on page 18-352	0x6F8	32	RW	0x0	Host Channel 15 DMA Buffer Address Register

Device Mode Registers

Register	Offset	Width	Acces s	Reset Value	Description
dcfg on page 18-369	0x800	32	RW	0x8000000	Device Configuration Register
dctl on page 18-372	0x804	32	RW	0x0	Device Control Register
dsts on page 18-377	0x808	32	RO	0x2	Device Status Register
diepmsk on page 18-379	0x810	32	RW	0x0	Device IN Endpoint Common Interrupt Mask Register
doepmsk on page 18-381	0x814	32	RW	0x0	Device OUT Endpoint Common Interrupt Mask Register
daint on page 18-383	0x818	32	RO	0x0	Device All Endpoints Interrupt Register
daintmsk on page 18-389	0x81C	32	RW	0x0	Device All Endpoints Interrupt Mask Register
dvbusdis on page 18-394	0x828	32	RW	0x17D7	Device VBUS Discharge Time Register
dvbuspulse on page 18-395	0x82C	32	RW	0x5B8	Device VBUS Pulsing Time Register
dthrctl on page 18-395	0x830	32	RW	0x8100020	Device Threshold Control Register
diepempmsk on page 18-398	0x834	32	RW	0x0	Device IN Endpoint FIFO Empty Interrupt Mask Register
diepctl0 on page 18-401	0x900	32	RW	0x8000	Device Control IN Endpoint 0 Control Register
diepint0 on page 18-404	0x908	32	RO	0x80	Device IN Endpoint 0 Interrupt Register
dieptsiz0 on page 18-408	0x910	32	RW	0x0	Device IN Endpoint 0 Transfer Size Register
diepdma0 on page 18-409	0x914	32	RW	0x0	Device IN Endpoint 0 DMA Address Register
dtxfsts0 on page 18-410	0x918	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 0
diepdmab0 on page 18-410	0x91C	32	RO	0x0	Device IN Endpoint 0 DMA Buffer Address Register
diepctl1 on page 18-411	0x920	32	RW	0x0	Device Control IN Endpoint 1 Control Register

Register	Offset	Width	Acces s	Reset Value	Description
diepint1 on page 18-417	0x928	32	RO	0x80	Device IN Endpoint 1 Interrupt Register
dieptsiz1 on page 18-421	0x930	32	RW	0x0	Device IN Endpoint 1 Transfer Size Register
diepdma1 on page 18-422	0x934	32	RW	0x0	Device IN Endpoint 1 DMA Address Registe
dtxfsts1 on page 18-423	0x938	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 1
diepdmab1 on page 18-424	0x93C	32	RO	0x0	Device IN Endpoint 1 DMA Buffer Address Register
diepctl2 on page 18-424	0x940	32	RW	0x0	Device Control IN Endpoint 2 Control Register
diepint2 on page 18-430	0x948	32	RO	0x80	Device IN Endpoint 2 Interrupt Register
dieptsiz2 on page 18-434	0x950	32	RW	0x0	Device IN Endpoint 2 Transfer Size Register
diepdma2 on page 18-435	0x954	32	RW	0x0	Device IN Endpoint 2 DMA Address Register
DTXFSTS2 on page 18-436	0x958	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 2
diepdmab2 on page 18-437	0x95C	32	RO	0x0	Device IN Endpoint 2 DMA Buffer Address Register
diepctl3 on page 18-437	0x960	32	RW	0x0	Device Control IN Endpoint 3 Control Register
diepint3 on page 18-443	0x968	32	RO	0x80	Device IN Endpoint 3 Interrupt Register
dieptsiz3 on page 18-447	0x970	32	RW	0x0	Device IN Endpoint 3 Transfer Size Registe
diepdma3 on page 18-448	0x974	32	RW	0x0	Device IN Endpoint 3 DMA Address Registe
dtxfsts3 on page 18-449	0x978	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 3
diepdmab3 on page 18-450	0x97C	32	RO	0x0	Device IN Endpoint 3 DMA Buffer Address Register
diepctl4 on page 18-450	0x980	32	RW	0x0	Device Control IN Endpoint 4 Control Register
diepint4 on page 18-456	0x988	32	RO	0x80	Device IN Endpoint 4 Interrupt Register

Register	Offset	Width	Access	Reset Value	Description
dieptsiz4 on page 18-460	0x990	32	RW	0x0	Device IN Endpoint 4 Transfer Size Register
diepdma4 on page 18-461	0x994	32	RW	0x0	Device IN Endpoint 4 DMA Address Register
dtxfsts4 on page 18-462	0x998	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 4
diepdmab4 on page 18-463	0x99C	32	RO	0x0	Device IN Endpoint 4 DMA Buffer Address Register
diepctl5 on page 18-463	0x9A0	32	RW	0x0	Device Control IN Endpoint 5 Control Register
diepint5 on page 18-469	0x9A8	32	RO	0x80	Device IN Endpoint 5 Interrupt Register
dieptsiz5 on page 18-473	0x9B0	32	RW	0x0	Device IN Endpoint 5 Transfer Size Register
diepdma5 on page 18-474	0x9B4	32	RW	0x0	Device IN Endpoint 5 DMA Address Register
dtxfsts5 on page 18-475	0x9B8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 5
diepdmab5 on page 18-476	0x9BC	32	RO	0x0	Device IN Endpoint 5 DMA Buffer Address Register
diepctl6 on page 18-476	0x9C0	32	RW	0x0	Device Control IN Endpoint 6 Control Register
diepint6 on page 18-482	0x9C8	32	RO	0x80	Device IN Endpoint 6 Interrupt Register
dieptsiz6 on page 18-486	0x9D0	32	RW	0x0	Device IN Endpoint 6 Transfer Size Register
diepdma6 on page 18-487	0x9D4	32	RW	0x0	Device IN Endpoint 6 DMA Address Register
dtxfsts6 on page 18-488	0x9D8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 6
diepdmab6 on page 18-489	0x9DC	32	RO	0x0	Device IN Endpoint 6 DMA Buffer Address Register
diepctl7 on page 18-489	0x9E0	32	RW	0x0	Device Control IN Endpoint 7 Control Register
diepint7 on page 18-495	0x9E8	32	RO	0x80	Device IN Endpoint 7 Interrupt Register
dieptsiz7 on page 18-499	0x9F0	32	RW	0x0	Device IN Endpoint 7 Transfer Size Register

Register	Offset	Width	Acces s	Reset Value	Description
diepdma7 on page 18-500	0x9F4	32	RW	0x0	Device IN Endpoint 7 DMA Address Register
dtxfsts7 on page 18-501	0x9F8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 7
diepdmab7 on page 18-502	0x9FC	32	RO	0x0	Device IN Endpoint 7 DMA Buffer Address Register
diepctl8 on page 18-502	0xA00	32	RW	0x0	Device Control IN Endpoint 8 Control Register
diepint8 on page 18-508	0xA08	32	RO	0x80	Device IN Endpoint 8 Interrupt Register
dieptsiz8 on page 18-512	0xA10	32	RW	0x0	Device IN Endpoint 8 Transfer Size Register
diepdma8 on page 18-513	0xA14	32	RW	0x0	Device IN Endpoint 8 DMA Address Register
dtxfsts8 on page 18-514	0xA18	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 8
diepdmab8 on page 18-515	0xA1C	32	RO	0x0	Device IN Endpoint 8 DMA Buffer Address Register
diepctl9 on page 18-515	0xA20	32	RW	0x0	Device Control IN Endpoint 9 Control Register
diepint9 on page 18-521	0xA28	32	RO	0x80	Device IN Endpoint 9 Interrupt Register
dieptsiz9 on page 18-525	0xA30	32	RW	0x0	Device IN Endpoint 9 Transfer Size Register
diepdma9 on page 18-526	0xA34	32	RW	0x0	Device IN Endpoint 9 DMA Address Register
dtxfsts9 on page 18-527	0xA38	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 9
diepdmab9 on page 18-528	0xA3C	32	RO	0x0	Device IN Endpoint 9 DMA Buffer Address Register
diepctl10 on page 18-528	0xA40	32	RW	0x0	Device Control IN Endpoint 10 Control Register
diepint10 on page 18-534	0xA48	32	RO	0x80	Device IN Endpoint 10 Interrupt Register
dieptsiz10 on page 18-538	0xA50	32	RW	0x0	Device IN Endpoint 10 Transfer Size Register
diepdma10 on page 18-539	0xA54	32	RW	0x0	Device IN Endpoint 10 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
dtxfsts10 on page 18-540	0xA58	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 10
diepdmab10 on page 18-541	0xA5C	32	RO	0x0	Device IN Endpoint 10 DMA Buffer Address Register
diepctl11 on page 18-541	0xA60	32	RW	0x0	Device Control IN Endpoint 11 Control Register
diepint11 on page 18-547	0xA68	32	RO	0x80	Device IN Endpoint 11 Interrupt Register
dieptsiz11 on page 18-551	0xA70	32	RW	0x0	Device IN Endpoint 11 Transfer Size Register
diepdma11 on page 18-552	0xA74	32	RW	0x0	Device IN Endpoint 11 DMA Address Register
dtxfsts11 on page 18-553	0xA78	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 11
diepdmab11 on page 18-554	0xA7C	32	RO	0x0	Device IN Endpoint 11 DMA Buffer Address Register
diepctl12 on page 18-554	0xA80	32	RW	0x0	Device Control IN Endpoint 12 Control Register
diepint12 on page 18-560	0xA88	32	RO	0x80	Device IN Endpoint 12 Interrupt Register
dieptsiz12 on page 18-564	0xA90	32	RW	0x0	Device IN Endpoint 12 Transfer Size Register
diepdma12 on page 18-565	0xA94	32	RW	0x0	Device IN Endpoint 12 DMA Address Register
dtxfsts12 on page 18-566	0xA98	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 12
diepdmab12 on page 18-567	0xA9C	32	RO	0x0	Device IN Endpoint 12 DMA Buffer Address Register
diepctl13 on page 18-567	0xAA0	32	RW	0x0	Device Control IN Endpoint 13 Control Register
diepint13 on page 18-573	0xAA8	32	RO	0x80	Device IN Endpoint 13 Interrupt Register
dieptsiz13 on page 18-577	0xAB0	32	RW	0x0	Device IN Endpoint 13 Transfer Size Register
diepdma13 on page 18-578	0xAB4	32	RW	0x0	Device IN Endpoint 13 DMA Address Register
dtxfsts13 on page 18-579	0xAB8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 13

Register	Offset	Width	Acces s	Reset Value	Description
diepdmab13 on page 18-580	0xABC	32	RO	0x0	Device IN Endpoint 13 DMA Buffer Address Register
diepctl14 on page 18-580	0xAC0	32	RW	0x0	Device Control IN Endpoint 14 Control Register
diepint14 on page 18-586	0xAC8	32	RO	0x80	Device IN Endpoint 14 Interrupt Register
dieptsiz14 on page 18-590	0xAD0	32	RW	0x0	Device IN Endpoint 14 Transfer Size Register
diepdma14 on page 18-591	0xAD4	32	RW	0x0	Device IN Endpoint 14 DMA Address Register
dtxfsts14 on page 18-592	0xAD8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 14
diepdmab14 on page 18-593	0xADC	32	RO	0x0	Device IN Endpoint 14 DMA Buffer Address Register
diepctl15 on page 18-593	0xAE0	32	RW	0x0	Device Control IN Endpoint 15 Control Registe
diepint15 on page 18-599	0xAE8	32	RO	0x80	Device IN Endpoint 15 Interrupt Register
dieptsiz15 on page 18-603	0xAF0	32	RW	0x0	Device IN Endpoint 15 Transfer Size Register
diepdma15 on page 18-604	0xAF4	32	RW	0x0	Device IN Endpoint 15 DMA Address Register
dtxfsts15 on page 18-605	0xAF8	32	RO	0x2000	Device IN Endpoint Transmit FIFO Status Register 15
diepdmab15 on page 18-606	0xAFC	32	RO	0x0	Device IN Endpoint 15 DMA Buffer Address Register
doepctl0 on page 18-606	0xB00	32	RW	0x8000	Device Control OUT Endpoint 0 Control Register
doepint0 on page 18-609	0xB08	32	RO	0x0	Device OUT Endpoint 0 Interrupt Register
doeptsiz0 on page 18-613	0xB10	32	RW	0x0	Device OUT Endpoint 0 Transfer Size Register
doepdma0 on page 18-614	0xB14	32	RW	0x0	Device OUT Endpoint 0 DMA Address Register
doepdmab0 on page 18-615	0xB1C	32	RO	0x0	Device OUT Endpoint 16 DMA Buffer Address Register
doepctl1 on page 18-616	0xB20	32	RW	0x0	Device Control OUT Endpoint 1 Control Register

Register	Offset	Width	Acces s	Reset Value	Description
doepint1 on page 18-621	0xB28	32	RO	0x0	Device OUT Endpoint 1 Interrupt Register
doeptsiz1 on page 18-625	0xB30	32	RW	0x0	Device OUT Endpoint 1 Transfer Size Register
doepdma1 on page 18-626	0xB34	32	RW	0x0	Device OUT Endpoint 1 DMA Address Register
doepdmab1 on page 18-627	0xB3C	32	RO	0x0	Device OUT Endpoint 1 DMA Buffer Address Register
DOEPCTL2 on page 18-628	0xB40	32	RW	0x0	Device Control OUT Endpoint 2 Control Register
doepint2 on page 18-633	0xB48	32	RO	0x0	Device OUT Endpoint 2 Interrupt Register
doeptsiz2 on page 18-637	0xB50	32	RW	0x0	Device OUT Endpoint 2 Transfer Size Register
doepdma2 on page 18-638	0xB54	32	RW	0x0	Device OUT Endpoint 2 DMA Address Register
doepdmab2 on page 18-639	0xB5C	32	RO	0x0	Device OUT Endpoint 2 DMA Buffer Address Register
DOEPCTL3 on page 18-640	0xB60	32	RW	0x0	Device Control OUT Endpoint 3 Control Register
doepint3 on page 18-645	0xB68	32	RO	0x0	Device OUT Endpoint 3 Interrupt Register
doeptsiz3 on page 18-649	0xB70	32	RW	0x0	Device OUT Endpoint 3 Transfer Size Register
doepdma3 on page 18-650	0xB74	32	RW	0x0	Device OUT Endpoint 3 DMA Address Register
doepdmab3 on page 18-651	0xB7C	32	RO	0x0	Device OUT Endpoint 3 DMA Buffer Address Register
doepctl4 on page 18-652	0xB80	32	RW	0x0	Device Control OUT Endpoint 4 Control Register
Doepint4 on page 18-657	0xB88	32	RO	0x0	Device OUT Endpoint 4 Interrupt Register
doeptsiz4 on page 18-661	0xB90	32	RW	0x0	Device OUT Endpoint 4 Transfer Size Register
doepdma4 on page 18-662	0xB94	32	RW	0x0	Device OUT Endpoint 4 DMA Address Register
doepdmab4 on page 18-663	0xB9C	32	RO	0x0	Device OUT Endpoint 4 Buffer Address Register

Register	Offset	Width	Acces s	Reset Value	Description
doepctl5 on page 18-664	0xBA0	32	RW	0x0	Device Control OUT Endpoint 5 Control Register
doepint5 on page 18-669	0xBA8	32	RO	0x0	Device OUT Endpoint 5 Interrupt Register
doeptsiz5 on page 18-673	0xBB0	32	RW	0x0	Device OUT Endpoint 5 Transfer Size Register
doepdma5 on page 18-674	0xBB4	32	RW	0x0	Device OUT Endpoint 5 DMA Address Register
doepdmab5 on page 18-675	0xBBC	32	RO	0x0	Device OUT Endpoint 5 DMA Buffer Address Register
doepctl6 on page 18-676	0xBC0	32	RW	0x0	Device Control OUT Endpoint 6 Control Register
doepint6 on page 18-681	0xBC8	32	RO	0x0	Device OUT Endpoint 6 Interrupt Register
doeptsiz6 on page 18-685	0xBD0	32	RW	0x0	Device OUT Endpoint 6 Transfer Size Register
doepdma6 on page 18-686	0xBD4	32	RW	0x0	Device OUT Endpoint 6 DMA Address Register
doepdmab6 on page 18-687	0xBDC	32	RO	0x0	Device OUT Endpoint 6 DMA Buffer Address Register
doepctl7 on page 18-688	0xBE0	32	RW	0x0	Device Control OUT Endpoint 7 Control Register
doepint7 on page 18-694	0xBE8	32	RO	0x0	Device OUT Endpoint 7 Interrupt Register
doeptsiz7 on page 18-698	0xBF0	32	RW	0x0	Device OUT Endpoint 7 Transfer Size Register
doepdma7 on page 18-699	0xBF4	32	RW	0x0	Device OUT Endpoint 7 DMA Address Register
doepdmab7 on page 18-699	0xBFC	32	RO	0x0	Device OUT Endpoint 7 Buffer Address
doepctl8 on page 18-700	0xC00	32	RW	0x0	Device Control OUT Endpoint 8 Control Register
doepint8 on page 18-705	0xC08	32	RO	0x0	Device OUT Endpoint 8 Interrupt Register
doeptsiz8 on page 18-709	0xC10	32	RW	0x0	Device OUT Endpoint 8 Transfer Size Register
doepdma8 on page 18-710	0xC14	32	RW	0x0	Device OUT Endpoint 8 DMA Address Register

Register	Offset	Width	Acces s	Reset Value	Description
doepdmab8 on page 18-711	0xC1C	32	RO	0x0	Device OUT Endpoint 8 DMA Buffer Address Register
doepctl9 on page 18-712	0xC20	32	RW	0x0	Device Control OUT Endpoint 9 Control Register
doepint9 on page 18-717	0xC28	32	RO	0x0	Device OUT Endpoint 9 Interrupt Register
doeptsiz9 on page 18-721	0xC30	32	RW	0x0	Device OUT Endpoint 9 Transfer Size Register
doepdma9 on page 18-722	0xC34	32	RW	0x0	Device OUT Endpoint 9 DMA Address Register
doepdmab9 on page 18-723	0xC3C	32	RO	0x0	Device OUT Endpoint 9 DMA Buffer Address Register
doepctl10 on page 18-724	0xC40	32	RW	0x0	Device Control OUT Endpoint 10 Control Register
doepint10 on page 18-729	0xC48	32	RO	0x0	Device OUT Endpoint 10 Interrupt Register
doeptsiz10 on page 18-733	0xC50	32	RW	0x0	Device OUT Endpoint 10 Transfer Size Register
doepdma10 on page 18-734	0xC54	32	RW	0x0	Device OUT Endpoint 10 DMA Address Register
doepdmab10 on page 18-735	0xC5C	32	RO	0x0	Device OUT Endpoint 10 DMA Buffer Address Register
doepctl11 on page 18-736	0xC60	32	RW	0x0	Device Control OUT Endpoint 11 Control Register
doepint11 on page 18-741	0xC68	32	RO	0x0	Device OUT Endpoint 11 Interrupt Register
doeptsiz11 on page 18-745	0xC70	32	RW	0x0	Device OUT Endpoint 11 Transfer Size Register
doepdma11 on page 18-746	0xC74	32	RW	0x0	Device OUT Endpoint 11 DMA Address Register
doepdmab11 on page 18-747	0xC7C	32	RO	0x0	Device OUT Endpoint 11 DMA Buffer Address Register
doepctl12 on page 18-748	0xC80	32	RW	0x0	Device Control OUT Endpoint 12 Control Register
doepint12 on page 18-753	0xC88	32	RO	0x0	Device OUT Endpoint 12 Interrupt Register
doeptsiz12 on page 18-757	0xC90	32	RW	0x0	Device OUT Endpoint 12 Transfer Size Register

Register	Offset	Width	Access	Reset Value	Description
doepdma12 on page 18-758	0xC94	32	RW	0x0	Device OUT Endpoint 12 DMA Address Register
doepdmab12 on page 18-759	0xC9C	32	RO	0x0	Device OUT Endpoint 12 DMA Buffer Address Register
doepctl13 on page 18-760	0xCA0	32	RW	0x0	Device Control OUT Endpoint 13 Control Register
doepint13 on page 18-765	0xCA8	32	RO	0x0	Device OUT Endpoint 13 Interrupt Register
doeptsiz13 on page 18-769	0xCB0	32	RW	0x0	Device OUT Endpoint 13 Transfer Size Register
doepdma13 on page 18-770	0xCB4	32	RW	0x0	Device OUT Endpoint 13 DMA Address Register
doepdmab13 on page 18-771	0xCBC	32	RO	0x0	Device OUT Endpoint 13 DMA Buffer Address Register
doepctl14 on page 18-772	0xCC0	32	RW	0x0	Device Control OUT Endpoint 14 Control Register
doepint14 on page 18-777	0xCC8	32	RO	0x0	Device OUT Endpoint 14 Interrupt Register
doeptsiz14 on page 18-781	0xCD0	32	RW	0x0	Device OUT Endpoint 14 Transfer Size Register
doepdma14 on page 18-782	0xCD4	32	RW	0x0	Device OUT Endpoint 14 DMA Address Register
doepdmab14 on page 18-783	0xCDC	32	RO	0x0	Device OUT Endpoint 14 DMA Buffer Address Register
doepctl15 on page 18-784	0xCE0	32	RW	0x0	Device Control OUT Endpoint 15 Control Register
doepint15 on page 18-789	0xCE8	32	RO	0x0	Device OUT Endpoint 15 Interrupt Register
doeptsiz15 on page 18-793	0xCF0	32	RW	0x0	Device OUT Endpoint 15 Transfer Size Register
doepdma15 on page 18-794	0xCF4	32	RW	0x0	Device OUT Endpoint 15 DMA Address Register
doepdmab15 on page 18-795	0xCFC	32	RO	0x0	Device OUT Endpoint 15 DMA Buffer Address Register

Power and Clock Gating Register

Register	Offset	Width	Acces s	Reset Value	Description
pcgcctl on page 18-796	0xE00	32	RW	0x0	Power and Clock Gating Control Register

USB Data FIFO Address Map

Name	Description	Start Address Offset	End Address Offset
EP0/HC0 FIFO	This address space is allocated for Endpoint 0/Host Channel 0 push/pop FIFO access.	0x1000	0x1FFF
EP1/HC1 FIFO	This address space is allocated for Endpoint 1/Host Channel 1 push/pop FIFO access.	0x2000	0x2FFF
EP2/HC2 FIFO	This address space is allocated for Endpoint 2/Host Channel 2 push/pop FIFO access.	0x3000	0x3FFF
EP3/HC3 FIFO	This address space is allocated for Endpoint 3/Host Channel 3 push/pop FIFO access.	0x4000	0x4FFF
EP4/HC4 FIFO	This address space is allocated for Endpoint 4/Host Channel 4 push/pop FIFO access.	0x5000	0x5FFF
EP5/HC5 FIFO	This address space is allocated for Endpoint 5/Host Channel 5 push/pop FIFO access.	0x6000	0x6FFF
EP6/HC6 FIFO	This address space is allocated for Endpoint 6/Host Channel 6 push/pop FIFO access.	0x7000	0x7FFF

Name	Description	Start Address Offset	End Address Offset
EP7/HC7 FIFO	This address space is allocated for Endpoint 7/Host Channel 7 push/pop FIFO access.	0x8000	0x8FFF
EP8/HC8 FIFO	This address space is allocated for Endpoint 8/Host Channel 8 push/pop FIFO access.	0x9000	0x9FFF
EP9/HC9 FIFO	This address space is allocated for Endpoint 9/Host Channel 9 push/pop FIFO access.	0xA000	0xAFFF
EP10/HC10 FIFO	This address space is allocated for Endpoint 10/Host Channel 10 push/pop FIFO access.	0xB000	0xBFFF
EP11/HC11 FIFO	This address space is allocated for Endpoint 11/Host Channel 11 push/pop FIFO access.	0xC000	0xCFFF
EP12/HC12 FIFO	This address space is allocated for Endpoint 12/Host Channel 12 push/pop FIFO access.	0xD000	0xDFFF
EP13/HC13 FIFO	This address space is allocated for Endpoint 13/Host Channel 13 push/pop FIFO access.	0xE000	0xEFFF
EP14/HC14 FIFO	This address space is allocated for Endpoint 14/Host Channel 14 push/pop FIFO access.	0xF000	0xFFFF
EP15/HC15 FIFO	This address space is allocated for Endpoint 15/Host Channel 15 push/pop FIFO access.	0x10000	0x10FFF

USB Direct Access FIFO RAM Address Map

Name	Description	Start Address Offset	End Address Offset
`Direct_FIFO`	This address space is allocated for directly accessing the data FIFO for debugging purposes.	`0x20000`	`0x3FFFF`

Global Registers Register Descriptions

These registers are available in both Host and Device modes, and do not need to be reprogrammed when switching between these modes.

Offset: `0x0`

gotgctl on page 18-39
The OTG Control and Status register controls the behavior and reflects the status of the OTG function.

gotgint on page 18-45
The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

gahbcfg on page 18-47
This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

gusbcfg on page 18-51
This register can be used to configure the core after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

grstctl on page 18-56
The application uses this register to reset various hardware features inside the core

gintsts on page 18-60
This register interrupts the application for system-level events in the current mode (Device mode or Host mode). Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode. To clear the interrupt status bits of type R_SS_WC, the application must write 1 into the bit. The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically. The application must clear the GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

gintmsk on page 18-69
This register works with the Interrupt Register (GINTSTS) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the GINTSTS register bit corresponding to that interrupt is still set.

grxstsr on page 18-74
A read to the Receive Status Read and Pop register additionally pops the: top data entry out of the RxFIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted. Use of these fields vary based on whether the HS OTG core is functioning as a host or a device. Do not read this register's reset value before configuring the core because the read value is "X" in the simulation.

grxstsp on page 18-75
A read to the Receive Status Read and Pop register additionally pops the: top data entry out of the RxFIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted. Use of these fields vary based on whether the HS OTG core is functioning as a host or a device. Do not read this register'ss reset value before configuring the core because the read value is "X" in the simulation.

grxfsiz on page 18-77
The application can program the RAM size that must be allocated to the RxFIFO.

gnptxfsiz on page 18-78
The application can program the RAM size and the memory start address for the Non-periodic TxFIFO. The fields of this register change, depending on host or device mode.

gnptxsts on page 18-78
In Device mode, this register is valid only in Shared FIFO operation. It contains the free space information for the Non-periodic TxFIFO and the Nonperiodic Transmit RequestQueue

gpvndctl on page 18-80
The application can use this register to access PHY registers. for a ULPI PHY, the core uses the ULPI interface for PHY register access. The application sets Vendor Control register for PHY register access and times the PHY register access. The application polls the VStatus Done bit in this register for the completion of the PHY register access

ggpio on page 18-81
The application can use this register for general purpose input/output ports or for debugging.

guid on page 18-82
This is a read/write register containing the User ID. This register can be used in the following ways: -To store the version or revision of your system -To store hardware configurations that are outside the otg core As a scratch register

gsnpsid on page 18-83
This read-only register contains the release number of the core being used.

ghwcfg1 on page 18-83
This register contains the logical endpoint direction(s).

ghwcfg2 on page 18-84
This register contains configuration options.

ghwcfg3 on page 18-88
This register contains the configuration options.

ghwcfg4 on page 18-90
This register contains the configuration options.

gdfifocfg on page 18-95
Specifies whether Dedicated Transmit FIFOs should be enabled in device mode.

hptxfsiz on page 18-95
This register holds the size and the memory start address of the Periodic TxFIFO

dieptxf1 on page 18-96
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf2 on page 18-97
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf3 on page 18-98
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf4 on page 18-98
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf5 on page 18-99
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf6 on page 18-100
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf7 on page 18-101
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf8 on page 18-101
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf9 on page 18-102
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf10 on page 18-103
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf11 on page 18-104
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf12 on page 18-104
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf13 on page 18-105
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf14 on page 18-106
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

dieptxf15 on page 18-107
This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

gotgctl

The OTG Control and Status register controls the behavior and reflects the status of the OTG function.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00000
usb1	0xFFB40000	0xFFB40000

Offset: 0x0

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											otgver RW 0x0	bsesvld RO 0x0	asesvld RO 0x0	dbnctime RO 0x0	conidsts RO 0x1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				devhnpen RW 0x0	hstsethnpen RW 0x0	hnpreq RW 0x0	hstnegscs RO 0x0	bvalidovval RW 0x0	bvalidoven RW 0x0	avalidovval RW 0x0	avalidoven RW 0x0	vbvalidovval RW 0x0	vbvalidoven RW 0x0	sesreq RW 0x0	sesreqscs RO 0x0

gotgctl Fields

Bit	Name	Description	Access	Reset
20	otgver	Indicates the OTG revision. In OTG Version 1.3. the core supports Data line pulsing and VBus pulsing for SRP. In OTG Version 2.0 the core supports only Data line pulsing for SRP. **Value** **Description** 0x0 OTG Version 1.3. In this version the core supports Data line 0x1 OTG Version 2.0. In this version the core supports only Data line pulsing for SRP	RW	0x0

Bit	Name	Description	Access	Reset
19	bsesvld	Mode: Device only. Indicates the Device mode transceiver status. In OTG mode, you can use this bit to determine IF the device is connected or disconnected. If you do not enable OTG features (such as SRP and HNP), the read reset value will be 1. The vbus assigns the values internally for non-SRP or non-HNP configurations. **Value** **Description** 0x0 B-session is not valid 0x1 B-session is valid	RO	0x0
18	asesvld	Mode: Host only. Indicates the Host mode transceiver status. If you do not enabled OTG features (such as SRP and HNP), the read reset value will be 1.The vbus assigns the values internally for non-SRP or non-HNP configurations. **Value** **Description** 0x0 A-session is not valid 0x1 A-session is valid	RO	0x0
17	dbnctime	Mode: Host only. Indicates the debounce time of a detected connection. **Value** **Description** 0x0 Long debounce time, used FOR physical connections (100 ms + 2.5 s) 0x1 Short debounce time, used FOR soft connections (2.5 s	RO	0x0
16	conidsts	Mode: Host and Device. Indicates the connector ID status on a connect event.This bit is valid only for Host and Device mode. **Value** **Description** 0x0 The DWC_otg core is in A-Device mode 0x1 The otg core is in B-Device mode	RO	0x1

Bit	Name	Description	Access	Reset
11	`devhnpen`	Mode: Device only. The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host. **Value** **Description** 0x0 HNP is not enabled in the application 0x1 HNP Enabled	RW	0x0
10	`hstsethnpen`	Mode: Host only. The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device. **Value** **Description** 0x0 Host Set HNP is not enabled 0x1 Host Set HNP is enabled	RW	0x0
9	`hnpreq`	Mode: Device only. The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is SET.The core clears this bit when the HstNegSucStsChng bit iscleared. **Value** **Description** 0x0 No HNP request 0x1 HNP request	RW	0x0
8	`hstnegscs`	Mode: Device only. Host Negotiation Success (HstNegScs) The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPReq) bit in this register is SET. **Value** **Description** 0x0 Host negotiation failure 0x1 Host negotiation success	RO	0x0

Bit	Name	Description	Access	Reset
7	bvalidovval	This bit is used to set Override value for Bvalid signalwhen GOTGCTL.BvalidOvEn is set. **Value** **Description** 0x0 Bvalid value when GOTGCTL.AvalidOvEn =1 0x1 Bvalid value when GOTGCTL.AvalidOvEn =1	RW	0x0
6	bvalidoven	This bit is used to enable/disable the software to override the Bvalid signal using the GOTGCTL.BvalidOvVal. **Value** **Description** 0x0 Override is disabled and bvalid signal from the respective PHY selected is used internally by the core 0x1 Internally Bvalid received from the PHY is overridden with GOTGCTL.BvalidOvVal	RW	0x0
5	avalidovval	This bit is used to set Override value for Avalid signal when GOTGCTL.BvalidOvEn is set. **Value** **Description** 0x0 Avalid value is 1'b0 when GOTGCTL.BvalidOvEn =1 0x1 Avalid value is 1'b1 when GOTGCTL.BvalidOvEn =1	RW	0x0
4	avalidoven	This bit is used to enable/disable the software to override the Avalid signal using the GOTGCTL.AvalidOvVal. **Value** **Description** 0x0 Override is disabled and Avalid signal from the respective PHY is used internally by the core. 0x1 Internally Avalid received from the PHY is overridden with GOTGCTL.AvalidOvVa	RW	0x0

Bit	Name	Description	Access	Reset
3	`vbvalidovval`	This bit is used to set Override value for vbus valid signal when GOTGCTL.VbvalidOvEn is set. **Value** **Description** 0x0 vbusvalid value when GOTGCTL.VbvalidOvEn = 1 0x1 vbusvalid value when GOTGCTL.VbvalidOvEn is 1	RW	0x0
2	`vbvalidoven`	This bit is used to enable/disable the software to override the vbus-valid signal using the GOTGCTL.vbvalidOvVal.. **Value** **Description** 0x0 Override is disabled and bvalid signal from the respective PHY selected is used internally by the force 0x1 The vbus-valid signal received from the PHY is overridden with GOTGCTL.vbvalidOvVal	RW	0x0
1	`sesreq`	The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the Host Negotiation Success Status Change bit in the OTG Interrupt register (GOTGINT.HstNegSucStsChng) is SET. The core clears this bit when the HstNegSucStsChng bit is cleared. If you use the USB 1.1 Full-Speed Serial Transceiver interface to initiate the session request, the application must wait until the VBUS discharges to 0.2 V, after the B-Session Valid bit in this register (GOTGCTL.BSesVld) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor. **Value** **Description** 0x0 No session request 0x1 Session request	RW	0x0

Bit	Name	Description	Access	Reset
0	sesreqscs	This bit is set when a session request initiation is successful. This bit is valid only For Device Only configuration when OTG_MODE == 3 or OTG_MODE == 4. Applies for device only. **Value** / **Description** 0x0 Session request failure 0x1 Session request success	RO	0x0

gotgint

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00004
usb1	0xFFB40000	0xFFB40004

Offset: 0x4

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												dbncedone RO 0x0	adevtoutchg RO 0x0	hstnegdet RO 0x0	Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						hstnegsucstschng RO 0x0	sesreqsucstschng RO 0x0	Reserved					sesenddet RO 0x0	Reserved	

gotgint Fields

Bit	Name	Description	Access	Reset
19	`dbncedone`	Mode: Host only. The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is SET in the Core USB Configuration register (GUSBCFG.HNPCap or GUSBCFG.SRPCap, respectively). This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Change 0x1 — Debounce completed	RO	0x0
18	`adevtoutchg`	Mode:Host and Device. The core sets this bit to indicate that the A-device has timed out WHILE waiting FOR the B-device to connect. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Change 0x1 — A-Device Timeout	RO	0x0
17	`hstnegdet`	Mode:Host and Device. The core sets this bit when it detects a host negotiation request on the USB. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Change 0x1 — Host Negotiation Detected	RO	0x0

Bit	Name	Description	Access	Reset
9	`hstnegsucstschng`	Mode: Host and Device. The core sets this bit on the success or failure of a USB host negotiation request. The application must read the Host Negotiation Success bit of the OTG Control and Status register (GOTGCTL.HstNegScs) to check for success or failure. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Change 0x1 Host Negotiation Status Change	`RO`	`0x0`
8	`sesreqsucstschng`	Mode: Host and Device. The core sets this bit on the success or failure of a session request. The application must read the Session Request Success bit in the OTG Control and Status register (GOTGCTL.SesReqScs) to check for success or failure. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No change 0x1 Session Request Status	`RO`	`0x0`
2	`sesenddet`	Mode:Host and Device.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 Non Active State 0x1 Set when utmisrp_bvalid signal is deasserted	`RO`	`0x0`

gahbcfg

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00008`
`usb1`	`0xFFB40000`	`0xFFB40008`

Offset: `0x8`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									notialldmawrit RW 0x0	remmemsupp RW 0x0	Reserved				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ptxfemplvl RW 0x0	nptxfemplvl RW 0x0	Reserved	dmaen RW 0x0	hbstlen RW 0x0				glblintrmsk RW 0x0

gahbcfg Fields

Bit	Name	Description	Access	Reset
22	notialldmawrit	This bit is programmed to enable the System DMA Done functionality for all the DMA write Transactions corresponding to the Channel/Endpoint. This bit is valid only when GAHBCFG.RemMemSupp is set to 1. **Value** **Description** 0x1 HSOTG core asserts int_dma_req for all the DMA write transactions on the AHB interface along with int_dma_done, chep_last_transact and chep_number signal informations. The core waits for sys_dma_done signal for all the DMA write transactions in order to complete the transfer of a particular Channel/Endpoint 0x0 HSOTG core asserts int_dma_req signal only for the last transaction of DMA write transfer corresponding to a particular Channel/Endpoint. Similarly, the core waits for sys_dma_done signal only for that transaction of DMA write to complete the transfer of a particular Channel/Endpoint	RW	0x0

Bit	Name	Description	Access	Reset
21	remmemsupp	This bit is programmed to enable/disable the functionality to wait for the system DMA Done Signal for the DMA Write Transfers. -The int_dma_req output signal is asserted when HSOTG DMA starts write transfer to the external memory. When the core is done with the Transfers it asserts int_dma_done signal to flag the completion of DMA writes from HSOTG. The core then waits for sys_dma_done signal from the system to proceed further and complete the Data Transfer corresponding to a particular Channel/Endpoint. -The int_dma_req and int_dma_done signals are not asserted and the core proceeds with the assertion of the XferComp interrupt as soon as wait for the system DMA Done Signal for the DMA Write Transfers the DMA write transfer is done at the HSOTG Core Boundary and it doesn't wait for the sys_dma_done signal to complete the DATA **Value** **Description** 0x0 Disable wait for system DMA Done Signal 0x1 Enable wait for the system DMA Done Signal for the DMA Write Transfers	RW	0x0
8	ptxfemplvl	Mode:Host only. Indicates when the Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.PTxFEmp) is triggered. This bit is used only in Slave mode. **Value** **Description** 0x0 GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is half empty 0x1 GINTSTS.PTxFEmp interrupt indicates that the Periodic TxFIFO is completely empty	RW	0x0

Bit	Name	Description	Access	Reset
7	`nptxfemplvl`	Mode:Host and device. This bit is used only in Slave mode. In host mode and with Shared FIFO with device mode, this bit indicates when the Non-Periodic TxFIFO Empty Interrupt bit in the Core Interrupt register (GINTSTS.NPTxFEmp) is triggered. With dedicated FIFO in device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (DIEPINTn.TxFEmp) is triggered. Host mode and with Shared FIFO with device mode: **Value** — **Description** 0x0 — DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is half empty or DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is half empty 0x1 — GINTSTS.NPTxFEmp interrupt indicates that the Non-Periodic TxFIFO is completely empty or DIEPINTn.TxFEmp interrupt indicates that the IN Endpoint TxFIFO is completely empty	RW	0x0
5	`dmaen`	Mode:Host and device. Enables switching from DMA mode to slave mode. **Value** — **Description** 0x0 — Core operates in Slave mode 0x1 — Core operates in a DMA mode	RW	0x0
4:1	`hbstlen`	Mode:Host and device. This field is used in Internal DMA modes. **Value** — **Description** 0x0 — 1 word or single 0x1 — 4 word or incr 0x2 — 8 word 0x3 — 16 word or incr4 0x4 — 32 word 0x5 — 64 word or incr8 0x6 — 128 word 0x7 — 256 word or incr16 0x8 — Others reserved	RW	0x0

Bit	Name	Description	Access	Reset
0	`glblintrmsk`	Mode: Host and device. The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bits setting, the interrupt status registers are updated by the core. **Value** **Description** 0x0 Mask the interrupt assertion to the application 0x1 Unmask the interrupt assertion to the application.	RW	0x0

gusbcfg

This register can be used to configure the core after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0000C
usb1	0xFFB40000	0xFFB4000C

Offset: `0xC`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
corrupttxpkt WO 0x0	forcedevmode RW 0x0	forcehstmode RW 0x0	txenddelay RW 0x0	Reserved		ulpi RW 0x0	indicator RW 0x0	complement RW 0x0	termseldlpulse RW 0x0	ulpiextvbusindicator RW 0x0	ulpiextvbusdrv RW 0x0	ulpiclksusm RW 0x0	ulpiautores RW 0x0	Reserved	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		usbtrdtim RW 0x5				hnpcap RW 0x0	srpcap RW 0x0	ddrsel RW 0x0	physel RO 0x0	fsintf RO 0x0	ulpi_utmi_sel RO 0x1	phyif RO 0x0	toutcal RW 0x0		

gusbcfg Fields

Bit	Name	Description	Access	Reset
31	`corrupttxpkt`	Mode: Host and device. This bit is for debug purposes only. Never Set this bit to 1. The application should always write 0 to this bit. **Value** **Description** 0x0 Normal Mode 0x1 Debug Mode	WO	0x0
30	`forcedevmode`	Mode:Host and device. Writing a 1 to this bit forces the core to device mode. After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 micro-sec is sufficient. **Value** **Description** 0x0 Normal Mode 0x1 Force Device Mode	RW	0x0
29	`forcehstmode`	Mode:Host and device. Writing a 1 to this bit forces the core to host mode After setting the force bit, the application must wait at least 25 ms before the change to take effect. When the simulation is in scale down mode, waiting for 500 micro-sec is sufficient. **Value** **Description** 0x0 Normal Mode 0x1 Force Host Mode	RW	0x0
28	`txenddelay`	Mode: Device only. Set to non UTMI+. **Value** **Description** 0x0 Normal Mode	RW	0x0
25	`ulpi`	Mode:Host only. Controls circuitry built into the PHY for protecting the ULPI interface when the link tri-states STP and data. Any pull-ups or pull-downs employed by this feature can be disabled. **Value** **Description** 0x0 Enables the interface protect circuit 0x1 Disables the interface protect circuit	RW	0x0

Bit	Name	Description	Access	Reset
24	`indicator`	Mode:Host only. Controls wether the Complement Output is qualified with the Internal Vbus Valid comparator before being used in the Vbus State in the RX CMD. **Value** **Description** 0x0 Complement Output signal is qualified with the Internal VbusValid comparator 0x1 Complement Output signal is not qualified with the Internal VbusValid comparator	RW	0x0
23	`complement`	Mode:Host only. Controls the PHY to invert the ExternalVbusIndicator inputsignal, generating the ComplementOutput. Please refer to the ULPI Spec for more detail. **Value** **Description** 0x0 PHY does not invert ExternalVbusIndicator signal 0x1 PHY does invert ExternalVbusIndicator signal	RW	0x0
22	`termseldlpulse`	Mode:Device only. This bit selects utmi_termselect to drive data line pulse during SRP. **Value** **Description** 0x0 Data line pulsing using utmi_txvalid 0x1 Data line pulsing using utmi_termsel	RW	0x0
21	`ulpiextvbusindicator`	Mode:Host only. This bit indicates to the ULPI PHY to use an external VBUS overcurrent indicator. **Value** **Description** 0x0 PHY uses internal VBUS valid comparator 0x1 PHY uses external VBUS valid comparator	RW	0x0
20	`ulpiextvbusdrv`	Mode:Host only. This bit selects between internal or external supply to drive 5V on VBUS, in ULPI PHY. **Value** **Description** 0x0 PHY drives VBUS using internal charge pump 0x1 PHY drives VBUS using external supply	RW	0x0

Bit	Name	Description	Access	Reset
19	`ulpiclksusm`	Mode:Host and Device. This bit sets the ClockSuspendM bit in the Interface Control register on the ULPI PHY. This bit applies only in serial or carkit modes. **Value** **Description** 0x0 PHY powers down internal clock during suspend 0x1 PHY does not power down internal clock	RW	0x0
18	`ulpiautores`	Mode:Host and Device. This bit sets the AutoResume bit in the Interface Control register on the ULPI PHY. **Value** **Description** 0x0 PHY does not use AutoResume feature 0x1 PHY uses AutoResume feature	RW	0x0
13:10	`usbtrdtim`	Mode: Device only. Sets the turnaround time in PHY clocks. Specifies the response time for a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM). The value is calculated for the minimum AHB frequency of 30 MHz. USB turnaround time is critical for certification where long cables and 5-Hubs are used, so If you need the AHB to run at less than 30 MHz, and If USB turnaround time is not critical, these bits can be programmed to a larger value. **Value** **Description** 0x9 MAC interface is 8-bit UTMI+.	RW	0x5
9	`hnpcap`	Mode:Host and Device. The application uses this bit to control the otg core's HNP capabilities. **Value** **Description** 0x0 HNP capability is not enabled. 0x1 HNP capability is enabled	RW	0x0

Bit	Name	Description	Access	Reset
8	`srpcap`	Mode:Host and Device. The application uses this bit to control the otg core SRP capabilities. If the core operates as a non-SRP-capable B-device, it cannot request the connected A-device (host) to activate VBUS and start a session. This bit is writable only If an SRP mode was specified for Mode of Operation in coreConsultant (parameter OTG_MODE). Otherwise, reads Return 0. **Value** **Description** 0x0 SRP capability is not enabled 0x1 SRP capability is enabled	`RW`	`0x0`
7	`ddrsel`	Mode:Host and Device. The application uses this bit to select a Single Data Rate (SDR) or Double Data Rate (DDR) or ULPI interface. **Value** **Description** 0x0 Single Data Rate ULPI Interfacewith 8-bit-wide data bus 0x1 Double Data Rate ULPI Interface, with 4-bit-wide data bus	`RW`	`0x0`
6	`physel`	Mode:Host and Device. The application uses USB 2.0. **Value** **Description** 0x0 USB 2.0 high-speed ULPI	`RO`	`0x0`
5	`fsintf`	Mode:Host and Device. The application can Set this bit to select between the 3- and 6-pin interfaces, and access is Read and Write. **Value** **Description** 0x0 6-pin unidirectional full-speed serial interface 0x1 3-pin bidirectional full-speed serial interface	`RO`	`0x0`
4	`ulpi_utmi_sel`	Mode:Host and Device. The application uses ULPI Only in 8bit mode. **Value** **Description** 0x0 ULPI PHY	`RO`	`0x1`

Bit	Name	Description	Access	Reset
3	phyif	Mode:Host and Device. This application uses a ULPI interface only. Hence only 8-bit setting is relevant. This setting should not matter since UTMI is not enabled. **Value** — **Description** 0x0 — PHY 8bit Mode	RO	0x0
2:0	toutcal	Mode:Host and Device. The number of PHY clocks that the application programs in this field is added to the high-speed/full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the linestate condition can vary from one PHY to another. The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are: High-speed operation: -One 30-MHz PHY clock = 16 bit times -One 60-MHz PHY clock = 8 bit times Full-speed operation: -One 30-MHz PHY clock = 0.4 bit times -One 60-MHz PHY clock = 0.2 bit times -One 48-MHz PHY clock = 0.25 bit times	RW	0x0

grstctl

The application uses this register to reset various hardware features inside the core

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00010
usb1	0xFFB40000	0xFFB40010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ahbidle RO 0x1	dmareq RO 0x0	Reserved													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					txfnum RW 0x0					txfflsh RO 0x0	rxfflsh RO 0x0	Reserved	frmcntrrst RO 0x0	Reserved	csftrst RO 0x0

grstctl Fields

Bit	Name	Description	Access	Reset
31	ahbidle	Mode:Host and Device. Indicates that the AHB Master State Machine is in the IDLE condition. **Value** — **Description** 0x0 — Not Idle 0x1 — AHB Master Idle	RO	0x1
30	dmareq	Mode:Host and Device. Indicates that the DMA request is in progress. Used for debug. **Value** — **Description** 0x0 — No DMA request 0x1 — DMA request is in progress	RO	0x0

Bit	Name	Description	Access	Reset
10:6	`txfnum`	Mode:Host and Device. This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit. **Value** **Description** 0x0 - Non-periodic TxFIFO flush in Host mode - Non-periodic TxFIFO flush in device mode when in shared FIFO operation 0x1 - Periodic TxFIFO flush in Host mode - Periodic TxFIFO 1 flush in Device mode when in sharedFIFO operation 0x2 - Periodic TxFIFO 2 flush in Device mode when in sharedFIFO operation- TXFIFO 2 flush in device mode when in dedicated FIFO mode 0xf - Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation - TXFIFO 15 flush in device mode when in dedicated FIFO mode 0x10 Flush all the transmit FIFOs in device or host mode.	`RW`	`0x0`
5	`txfflsh`	Mode:Host and Device. This bit selectively flushes a single or all transmit FIFOs, but cannot do so If the core is in the midst of a transaction. The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers: ReadNAK Effective Interrupt ensures the core is notreading from the FIFO WriteGRSTCTL.AHBIdle ensures the core is not writinganything to the FIFO. Flushing is normally recommended when FIFOs are reconfigured or when switching between Shared FIFO and Dedicated Transmit FIFO operation. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk. **Value** **Description** 0x0 No Flush 0x1 selectively flushes a single or all transmit FIFOs	`RO`	`0x0`

Bit	Name	Description	Access	Reset
4	`rxfflsh`	Mode:Host and Device. The application can flush the entire RxFIFO using this bit, but must first ensure that the core is not in the middle of a transaction. The application must only write to this bit after checking that the core is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear. **Value** **Description** 0x0 no flush the entire RxFIFO 0x1 flush the entire RxFIFO	`RO`	`0x0`
2	`frmcntrrst`	Mode:Host only. The application writes this bit to reset the (micro)frame number counter inside the core. When the (micro)frame counter is reset, the subsequent SOF sent out by the core has a (micro) frame number of 0. When application writes 1 to the bit, it might not be able to read back the value as it will get cleared by the core in a few clock cycles. **Value** **Description** 0x0 No reset 0x1 Host Frame Counter Reset	`RO`	`0x0`

Bit	Name	Description	Access	Reset
0	csftrst	Mode:Host and Device. Resets the hclk and phy_clock domains as follows:Clears the interrupts and all the CSR registers except the following register bits: - PCGCCTL.RstPdwnModule - PCGCCTL.GateHclk - PCGCCTL.PwrClmp - PCGCCTL.StopPPhyLPwrClkSelclk - GUSBCFG.PhyLPwrClkSel - GUSBCFG.DDRSel - GUSBCFG.PHYSel - GUSBCFG.FSIntf - GUSBCFG.ULPI_UTMI_Sel - GUSBCFG.PHYIf - HCFG.FSLSPclkSel - DCFG.DevSpd - GGPIO - GPWRDN - GADPCTL All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. Any transactions on the AHB Master are terminated as soonas possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. When Hibernation or ADP feature is enabled, the PMU module is not reset by the Core Soft Reset.The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also must check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation.Typically software reset is used during software development and also when you dynamically change the PHY selection bits in the USB configuration registers listed above. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation. **Value** / **Description** 0x0 No reset 0x1 Resets hclk and phy_clock domains	RO	0x0

gintsts

This register interrupts the application for system-level events in the current mode (Device mode or Host mode). Some of the bits in this register are valid only in Host mode, while others are valid in Device mode only. This register also indicates the current mode. To clear the interrupt status bits of type R_SS_WC, the application must write 1 into the bit. The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically. The application must clear the GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00014
usb1	0xFFB40000	0xFFB40014

Offset: 0x14

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wkupint RO 0x0	sessreqint RO 0x0	disconnint RO 0x0	ConIDStsChng RO 0x1	Reserved	ptxfemp RO 0x1	hchint RO 0x0	prtint RO 0x0	resetdet RO 0x0	fetsusp RO 0x0	incomplp RO 0x0	incompisoin RO 0x0	oepint RO 0x0	iepint RO 0x0	epmis RO 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	isooutdrop RO 0x0	enumdone RO 0x0	usbrst RO 0x0	usbsusp RO 0x0	erlysusp RO 0x0	Reserved		goutnakeff RO 0x0	ginnakeff RO 0x0	Reserved	rxflvl RO 0x0	sof RO 0x0	otgint RO 0x0	modemis RO 0x0	curmod RO 0x0

gintsts Fields

Bit	Name	Description	Access	Reset
31	wkupint	Mode:Host and Device. Wakeup Interrupt during Suspend(L2) or LPM(L1) state. -During Suspend(L2): - Device Mode - This interrupt is asserted only when Host Initiated Resume is detected on USB. - Host Mode - This interrupt is asserted only when Device Initiated Remote Wakeup is detected on USB - During LPM(L1):- - Device Mode - This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB. - Host Mode - This interrupt is asserted for either Host Initiated Resume or Device Initiated Remote Wakeup on USB. **Value** **Description** 0x0 Not active 0x1 Resume Remote Wakeup Detected Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
30	`sessreqint`	Mode:Host and Device. In Host mode, this interrupt is asserted when a session request is detected from the device. In Host mode, this interrupt is asserted when a session request is detected from the device. In Device mode, this interrupt is asserted when the utmisrp_bvalid signal goes high. This bit can be set only by the core and the application should write 1 to clear. **Value** **Description** 0x0 Not active 0x1 Session Request New Session Detected Interrupt	RO	0x0
29	`disconnint`	Mode:Host only. Asserted when a device disconnect is detected. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 Not active 0x1 Disconnect Detected Interrupt	RO	0x0
28	`ConIDStsChng`	Mode:Host and Device. The core sets this bit when there is a change in connector ID status. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 Not Active 0x1 Connector ID Status Change	RO	0x1
26	`ptxfemp`	Mode:Host only. This interrupt is asserted when the Periodic Transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the Periodic Request Queue. The half or completely empty status is determined by the Periodic TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.PTxFEmpLvl). **Value** **Description** 0x0 Not active 0x1 Periodic TxFIFO Empty	RO	0x1

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>25</td><td>hchint</td><td>Mode:Host only. The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in Host mode). The application must read the Host All Channels Interrupt (HAINT) register to determine the exact number of the channel on which the interrupt occurred, and Then read the corresponding Host Channel-n Interrupt (HCINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the HCINTn register to clear this bit.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Not active</td></tr>
<tr><td>0x1</td><td>Host Channels Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>24</td><td>prtint</td><td>Mode:Host only. The core sets this bit to indicate a change in port status of one of the otg core ports in Host mode. The application must read the Host Port Control and Status (HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the Host PC Control and Status register to clear this bit.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td></td></tr>
<tr><td>0x1</td><td>Host Port Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>23</td><td>resetdet</td><td>Mode: Device only. In Device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in Suspend. In Host mode, this interrupt is not asserted.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Not active</td></tr>
<tr><td>0x1</td><td>Reset detected Interrup</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
22	`fetsusp`	Mode: Device only. This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or Request Queue space. This interrupt is used by the application for an endpoint mismatch algorithm. for example, after detecting an endpoint mismatch, the application: - Sets a Global non-periodic IN NAK handshake - Disables In endpoints - Flushes the FIFO - Determines the token sequence from the IN Token Sequence Learning Queue - Re-enables the endpoints - Clears the Global non-periodic IN NAK handshake If the Global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an IN token received when FIFO empty interrupt. The OTG Then sends the host a NAK response. To avoid this scenario, the application can check the GINTSTS.FetSusp interrupt, which ensures that the FIFO is full before clearing a Global NAK handshake. Alternatively, the application can mask the "IN token received when FIFO empty" interrupt when clearing a Global IN NAKhandshake. **Value** — **Description** 0x0 — Not active 0x1 — Data Fetch Suspended	RO	0x0
21	`incomplp`	Mode: Device only. In Host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending which arescheduled for the current microframe. Incomplete Isochronous OUT Transfer (incompISOOUT) The Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register. This bit can be set only by the core and the application should write 1 to clear it **Value** — **Description** 0x0 — Not active 0x1 — Incomplete Periodic Transfer	RO	0x0

Bit	Name	Description	Access	Reset
20	`incompisoin`	Mode: Device only. The core sets this interrupt to indicate that there is at least isochronous IN endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register. This interrupt is not asserted in Scatter/Gather DMA mode. **Value** **Description** 0x0 Not active 0x1 Incomplete Isochronous IN Transfer	`RO`	`0x0`
19	`oepint`	Mode: Device only. The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and Then read the corresponding Device OUT Endpoint-n Interrupt (DOEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DOEPINTn register to clear this bit. **Value** **Description** 0x0 Not active 0x1 OUT Endpoints Interrupt	`RO`	`0x0`
18	`iepint`	Mode: Device only. The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (DAINT) register to determine the exact number of the IN endpoint on Device IN Endpoint-n Interrupt (DIEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding DIEPINTn register to clear this bit. **Value** **Description** 0x0 Not active 0x1 IN Endpoints Interrupt	`RO`	`0x0`

Bit	Name	Description	Access	Reset
17	`epmis`	Mode: Device only. This interrupt is valid only in shared FIFO operation. Indicates that an IN token has been received for a non-periodic endpoint, but the data for another endpoint is present in the top of the Non-periodic Transmit FIFO and the IN endpoint mismatch count programmed by the application has expired. **Value** **Description** 0x0 Not active 0x1 Endpoint Mismatch Interrup	RO	0x0
14	`isooutdrop`	Mode: Device only. The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint. **Value** **Description** 0x0 Not active 0x1 Isochronous OUT Packet Dropped Interrup	RO	0x0
13	`enumdone`	Mode: Device only. The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status register to obtain the enumerated speed. **Value** **Description** 0x0 Not active 0x1 Enumeration Done	RO	0x0
12	`usbrst`	Mode: Device only. The core sets this bit to indicate that a reset is detected on the USB. **Value** **Description** 0x0 Not active 0x1 USB Reset	RO	0x0

Bit	Name	Description	Access	Reset
11	`usbsusp`	Mode: Device only. The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal for an extended period of time. **Value** **Description** 0x0 Not Active 0x1 USB Suspend	RO	0x0
10	`erlysusp`	Mode: Device only. The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms. **Value** **Description** 0x0 No Idle 0x1 Idle state detecetd	RO	0x0
7	`goutnakeff`	Mode: Device only. Indicates that the Set Global OUT NAK bit in the Device Control register (DCTL.SGOUTNak), Set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (DCTL.CGOUTNak). **Value** **Description** 0x0 No Active 0x1 Global OUT NAK Effective	RO	0x0
6	`ginnakeff`	Mode: Device only. Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (DCTL.SGNPInNak), Set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit Set by the application. This bit can be cleared by clearing the Clear Global Non-periodic IN NAK bit in the Device Control register (DCTL.CGNPInNak). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit. **Value** **Description** 0x0 Not active 0x1 Set Global Non-periodic IN NAK bi	RO	0x0

Bit	Name	Description	Access	Reset
4	rxflvl	Mode: Host and Device. Indicates that there is at least one packet pending to be read from the RxFIFO. **Value** **Description** 0x0 Not Active 0x1 Rx Fifo Non Empty	RO	0x0
3	sof	Mode: Host and Device. In Host mode, the core sets this bit to indicate that an SOF (FS), micro-SOF (HS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt. In Device mode, the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current (micro)Frame number. This interrupt is seen only when the core is operating at either HS or FS. This bit can be set only by the core and the application should write 1 to clear it. This register may return 1 if read immediately after power on reset. If the register bit reads 1 immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit. **Value** **Description** 0x0 No sof 0x1 Start of Frame	RO	0x0
2	otgint	Mode: Host and Device. The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the GOTGINT register to clear this bit. **Value** **Description** 0x0 No Interrupt 0x1 OTG Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	modemis	Mode: Host and Device. The core sets this bit when the application is trying to access: -A Host mode register, when the core is operating in Device mode. -A Device mode register, when the core is operating in Host mode. The register access is completed on the AHB with an OKAYresponse, but is ignored by the core internally and does not affect the operation of the core. This bit can be set only by the core and the application should write 1 to clearit **Value** — **Description** 0x0 — No Mode Mismatch Interrupt 0x1 — Mode Mismatch Interrupt	RO	0x0
0	curmod	Mode: Host and Device. Indicates the current mode. **Value** — **Description** 0x0 — Device mode 0x1 — Host mode	RO	0x0

gintmsk

This register works with the Interrupt Register (GINTSTS) to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the GINTSTS register bit corresponding to that interrupt is still set.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00018
usb1	0xFFB40000	0xFFB40018

Offset: 0x18

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wkupintmsk RW 0x0	sessreqintmsk RW 0x0	disconnintmsk RW 0x0	conidstschngmsk RW 0x0	Reserved	ptxfempmsk RW 0x0	hchintmsk RW 0x0	prtintmsk RW 0x0	resetdetmsk RW 0x0	fetsuspmsk RW 0x0	incomplpmsk RW 0x0	incompisoinmsk RW 0x0	oepintmsk RW 0x0	iepintmsk RW 0x0	epmismsk RW 0x0	Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
eopfmsk RW 0x0	isooutdropmsk RW 0x0	enumdonemsk RW 0x0	usbrstmsk RW 0x0	usbsuspmsk RW 0x0	erlysuspmsk RW 0x0	Reserved		goutnakeffmsk RW 0x0	ginnakeffmsk RW 0x0	Reserved	rxflvlmsk RW 0x0	sofmsk RW 0x0	otgintmsk RW 0x0	modemismsk RW 0x0	Reserved

gintmsk Fields

Bit	Name	Description	Access	Reset
31	wkupintmsk	Mode: Host and Device. **Value** **Description** 0x0 Resume Remote Wakeup Detected Interrupt Mask 0x1 No maskResume Remote Wakeup Detected Interrupt	RW	0x0
30	sessreqintmsk	Mode: Host and Device. **Value** **Description** 0x0 Session Request New Session Detected Interrupt Mask 0x1 No mask Session RequestNew Session Detected Interrupt	RW	0x0
29	disconnintmsk	Mode: Host and Device. **Value** **Description** 0x0 Disconnect Detected Interrupt Mask 0x1 No mask Disconnect Detected Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
28	conidstschngmsk	Mode: Host and Device. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 Connector ID Status Change Mask 0x1 No mask Connector ID Status Change	RW	0x0
26	ptxfempmsk	Mode: Host only. **Value** **Description** 0x0 Periodic TxFIFO Empty Mask 0x1 No mask Periodic TxFIFO Empty	RW	0x0
25	hchintmsk	Mode: Host only. **Value** **Description** 0x0 Host Channels Interrupt Mask 0x1 No mask Host Channels Interrupt	RW	0x0
24	prtintmsk	Mode: Host only. **Value** **Description** 0x0 Host Port Interrupt Mask 0x1 No mask Host Port Interrupt	RW	0x0
23	resetdetmsk	Mode: Device only. **Value** **Description** 0x0 Reset detected Interrupt Mask 0x1 No mask Reset detected Interrupt	RW	0x0
22	fetsuspmsk	Mode: Device only. **Value** **Description** 0x0 Data Fetch Suspended Mask 0x1 No mask Data Fetch Suspended	RW	0x0

Bit	Name	Description	Access	Reset
21	incomplpmsk	Mode: Host only. Value Description 0x0 Incomplete Periodic Transfer Mask 0x1 No mask Incomplete Periodic Transfer	RW	0x0
20	incompisoinmsk	Mode: Device only. Value Description 0x0 Incomplete Isochronous IN Transfer Mask 0x1 No mask Incomplete Isochronous IN Transfer	RW	0x0
19	oepintmsk	Mode: Device only. Value Description 0x0 OUT Endpoints Interrupt Mask 0x1 No mask OUT Endpoints Interrupt	RW	0x0
18	iepintmsk	Mode: Device only. Value Description 0x0 IN Endpoints Interrupt Mask 0x1 No mask IN Endpoints Interrupt	RW	0x0
17	epmismsk	Mode: Device only. Value Description 0x0 Endpoint Mismatch Interrupt Mask 0x1 No mask Endpoint Mismatch Interrupt	RW	0x0
15	eopfmsk	Mode: Device only. End of Periodic Frame Interrupt Mask (EOPFMsk) Value Description 0x0 End of Periodic Frame Interrupt Mask 0x1 No mask End of Periodic Frame Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
14	isooutdropmsk	Mode: Device only. **Value** / **Description** 0x0 Isochronous OUT Packet Dropped Interrupt Mask 0x1 No mask Isochronous OUT Packet Dropped Interrupt	RW	0x0
13	enumdonemsk	Mode: Device only. **Value** / **Description** 0x0 Enumeration Done Mask 0x1 No mask Enumeration Done	RW	0x0
12	usbrstmsk	Mode: Device only. **Value** / **Description** 0x0 USB Reset Mask 0x1 No mask USB Reset	RW	0x0
11	usbsuspmsk	Mode: Device only. **Value** / **Description** 0x0 USB Suspend Mask 0x1 No mask USB Suspend	RW	0x0
10	erlysuspmsk	Mode: Device only. **Value** / **Description** 0x0 Early Suspend Mask 0x1 No mask Early Suspend Mask	RW	0x0
7	goutnakeffmsk	Mode: Device only. **Value** / **Description** 0x0 Global OUT NAK Effective Mask 0x1 No mask Global OUT NAK Effective	RW	0x0

Bit	Name	Description	Access	Reset
6	`ginnakeffmsk`	Mode: Device only. **Value** **Description** 0x0 Global Non-periodic IN NAK Effective Mask 0x1 No mask Global Non-periodic IN NAK Effective	RW	0x0
4	`rxflvlmsk`	Mode: Host and Device. **Value** **Description** 0x0 Receive FIFO Non-Empty Mask 0x1 No maks Receive FIFO Non-Empty	RW	0x0
3	`sofmsk`	Mode: Host and Device. **Value** **Description** 0x0 Start of Frame Mask 0x1 No Mask Start of Frame	RW	0x0
2	`otgintmsk`	Mode: Host and Device. **Value** **Description** 0x0 OTG Interrupt Mask 0x1 No mask OTG Interrupt	RW	0x0
1	`modemismsk`	Mode: Host and Device. **Value** **Description** 0x0 Mode Mismatch Interrupt Mask 0x1 No Mask Mode Mismatch Interrupt	RW	0x0

grxstsr

A read to the Receive Status Read and Pop register additionally pops the: top data entry out of the RxFIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted. Use of these fields vary based on whether the HS OTG core is functioning as a host or a device. Do not read this register's reset value before configuring the core because the read value is "X" in the simulation.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0001C
usb1	0xFFB40000	0xFFB4001C

Offset: 0x1C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											pktsts RO 0x0				dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dpid RO 0x0	bcnt RO 0x0											chnum RO 0x0			

grxstsr Fields

Bit	Name	Description	Access	Reset
20:17	pktsts	Mode: Host only. Others: Reserved. Indicates the status of the received packet **Value** **Description** 0x2 IN data packet received 0x3 IN transfer completed (triggers an interrupt 0x5 Data toggle error (triggers an interrupt) 0x7 Channel halted (triggers an interrupt)	RO	0x0
16:15	dpid	Indicates the Data PID of the received packet. **Value** **Description** 0x0 DATA0 0x2 DATA1 0x1 DATA2 0x3 MDATA	RO	0x0
14:4	bcnt	Indicates the byte count of the received data packet.	RO	0x0
3:0	chnum	Indicates the endpoint number to which the current received packet belongs.	RO	0x0

grxstsp

A read to the Receive Status Read and Pop register additionally pops the: top data entry out of the RxFIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (GINTSTS.RxFLvl) is asserted. Use of these fields vary based on whether the HS OTG core is functioning as a host or a device. Do not read this register'ss reset value before configuring the core because the read value is "X" in the simulation.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00020`
`usb1`	`0xFFB40000`	`0xFFB40020`

Offset: `0x20`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							fn RO 0x0				pktsts RO 0x0				dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dpid RO 0x0	bcnt RO 0x0											chnum RO 0x0			

grxstsp Fields

Bit	Name	Description	Access	Reset
24:21	`fn`	Mode: Device only. This is the least significant 4 bits of the (micro)Frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.	RO	0x0
20:17	`pktsts`	Mode: Host only. Others: Reserved. Indicates the status of the received packet **Value** **Description** 0x0 DATA0 0x2 DATA1 0x1 DATA2 0x3 MDATA	RO	0x0

Bit	Name	Description	Access	Reset
16:15	`dpid`	Indicates the Data PID of the received OUT data packet. **Value** — **Description** 0x0 — DATA0 0x2 — DATA1 0x1 — DATA2 0x3 — MDATA	RO	0x0
14:4	`bcnt`	Mode: Host only. Indicates the byte count of the received IN data packet.	RO	0x0
3:0	`chnum`	Mode: Host only. Indicates the channel number to which the current received packet belongs.	RO	0x0

grxfsiz

The application can program the RAM size that must be allocated to the RxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00024
usb1	0xFFB40000	0xFFB40024

Offset: `0x24`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		rxfdep RW 0x2000													

grxfsiz Fields

Bit	Name	Description	Access	Reset
13:0	`rxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 32,768 The power-on reset value of this register is specified as the Largest Rx Data FIFO Dept 8192. Using the Dynamic FIFO Sizing, you can write a new value in this field. Programmed values must not exceed 8192.	RW	0x2000

gnptxfsiz

The application can program the RAM size and the memory start address for the Non-periodic TxFIFO. The fields of this register change, depending on host or device mode.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00028
usb1	0xFFB40000	0xFFB40028

Offset: 0x28

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		nptxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		nptxfstaddr RW 0x2000													

gnptxfsiz Fields

Bit	Name	Description	Access	Reset
29:16	nptxfdep	Mode: Host only. for host mode, this field is always valid. The application can write a new value in this field. Programmed values must not exceed 8192	RW	0x2000
13:0	nptxfstaddr	Mode: Host only. for host mode, this field is always valid.This field contains the memory start address for Non-periodic Transmit FIFO RAM. This field is set from 16-8192 32 bit words. The application can write a new value in this field. Programmed values must not exceed 8192.	RW	0x2000

gnptxsts

In Device mode, this register is valid only in Shared FIFO operation. It contains the free space information for the Non-periodic TxFIFO and the Nonperiodic Transmit RequestQueue

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0002C
usb1	0xFFB40000	0xFFB4002C

Offset: 0x2C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	nptxqtop RO 0x0							nptxqspcavail RO 0x8							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nptxfspcavail RO 0x400															

gnptxsts Fields

Bit	Name	Description	Access	Reset
30:24	nptxqtop	Entry in the Non-periodic Tx Request Queue that is currently being processed by the MAC. -Bits [30:27]: Channel/endpoint number -Bits [26:25]: -Bit [24]: Terminate (last Entry for selected channel endpoint) **Value** **Description** 0x0 IN/OUT token 0x1 Zero-length transmit packet (device IN/host OUT) 0x2 PING/CSPLIT token 0x3 Channel halt command	RO	0x0
23:16	nptxqspcavail	Indicates the amount of free space available in the Non-periodic Transmit Request Queue. This queue holds both IN and OUT requests in Host mode. Device mode has only IN requests. -Others: Reserved **Value** **Description** 0x0 Non-periodic Transmit Request Queue is full 0x1 1 location available 0x2 2 locations available 0x3 3 locations available 0x4 4 locations available 0x5 5 locations available 0x6 6 locations available 0x7 7 locations available 0x8 8 locations available	RO	0x8

Bit	Name	Description	Access	Reset
15:0	nptxfspcavail	Indicates the amount of free space available in the Non-periodic TxFIFO.Values are in terms of 32-bit words. 16h0: Non-periodic TxFIFO is full 16h1: 1 word available 16h2: 2 words available 16hn: n words available (where 0 n 32,768) 16h8000: 32,768 words available Others: Reserved	RO	0x400

gpvndctl

The application can use this register to access PHY registers. for a ULPI PHY, the core uses the ULPI interface for PHY register access. The application sets Vendor Control register for PHY register access and times the PHY register access. The application polls the VStatus Done bit in this register for the completion of the PHY register access

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00034
usb1	0xFFB40000	0xFFB40034

Offset: 0x34

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>disulpidrvr
RO 0x0</td><td colspan="3">Reserved</td><td>vstsdone
RO
0x0</td><td>vstsbsy
RO
0x0</td><td>newregreq
RO
0x0</td><td colspan="2">Reserved</td><td>regwr
RW
0x0</td><td colspan="6">regaddr
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">vctrl
RW 0x0</td><td colspan="8">regdata
RW 0x0</td></tr>
</table>

gpvndctl Fields

Bit	Name	Description	Access	Reset
31	disulpidrvr	The application sets this bit when it has finished processing the ULPI Carkit Interrupt (GINTSTS.ULPICKINT). When Set, the otg core disables drivers for output signals and masks input signal for the ULPI interface. otg clears this bit before enabling the ULPI interface. **Value** **Description** 0x0 ULPI ouput signals 0x1 Disable ULPI ouput signals	RO	0x0

Bit	Name	Description	Access	Reset
27	`vstsdone`	The core sets this bit when the vendor control access isdone. This bit is cleared by the core when the application sets the New Register Request bit (bit 25). **Value** / **Description** 0x0 VStatus Done inactive 0x1 VStatus Done active	RO	0x0
26	`vstsbsy`	The core sets this bit when the vendor control access is in progress and clears this bit when done. **Value** / **Description** 0x0 VStatus Busy inactive 0x1 VStatus Busy active	RO	0x0
25	`newregreq`	The application sets this bit for a new vendor controlaccess. **Value** / **Description** 0x0 New Register Request not active 0x1 New Register Request active	RO	0x0
22	`regwr`	Set this bit for register writes, and clear it for register reads. **Value** / **Description** 0x0 Register Write 0x1 Register Write	RW	0x0
21:16	`regaddr`	The 6-bit PHY register address for immediate PHY Register Set access. Set to 0x2F for Extended PHY Register Set access.	RW	0x0
15:8	`vctrl`	The 4-bit register address a vendor defined 4-bit parallel output bus. ULPI Extended Register Address and the 6-bit PHY extended register address.	RW	0x0
7:0	`regdata`	Contains the write data for register write. Read data for register read, valid when VStatus Done is Set.	RW	0x0

ggpio

The application can use this register for general purpose input/output ports or for debugging.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00038
usb1	0xFFB40000	0xFFB40038

Offset: 0x38

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpo RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpi RO 0x0															

ggpio Fields

Bit	Name	Description	Access	Reset
31:16	gpo	This field is driven as an output from the core, gp_o[15:0]. The application can program this field to determine the corresponding value on the gp_o[15:0] output.	RW	0x0
15:0	gpi	This field's read value reflects the gp_i[15:0] core input value.	RO	0x0

guid

This is a read/write register containing the User ID. This register can be used in the following ways: -To store the version or revision of your system -To store hardware configurations that are outside the otg core As a scratch register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0003C
usb1	0xFFB40000	0xFFB4003C

Offset: 0x3C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
guid RW 0x12345678															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
guid RW 0x12345678															

guid Fields

Bit	Name	Description	Access	Reset
31:0	guid	Application-programmable ID field.	RW	0x12345678

gsnpsid

This read-only register contains the release number of the core being used.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00040
usb1	0xFFB40000	0xFFB40040

Offset: 0x40

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gsnpsid RO 0x4F54293A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gsnpsid RO 0x4F54293A															

gsnpsid Fields

Bit	Name	Description	Access	Reset
31:0	gsnpsid	Release number of the otg core being used is currently OTG 2.93a	RO	0x4F54293A

ghwcfg1

This register contains the logical endpoint direction(s).

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00044
usb1	0xFFB40000	0xFFB40044

Offset: 0x44

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ghwcfg1 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ghwcfg1 RO 0x0															

ghwcfg1 Fields

Bit	Name	Description	Access	Reset
31:0	ghwcfg1	This 32-bit field uses two bits per endpoint to determine the endpoint direction. Endpoint -Bits [31:30]: Endpoint 15 direction -Bits [29:28]: Endpoint 14 direction ... -Bits [3:2]: Endpoint 1 direction -Bits[1:0]: Endpoint 0 direction (always BIDIR) **Value** **Description** 0x0 BIDIR (IN and OUT) endpoint 0x1 IN endpoint 0x2 OUT endpoint 0x3 Reserved	RO	0x0

ghwcfg2

This register contains configuration options.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00048
usb1	0xFFB40000	0xFFB40048

Offset: 0x48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	tknqdepth RO 0x8					ptxqdepth RO 0x0		nptxqdepth RO 0x2		Reserved	multiprocintrpt RO 0x0	dynfifosizing RO 0x1	periosupport RO 0x1	numhstchnl RO 0xF	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
numhstchnl RO 0xF		numdeveps RO 0xF				fsphytype RO 0x0		hsphytype RO 0x2		singpnt RO 0x0	otgarch RO 0x2		otgmode RO 0x0		

ghwcfg2 Fields

Bit	Name	Description	Access	Reset
30:26	`tknqdepth`	Range: 0 to 30.	RO	0x8
25:24	`ptxqdepth`	Specifies the Host mode Periodic Request Queue depth.That is, the maximum number of packets that can reside in the Host Periodic TxFIFO. This queue holds one entry corresponding to each IN or OUT periodic request. This queue is 9 bits wide. **Value** — **Description** 0x0 — Que Depth 2 0x1 — Que Depth 4 0x2 — Que Depth 8 0x3 — Que Depth 16	RO	0x0
23:22	`nptxqdepth`	Specifies the Non-periodic Request Queue depth, the maximum number of packets that can reside in the Non-periodic TxFIFO. In Device mode, the queue is used only in Shared FIFO Mode (Enable Dedicated Transmit FIFO for device IN Endpoints? =No). In this mode, there is one entry in the Non-periodic Request Queue for each packet in the Non-periodic TxFIFO. In Host mode, this queue holds one entry corresponding to each IN or OUT nonperiodic request. This queue is seven bits wide. **Value** — **Description** 0x0 — Que size 2 0x1 — Que size 4 0x2 — Que size 8	RO	0x2
20	`multiprocintrpt`	Not implemented. **Value** — **Description** 0x0 — No Multi Processor Interrupt Enabled	RO	0x0
19	`dynfifosizing`	Feature supported. **Value** — **Description** 0x1 — Dynamic FIFO Sizing Enabled	RO	0x1

Bit	Name	Description	Access	Reset
18	`periosupport`	Feature supported. **Value** **Description** 0x1 Periodic OUT Channels Supported in Host Mode Supported	RO	0x1
17:14	`numhstchnl`	Indicates the number of host channels supported by the core in Host mode. **Value** **Description** 0x0 Host Channel 1 0x1 Host Channel 2 0x2 Host Channel 3 0x3 Host Channel 4 0x4 Host Channel 5 0x5 Host Channel 6 0x6 Host Channel 7 0x7 Host Channel 8 0x8 Host Channel 9 0x9 Host Channel 10 0xa Host Channel 11 0xb Host Channel 12 0xc Host Channel 13 0xd Host Channel 14 0xe Host Channel 15 0xf Host Channel 16	RO	0xF

Bit	Name	Description	Access	Reset
13:10	`numdeveps`	The number of endpoints is 1 to 15 in Device mode in addition to control endpoint 0. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RO	0xF
9:8	`fsphytype`	Specifies the Full Speed PHY in use. **Value** **Description** 0x2 ULPI Type	RO	0x0
7:6	`hsphytype`	Specifies the High Speed PHY in use. **Value** **Description** 0x0 High-Speed interface not supported 0x2 ULPI	RO	0x2
5	`singpnt`	Single Point Only. **Value** **Description** 0x1 Single-point applicatio	RO	0x0

Bit	Name	Description	Access	Reset
4:3	`otgarch`	DMA Architecture. **Value** **Description** 0x2 Internal DMA	`RO`	`0x2`
2:0	`otgmode`	HNP- and SRP-Capable OTG (Device and Host). **Value** **Description** 0x0 HNP- and SRP-Capable OTG (Host & Device 0x1 SRP-Capable OTG (Host & Device) 0x2 Non-HNP and Non-SRP Capable OTG (Host & Device) 0x3 SRP-Capable Device 0x4 Non-OTG Device 0x5 SRP-Capable Host 0x6 Non-OTG Host	`RO`	`0x0`

ghwcfg3

This register contains the configuration options.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB0004C`
`usb1`	`0xFFB40000`	`0xFFB4004C`

Offset: `0x4C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dfifodepth RO 0x1F80															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lpmmode RO 0x0	bcsupport RO 0x0	hsicmode RO 0x0	adpsupport RO 0x0	rsttype RO 0x0	optfeature RO 0x0	vndctlsupt RO 0x1	i2cintsel RO 0x0	otgen RO 0x1	pktsizewidth RO 0x6			xfersizewidth RO 0x8			

ghwcfg3 Fields

Bit	Name	Description	Access	Reset
31:16	dfifodepth	DFIFO Depth. This value is in terms of 35-bit words. Minimum value is 32 Maximum value is 8192	RO	0x1F80
15	lpmmode	LPM Mode Enabled/Disabled. **Value** **Description** 0x0 LPM disabled	RO	0x0
14	bcsupport	Battery Charger Support. **Value** **Description** 0x0 No Battery Charger Support	RO	0x0
13	hsicmode	Supports HSIC and Non-HSIC Modes. **Value** **Description** 0x0 Non-HSIC-capable	RO	0x0
12	adpsupport	ADP logic support. **Value** **Description** 0x1 ADP logic is present along with HSOTG controller	RO	0x0
11	rsttype	Defines what reset type is used in the core. **Value** **Description** 0x0 Asynchronous reset is used in the core	RO	0x0
10	optfeature	User ID register, GPIO interface ports, and SOF toggle and counter ports were removed. **Value** **Description** 0x0 No Optional features	RO	0x0
9	vndctlsupt	ULPI PHY internal registers can be accessed by software using register reads/writes to otg **Value** **Description** 0x1 Vendor Control Interface is not available on the	RO	0x1

Bit	Name	Description	Access	Reset
8	`i2cintsel`	I2C Interface not used. **Value** **Description** 0x0 I2C Interface	RO	0x0
7	`otgen`	HNP and SRP Capable OTG (Device and Host) **Value** **Description** 0x1 OTG Capable	RO	0x1
6:4	`pktsizewidth`	**Value** **Description** 0x0 Width of Packet Size Counter 4 0x1 Width of Packet Size Counter 5 0x2 Width of Packet Size Counter 6 0x3 Width of Packet Size Counter 7 0x4 Width of Packet Size Counter 8 0x5 Width of Packet Size Counter 9 0x6 Width of Packet Size Counter 10	RO	0x6
3:0	`xfersizewidth`	Width variable from 11 to 19 bits. **Value** **Description** 0x0 Width of Transfer Size Counter 11 bits 0x1 Width of Transfer Size Counter 12 bits 0x2 Width of Transfer Size Counter 13 bits 0x3 Width of Transfer Size Counter 14 bits 0x4 Width of Transfer Size Counter 15 bits 0x5 Width of Transfer Size Counter 16 bits 0x6 Width of Transfer Size Counter 17 bits 0x7 Width of Transfer Size Counter 18 bits 0x8 Width of Transfer Size Counter 19 bits	RO	0x8

ghwcfg4

This register contains the configuration options.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00050`

Module Instance	Base Address	Register Address
usb1	0xFFB40000	0xFFB40050

Offset: 0x50

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dma RO 0x1	dma_configuration RO 0x1	ineps RO 0xF				dedfifomode RO 0x1	sessendfltr RO 0x0	bvalidfltr RO 0x0	avalidfltr RO 0x0	vbusvalidfltr RO 0x0	iddgfltr RO 0x0	numctleps RO 0xF			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
phydatawidth RO 0x0		Reserved							hibernation RO 0x0	ahbfreq RO 0x1	partialpwrdn RO 0x0	numdevperioeps RO 0x0			

ghwcfg4 Fields

Bit	Name	Description	Access	Reset
31	dma	Enable descriptor based scatter/gather DMA. When enabled, DMA operations will be serviced with descriptor based scatter/gather DMA **Value** **Description** 0x1 Dynamic configuration	RO	0x1
30	dma_configuration	Selects bewteen scatter and nonscatter configuration **Value** **Description** 0x0 Non-Scatter/Gather DMA configuration 0x1 Scatter/Gather DMA configuration	RO	0x1

Bit	Name	Description	Access	Reset
29:26	`ineps`	Number of Device Mode IN Endpoints Including Control. **Value** **Description** 0x0 In Endpoint 1 0x1 In Endpoint 2 0x2 In Endpoint 3 0x3 In Endpoint 4 0x4 In Endpoint 5 0x5 In Endpoint 6 0x6 In Endpoint 7 0x7 In Endpoint 8 0x8 In Endpoint 9 0x9 In Endpoint 10 0xa In Endpoint 11 0xb In Endpoint 12 0xc In Endpoint 13 0xd In Endpoint 14 0xe In Endpoint 15 0xf In Endpoint 16	RO	0xF
25	`dedfifomode`	Specifies whether Dedicated Transmit FIFOs should be enabled in device mode. **Value** **Description** 0x1 Dedicated Transmit FIFO Operation enabled	RO	0x1
24	`sessendfltr`	Specifies whether to add a filter on the session_end input from the PHY. **Value** **Description** 0x0 No filter	RO	0x0
23	`bvalidfltr`	Specifies whether to add a filter on the b_valid input from the PHY. **Value** **Description** 0x0 No Filter	RO	0x0

Bit	Name	Description	Access	Reset
22	`avalidfltr`	Specifies whether to add a filter on the b_valid input from the PHY. **Value** **Description** 0x0 No filter	RO	0x0
21	`vbusvalidfltr`	Vbus Valid Filter Enabled (VBusValidFltr) 0: No filter 1: Filter(coreConsultant parameter: OTG_EN_VBUSVALID_FILTER) **Value** **Description** 0x0 Vbus Valid Filter Disabled	RO	0x0
20	`iddgfltr`	Specifies whether to add a filter on the iddig input from the PHY. This is not relevant since we only support ULPI and there is no iddig pin exposed to I/O pads. **Value** **Description** 0x0 Iddig Filter Disabled	RO	0x0

Bit	Name	Description	Access	Reset
19:16	`numctleps`	Specifies the number of Device mode control endpoints in addition to control endpoint 0, which is always present. Range: 0-15. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	`RO`	`0xF`
15:14	`phydatawidth`	Uses a ULPI interface only. Hence only 8-bit setting is relevant. This setting should not matter since UTMI is not enabled.	`RO`	`0x0`
6	`hibernation`	Enables power saving mode hibernation. **Value** **Description** 0x0 Hibernation feature disabled	`RO`	`0x0`
5	`ahbfreq`	When the AHB frequency is less than 60 MHz, 4-deep clock-domain crossing sink and source buffers are instantiated between the MAC and the Packet FIFO Controller (PFC); otherwise, two-deep buffers are sufficient. **Value** **Description** 0x1 Minimum AHB Frequency Less Than 60 MH	`RO`	`0x1`

Bit	Name	Description	Access	Reset
4	partialpwrdn	Specifies whether to enable power optimization. **Value** **Description** 0x0 Partial Power Down disabled	RO	0x0
3:0	numdevperioeps	The maximum number of device IN operations is 16 active at any time including endpoint 0, which is always present. This parameter determines the number of device mode Tx FIFOs to be instantiated.	RO	0x0

gdfifocfg

Specifies whether Dedicated Transmit FIFOs should be enabled in device mode.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0005C
usb1	0xFFB40000	0xFFB4005C

Offset: 0x5C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epinfobaseaddr RW 0x1F80															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gdfifocfg RW 0x2000															

gdfifocfg Fields

Bit	Name	Description	Access	Reset
31:16	epinfobaseaddr	This field provides the start address of the EP info controller.	RW	0x1F80
15:0	gdfifocfg	This field is for dynamic programming of the DFIFO Size. This value takes effect only when the application programs a non zero value to this register. The otg core does not have any corrective logic if the FIFO sizes are programmed incorrectly.	RW	0x2000

hptxfsiz

This register holds the size and the memory start address of the Periodic TxFIFO

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00100

Module Instance	Base Address	Register Address
usb1	0xFFB40000	0xFFB40100

Offset: 0x100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		ptxfsize RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ptxfstaddr RW 0x4000														

hptxfsiz Fields

Bit	Name	Description	Access	Reset
29:16	ptxfsize	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 1024 The power-on reset value of this register is specified as the 1024.	RW	0x2000
14:0	ptxfstaddr	The power-on reset value of this register is the sum of the Largest Rx Data FIFO Depth and Largest Non-periodic Tx Data FIFO. Programmed values must not exceed the power-on value	RW	0x4000

dieptxf1

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00104
usb1	0xFFB40000	0xFFB40104

Offset: 0x104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	inepntxfstaddr RW 0x4000														

dieptxf1 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
14:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 1.	RW	0x4000

dieptxf2

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00108
usb1	0xFFB40000	0xFFB40108

Offset: `0x108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	inepntxfstaddr RW 0x6000														

dieptxf2 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192	RW	0x2000
14:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 2.	RW	0x6000

dieptxf3

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0010C
usb1	0xFFB40000	0xFFB4010C

Offset: `0x10C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x8000															

dieptxf3 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words.Minimum value is 16Maximum value is 8192	RW	0x2000
15:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 3.	RW	0x8000

dieptxf4

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00110
usb1	0xFFB40000	0xFFB40110

Offset: 0x110

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">inepntxfdep
RW 0x2000</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">inepntxfstaddr
RW 0xA000</td></tr>
</table>

dieptxf4 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 4.	RW	0xA000

dieptxf5

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00114
usb1	0xFFB40000	0xFFB40114

Offset: 0x114

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0xC000															

dieptxf5 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 5.	RW	0xC000

dieptxf6

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00118
usb1	0xFFB40000	0xFFB40118

Offset: 0x118

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0xE000															

dieptxf6 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 6.	RW	0xE000

dieptxf7

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0011C
usb1	0xFFB40000	0xFFB4011C

Offset: `0x11C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x0															

dieptxf7 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 7.	RW	0x0

dieptxf8

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00120
usb1	0xFFB40000	0xFFB40120

Offset: 0x120

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">inepntxfdep
RW 0x2000</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">inepntxfstaddr
RW 0x2000</td></tr>
</table>

dieptxf8 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 8.	RW	0x2000

dieptxf9

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00124
usb1	0xFFB40000	0xFFB40124

Offset: 0x124

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x4000															

dieptxf9 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 9.	RW	0x4000

dieptxf10

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00128
usb1	0xFFB40000	0xFFB40128

Offset: 0x128

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x6000															

dieptxf10 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 10.	RW	0x6000

dieptxf11

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0012C
usb1	0xFFB40000	0xFFB4012C

Offset: 0x12C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x8000															

dieptxf11 Fields

Bit	Name	Description	Access	Reset
29:16	`inepntxfdep`	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	`inepntxfstaddr`	This field contains the memory start address for IN endpoint Transmit FIFO 11.	RW	0x8000

dieptxf12

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00130
usb1	0xFFB40000	0xFFB40130

Offset: 0x130

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0xA000															

dieptxf12 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 12.	RW	0xA000

dieptxf13

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00134
usb1	0xFFB40000	0xFFB40134

Offset: 0x134

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">inepntxfdep
RW 0x2000</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">inepntxfstaddr
RW 0xC000</td></tr>
</table>

dieptxf13 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 13.	RW	0xC000

dieptxf14

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00138
usb1	0xFFB40000	0xFFB40138

Offset: 0x138

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="14">inepntxfdep
RW 0x2000</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">inepntxfstaddr
RW 0xE000</td></tr>
</table>

dieptxf14 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 14.	RW	0xE000

dieptxf15

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for each instantiated IN endpoint FIFO. For IN endpoint FIFO 0 use GNPTXFSIZ register for programming the size and memory start address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0013C
usb1	0xFFB40000	0xFFB4013C

Offset: 0x13C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		inepntxfdep RW 0x2000													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepntxfstaddr RW 0x0															

dieptxf15 Fields

Bit	Name	Description	Access	Reset
29:16	inepntxfdep	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 8192.	RW	0x2000
15:0	inepntxfstaddr	This field contains the memory start address for IN endpoint Transmit FIFO 15.	RW	0x0

Host Mode Registers Register Descriptions

These registers must be programmed every time the USB OTG Controller changes to Host mode.

Offset: 0x400

hcfg on page 18-117
Host Mode control. This register must be programmed every time the core changes to Host mode

hfir on page 18-120
This register stores the frame interval information for the current speed to which the otg core has enumerated

hfnum on page 18-121
This register contains the free space information for the Periodic TxFIFO and the Periodic Transmit Request Queue

hptxsts on page 18-122
This register contains the free space information for the Periodic TxFIFO and the Periodic Transmit Request Queue.

haint on page 18-125
When a significant event occurs on a channel, the Host All Channels Interrupt register interrupts the application using the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt). There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Host Channel-n Interrupt register.

haintmsk on page 18-126
The Host All Channel Interrupt Mask register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

hflbaddr on page 18-126
This Register is valid only for Host mode Scatter-Gather DMA. Starting address of the Frame list. This register is used only for Isochronous and Interrupt Channels.

hprt on page 18-127
This register is available only in Host mode. Currently, the OTG Host supports only one port. A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port.The R_SS_WC bits in this register can trigger an interrupt to the application through the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. for the R_SS_WC bits, the application must write a 1 to the bit to clear the interrupt

hcchar0 on page 18-131
Channel_number: 0.

hcsplt0 on page 18-134
Channel_number 0

hcint0 on page 18-136
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk0 on page 18-140
This register reflects the mask for each channel status described in the previous section.

hctsiz0 on page 18-141
Buffer DMA Mode

hcdma0 on page 18-143
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab0 on page 18-144
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar1 on page 18-145
Host Channel 1 Characteristics Register

hcsplt1 on page 18-148
Channel_number 1

hcint1 on page 18-150
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk1 on page 18-154
This register reflects the mask for each channel status described in the previous section.

hctsiz1 on page 18-155
Buffer DMA Mode

hcdma1 on page 18-157
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab1 on page 18-158
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar2 on page 18-159
Host Channel 2 Characteristics Register

hcsplt2 on page 18-162
Channel_number 2

hcint2 on page 18-164
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk2 on page 18-168
This register reflects the mask for each channel status described in the previous section.

hctsiz2 on page 18-169
Buffer DMA Mode.

hcdma2 on page 18-171
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab2 on page 18-172
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar3 on page 18-173
Channel_number: 3.

hcsplt3 on page 18-176
Channel_number 3

hcint3 on page 18-178
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk3 on page 18-182
This register reflects the mask for each channel status described in the previous section.

hctsiz3 on page 18-183
Buffer DMA Mode

hcdma3 on page 18-185
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab3 on page 18-186
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar4 on page 18-187
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcsplt4 on page 18-187
Channel_number 4

hcint4 on page 18-189
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk4 on page 18-193
This register reflects the mask for Channel 4 interrupt status bits.

hctsiz4 on page 18-195
Buffer DMA Mode Channel 4

hcdma4 on page 18-196
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab4 on page 18-197
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar5 on page 18-198
Channel_number: 5.

hcsplt5 on page 18-202
Channel_number 5

hcint5 on page 18-204
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk5 on page 18-208
This register reflects the mask for each channel status described in the previous section.

hctsiz5 on page 18-209
Buffer DMA Mode

hcdma5 on page 18-211
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab5 on page 18-212
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar6 on page 18-213
Host Channel 6 Characteristics Register

hcsplt6 on page 18-216
Channel_number 6

hcint6 on page 18-218
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk6 on page 18-222
This register reflects the mask for each channel status described in the previous section.

hctsiz6 on page 18-223
Buffer DMA Mode

hcdma6 on page 18-225
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab6 on page 18-226
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar7 on page 18-227
Host Channel 7 Characteristics Register

hcsplt7 on page 18-230
Channel_number 7

hcint7 on page 18-232
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk7 on page 18-236
This register reflects the mask for each channel status described in the previous section.

hctsiz7 on page 18-237
Buffer DMA Mode

hcdma7 on page 18-239
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab7 on page 18-240
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar8 on page 18-241
Host Channel 8 Characteristics Register

hcsplt8 on page 18-244
Channel_number 8

hcint8 on page 18-246
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk8 on page 18-250
This register reflects the mask for each channel status described in the previous section.

hctsiz8 on page 18-251
Buffer DMA Mode

hcdma8 on page 18-253
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab8 on page 18-254
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar9 on page 18-255
Host Channel 9 Characteristics Register

hcsplt9 on page 18-258
Channel_number 9

hcint9 on page 18-260
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk9 on page 18-264
This register reflects the mask for each channel status described in the previous section.

hctsiz9 on page 18-265
Buffer DMA Mode

hcdma9 on page 18-267
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab9 on page 18-268
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar10 on page 18-269
Host Channel 1 Characteristics Register

hcsplt10 on page 18-272
Channel_number 1

hcint10 on page 18-274
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk10 on page 18-278
This register reflects the mask for each channel status described in the previous section.

hctsiz10 on page 18-279
Buffer DMA Mode

hcdma10 on page 18-281
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab10 on page 18-282
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar11 on page 18-283
Host Channel 11 Characteristics Register

HCSPLT11 on page 18-286
Channel number 11.

hcint11 on page 18-288
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk11 on page 18-292
This register reflects the mask for each channel status described in the previous section.

hctsiz11 on page 18-293
Buffer DMA Mode

hcdma11 on page 18-295
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab11 on page 18-296
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar12 on page 18-297
Host Channel 1 Characteristics Register

hcsplt12 on page 18-300
Channel_number 1

hcint12 on page 18-302
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk12 on page 18-306
This register reflects the mask for each channel status described in the previous section.

hctsiz12 on page 18-307
Buffer DMA Mode

hcdma12 on page 18-309
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab12 on page 18-310
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar13 on page 18-311
Host Channel 13 Characteristics Register

hcsplt13 on page 18-314
Channel_number 13.

hcint13 on page 18-316
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk13 on page 18-320
This register reflects the mask for each channel status described in the previous section.

hctsiz13 on page 18-321
Buffer DMA Mode

hcdma13 on page 18-323
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab13 on page 18-324
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar14 on page 18-325
Host Channel 1 Characteristics Register

hcsplt14 on page 18-328
Channel_number 14

hcint14 on page 18-330
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk14 on page 18-334
This register reflects the mask for each channel status described in the previous section.

hctsiz14 on page 18-335

hcdma14 on page 18-337
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab14 on page 18-338
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcchar15 on page 18-339
Host Channel 15 Characteristics Register

hcsplt15 on page 18-342
Channel_number 15.

hcint15 on page 18-344
This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

hcintmsk15 on page 18-348
This register reflects the mask for each channel status described in the previous section.

hctsiz15 on page 18-349

hcdma15 on page 18-351
This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

hcdmab15 on page 18-352
These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

hcfg

Host Mode control. This register must be programmed every time the core changes to Host mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00400
usb1	0xFFB40000	0xFFB40400

Offset: 0x400

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
modechtimen RW 0x0	Reserved				perschedena RW 0x0	frlisten RW 0x0		descdma RW 0x0	Reserved						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
resvalid RW 0x2								ena32khzs RW 0x0	Reserved				fslssupp RW 0x0	fslspclksel RW 0x0	

hcfg Fields

Bit	Name	Description	Access	Reset
31	modechtimen	This bit is used to enable or disable the host core to wait for 200 PHY clock cycles at the end of Resume to change the opmode signal to the PHY to 00 after Suspend or LPM. **Value** **Description** 0x0 The Host core waits for either 200 PHY clock cycles or a linestate of SE0 at the end of resume to change the opmode from 0x2 to 0x0 0x1 The Host core waits only for a linestate of SE0 at the end of resume to change the opmode from 0x2 to 0x0	RW	0x0
26	perschedena	Applicable in Scatter/Gather DMA mode only. Enables periodic scheduling within the core. Initially, the bit is reset. The core will not process any periodic channels. As soon as this bit is set, the core will get ready to start scheduling periodic channels. In non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 Disables periodic scheduling within the core 0x1 Enables periodic scheduling within the core	RW	0x0
25:24	frlisten	The value in the register specifies the number of entries in the Frame list. This field is valid only in Scatter/Gather DMA mode. **Value** **Description** 0x0 Reserved 0x1 8 Entries 0x2 16 Entries 0x3 32 Entries	RW	0x0

Bit	Name	Description	Access	Reset
23	descdma	The application can set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset. The following combinations are available for programming: GAHBCFG.DMAEn=0,HCFG.DescDMA=0 => Slave mode GAHBCFG.DMAEn=0,HCFG.DescDMA=1 => InvalidGAHBCFG.DMAEn=1,HCFG.DescDMA=0 => Buffered DMA mode GAHBCFG.DMAEn=1,HCFG.DescDMA=1 => Scatter/Gather DMA mode **Value** **Description** 0x0 No Scatter/Gather DMA 0x1 Scatter/Gather DMA selected	RW	0x0
15:8	resvalid	This field is effective only when HCFG.Ena32KHzS is set. It will control the resume period when the core resumes from suspend. The core counts for ResValid number of clock cycles to detect a valid resume when this is set.	RW	0x2
7	ena32khzs	This bit can only be set if the USB 1.1 Full-Speed Serial Transceiver Interface has been selected. If USB 1.1 Full-Speed Serial Transceiver Interface has not been selected, this bit must be zero. When the USB 1.1 Full-Speed Serial Transceiver Interface is chosen and this bit is set, the core expects the 48-MHz PHY clock to be switched to 32 KHz during a suspend. **Value** **Description** 0x0 USB 1.1 Full-Speed Not Selected 0x1 USB 1.1 Full-Speed Serial Transceiver Interface selected	RW	0x0
2	fslssupp	The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as a FS host, even If the connected device supports HS traffic. Do not make changes to this field after initial programming. **Value** **Description** 0x0 HS/FS/LS, based on the maximum speed supported by the connected device 0x1 FS/LS-only, even if the connected device can support HS	RW	0x0

Bit	Name	Description	Access	Reset
1:0	fslspclksel	When the core is in FS Host mode. The internal PHY clock is running at 30/60 MHZ for ULPI PHY Interfaces. The internal PHY clock is running at 48MHZ for 1.1 FS transceiver Interface When the core is in LS Host mode, the internal PHY clock is running at 30/60 MHZ for ULPI PHY Interfaces. The internal PHY clock is running at 6 MHZ and the external clock is running at 48MHZ. When you select a 6 MHz clock during LS Mode, you must do a soft reset for 1.1 FS transceiver Interface. * When Core in FS mode, the internal and external clocks have the same frequency. * When Core in LS mode, - If fslspclksel is 30/60 Mhz internal and external clocks have the same frequency. - If fslspclksel is 6Mhz the internal clock is divided by eight of external 48 MHz clock (utmifs_clk). **Value** — **Description** 0x0 — PHY clock is running at 30/60 MHz 0x1 — PHY clock is running at 48 MHz 0x2 — PHY clock is running at 6 MHz	RW	0x0

hfir

This register stores the frame interval information for the current speed to which the otg core has enumerated

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00404
usb1	0xFFB40000	0xFFB40404

Offset: 0x404

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															hfirrldctrl RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
frint RW 0xEA60															

hfir Fields

Bit	Name	Description	Access	Reset
16	`hfirrldctrl`	This bit allows dynamic reloading of the HFIR register during run time. 0x0 : The HFIR cannot be reloaded dynamically0x1: the HFIR can be dynamically reloaded during runtime. This bit needs to be programmed during initial configuration and its value should not be changed during runtime. **Value** — **Description** 0x0 — The HFIR cannot be reloaded dynamically 0x1 — The HFIR can be dynamically reloaded during runtime	`RW`	`0x0`
15:0	`frint`	The value that the application programs to this field specifies the interval between two consecutive SOFs (FS) or micro- SOFs (HS) or Keep-Alive tokens (HS). This field contains the number of PHY clocks that constitute the required frame interval. The Default value Set in this field for a FS operation when the PHY clock frequency is 60 MHz. The application can write a value to this register only after the Port Enable bit of the Host Port Control and Status register (HPRT.PrtEnaPort) has been Set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY Clock Select field of the Host Configuration register (HCFG.FSLSPclkSel). Do not change the value of this field after the initial configuration. 125 s * (PHY clock frequency for HS) 1 ms * (PHY clock frequency for FS/LS)	`RW`	`0xEA60`

hfnum

This register contains the free space information for the Periodic TxFIFO and the Periodic Transmit Request Queue

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00408`
`usb1`	`0xFFB40000`	`0xFFB40408`

Offset: `0x408`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
frrem RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
frnum RO 0x3FFF															

hfnum Fields

Bit	Name	Description	Access	Reset
31:16	frrem	Indicates the amount of time remaining in the current microframe (HS) or Frame (FS/LS), in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame Interval register and a new SOF is transmitted on the USB.	RO	0x0
15:0	frnum	This field increments when a new SOF is transmitted on the USB, and is reset to 0 when it reaches 0x3FFF. Reads Return the Frame number value. **Value** **Description** 0x0 No SOF is transmitted 0x1 SOF is transmitted	RO	0x3FFF

hptxsts

This register contains the free space information for the Periodic TxFIFO and the Periodic Transmit Request Queue.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00410
usb1	0xFFB40000	0xFFB40410

Offset: 0x410

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
oddevnmframe RO 0x0	chanendpt RO 0x0				type RO 0x0		term RO 0x0	ptxqspcavail RO 0x10							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ptxfspcavail RO 0x2000															

hptxsts Fields

Bit	Name	Description	Access	Reset
31	oddevnmframe	This indicates the odd/even micro frame that is currently being processes by the MAC. **Value** **Description** 0x0 Send in even (micro)Frame 0x1 Send in odd (micro)Frame	RO	0x0

Bit	Name	Description	Access	Reset
30:27	`chanendpt`	This indicates the channel endpoint number that is currently being processes by the MAC. **Value** **Description** 0x0 End point 1 0x1 End point 2 0x2 End point 3 0x3 End point 4 0x4 End point 5 0x5 End point 6 0x6 End point 7 0x7 End point 8 0x8 End point 9 0x9 End point 10 0xa End point 11 0xb End point 12 0xc End point 13 0xd End point 14 0xe End point 15 0xf End point 16	RO	0x0
26:25	`type`	This indicates the Entry in the Periodic Tx Request Queue that is currently being processes by the MAC. **Value** **Description** 0x0 IN/OUT type 0x1 Zero-length packet type 0x2 CSPLIT type 0x3 Disable channel command	RO	0x0
24	`term`	Terminate last entry for selected channel/endpoint. **Value** **Description** 0x0 No termination 0x1 Terminate last entry for selected channel/ endpoint	RO	0x0

Bit	Name	Description	Access	Reset
23:16	`ptxqspcavail`	Indicates the number of free locations available to be written in the Periodic Transmit Request Queue. This queue holds both IN and OUT requests. Others: Reserved **Value** — **Description** 0x0 — Periodic Transmit Request Queue is full 0x1 — 1 location available 0x2 — 2 location available 0x3 — 3 location available 0x4 — 4 location available 0x5 — 5 location available 0x6 — 6 location available 0x7 — 7 location available 0x8 — 8 location available	RO	0x10
15:0	`ptxfspcavail`	Indicates the number of free locations available to be written to in the Periodic TxFIFO. Values are in terms of 32-bit words 16h0: Periodic TxFIFO is full 16h1: 1 word available 16h2: 2 words available 16hn: n words available where n is 0 to 8192	RO	0x2000

haint

When a significant event occurs on a channel, the Host All Channels Interrupt register interrupts the application using the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt). There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Host Channel-n Interrupt register.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00414`
`usb1`	`0xFFB40000`	`0xFFB40414`

Offset: `0x414`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
haint RO 0x0															

haint Fields

Bit	Name	Description	Access	Reset
15:0	haint	One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15	RO	0x0

haintmsk

The Host All Channel Interrupt Mask register works with the Host All Channel Interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00418
usb1	0xFFB40000	0xFFB40418

Offset: 0x418

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
haintmsk RW 0x0															

haintmsk Fields

Bit	Name	Description	Access	Reset
15:0	haintmsk	One bit per channel: Bit 0 for channel 0, bit 15 for channel 15 **Value** **Description** 0x0 Mask interrupt 0x1 Unmask interrupt	RW	0x0

hflbaddr

This Register is valid only for Host mode Scatter-Gather DMA. Starting address of the Frame list. This register is used only for Isochronous and Interrupt Channels.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0041C
usb1	0xFFB40000	0xFFB4041C

Offset: 0x41C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hflbaddr RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hflbaddr RW 0x0															

hflbaddr Fields

Bit	Name	Description	Access	Reset
31:0	hflbaddr	This Register is valid only for Host mode Scatter-Gather DMA mode. Starting address of the Frame list. This register is used only for Isochronous and Interrupt Channels.	RW	0x0

hprt

This register is available only in Host mode. Currently, the OTG Host supports only one port. A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port.The R_SS_WC bits in this register can trigger an interrupt to the application through the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. for the R_SS_WC bits, the application must write a 1 to the bit to clear the interrupt

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00440
usb1	0xFFB40000	0xFFB40440

Offset: 0x440

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													prtspd RO 0x0		prttstctl RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
prttstctl RW 0x0			prtpwr RW 0x0	prtlnsts RO 0x0		Reserved	prtrst RW 0x0	prtsusp RO 0x0	prtres RW 0x0	prtovrcurrchng RO 0x0	prtovrcurract RO 0x0	prtenchng RO 0x0	prtena RO 0x0	PrtConnDet RO 0x0	prtconnsts RO 0x0

hprt Fields

Bit	Name	Description	Access	Reset
18:17	`prtspd`	Indicates the speed of the device attached to this port. **Value** **Description** 0x0 High speed 0x1 Full speed 0x2 Low speed 0x3 Reserved	RO	0x0
16:13	`prttstctl`	The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port. **Value** **Description** 0x0 Test mode disabled 0x1 Test_J mode 0x2 Test_K mode 0x3 Test_SE0_NAK mode 0x4 Test_Packet mode 0x5 Test_force_Enable	RW	0x0
12	`prtpwr`	The application uses this field to control power to this port, and the core can clear this bit on an over current condition. **Value** **Description** 0x0 Power off 0x1 Power on	RW	0x0
11:10	`prtlnsts`	Indicates the current logic level USB data lines. Bit [10]: Logic level of D+ Bit [11]: Logic level of D- **Value** **Description** 0x1 Logic level of D+ 0x2 Logic level of D-	RO	0x0

Bit	Name	Description	Access	Reset
8	prtrst	When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete. The application must leave this bit Set for at least a minimum duration mentioned below to start a reset on the port. The application can leave it Set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by theUSB standard. This bit is cleared by the core even if there is no device connected to the Host. High speed: 50 ms Full speed/Low speed: 10 ms **Value** — **Description** 0x0 — Port not in reset 0x1 — Port in reset	RW	0x0
7	prtsusp	The application sets this bit to put this port in Suspend mode. The core only stops sending SOFs when this is Set. To stop the PHY clock, the application must Set the Port Clock Stop bit, which asserts the suspend input pin of the PHY. The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the Port Reset bit or Port Resume bit in this register or the Resume/Remote Wakeup Detected Interrupt bit or Disconnect Detected Interrupt bit in the Core Interrupt register (GINTSTS.WkUpInt or GINTSTS.DisconnInt, respectively). This bit is cleared by the core even if there is no device connected to the Host. **Value** — **Description** 0x0 — Port not in Suspend mode 0x1 — Port in Suspend mode	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>6</td><td>prtres</td><td>The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit. If the core detects a USB remote wakeup sequence, as indicated by the Port Resume/Remote Wakeup Detected Interrupt bit of the Core Interrupt register (GINTSTS.WkUpInt), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently drivingresume signaling. When LPM is enabled and the core is in the L1 (Sleep) state, setting this bit results in the following behavior: The core continues to drive the resume signal until a pre-determined time specified in the GLPMCFG.HIRD_Thres[3:0] field. If the core detects a USB remote wakeup sequence, as indicated by the Port L1 Resume/Remote L1 Wakeup Detected Interrupt bit of the Core Interrupt register (GINTSTS.L1WkUpInt), the core starts driving resume signaling without application intervention and clears this bit at the end of the resume. The read value of this bit indicates whether the core is currently driving resume signaling.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No resume driven</td></tr>
<tr><td>0x1</td><td>Resume driven</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>5</td><td>prtovrcurrchng</td><td>The core sets this bit when the status of the PortOver-current Active bit (bit 4) in this register changes.This bit can be set only by the core and the application should write 1 to clear it
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Status of port overcurrent no change</td></tr>
<tr><td>0x1</td><td>Status of port overcurrent changed</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>4</td><td>prtovrcurract</td><td>Indicates the overcurrent condition of the port. 0x0: No overcurrent condition 0x1: Overcurrent condition
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No overcurrent condition</td></tr>
<tr><td>0x1</td><td>Overcurrent condition</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
3	prtenchng	The core sets this bit when the status of the Port Enable bit [2] of this register changes. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 Port Enable bit 2 no change 0x1 Port Enable bit 2 changed	RO	0x0
2	prtena	A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot Set this bit by a register write. It can only clear it to disable the port by writing 1. This bit does not trigger any interrupt to the application. **Value** **Description** 0x0 Port disabled 0x1 Port enabled	RO	0x0
1	PrtConnDet	The core sets this bit when a device connection is detected to trigger an interrupt to the application using the Host Port Interrupt bit of the Core Interrupt register (GINTSTS.PrtInt). This bit can be set only by the core and the application should write 1 to clear it.The application must write a 1 to this bit to clear the interrupt. **Value** **Description** 0x0 Device connection detected 0x1 No device connection detected	RO	0x0
0	prtconnsts	Defines whether port is attached. **Value** **Description** 0x0 No device is attached to the port 0x1 A device is attached to the port	RO	0x0

hcchar0

Channel_number: 0.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00500

Module Instance	Base Address	Register Address
usb1	0xFFB40000	0xFFB40500

Offset: 0x500

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar0 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled. This field is set by the application and cleared by the OTG host. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 No activity 0x1 Stop transmitting/receiving data	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1'b1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be Set to at least 2'b01. **Value** / **Description** 0x0 — Reserved This field yields undefined results 0x1 — 1 transaction 0x2 — 2 transactions to be issued for this endpoint per microframe 0x3 — 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** / **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`
17	`lspddev`	This field is Set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software. **Value** / **Description** 0x0 — Communicating with non lowspeed 0x1 — Communicating with lowspeed	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	`epdir`	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT 0x1 IN	`RW`	`0x0`
14:11	`epnum`	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	`RW`	`0x0`
10:0	`mps`	Indicates the maximum packet size of the associated endpoint.	`RW`	`0x0`

hcsplt0

Channel_number 0

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00504`
`usb1`	`0xFFB40000`	`0xFFB40504`

Offset: `0x504`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt0 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint0

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00508
usb1	0xFFB40000	0xFFB40508

Offset: 0x508

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint0 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>12</td><td>xcs_xact_err</td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Excessive Transaction Error</td></tr>
<tr><td>0x1</td><td>Excessive Transaction Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>bnaintr</td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No BNA Interrupt</td></tr>
<tr><td>0x1</td><td>BNA Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>10</td><td>datatglerr</td><td>This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Data Toggle Error</td></tr>
<tr><td>0x1</td><td>Data Toggle Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>9</td><td>frmovrun</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Frame Overrun</td></tr>
<tr><td>0x1</td><td>Frame Overrun</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk0

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0050C
usb1	0xFFB40000	0xFFB4050C

Offset: `0x50C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk0 Fields

Bit	Name	Description	Access	Reset
13	`frm_lst_rollintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz0

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00510
usb1	0xFFB40000	0xFFB40510

Offset: `0x510`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz0 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** — **Description** 0x0 — No ping protocol 0x1 — Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 0x2 — DATA1 0x3 — MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	For an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma0

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00514`
`usb1`	`0xFFB40000`	`0xFFB40514`

Offset: `0x514`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma0 RW 0x0															

hcdma0 Fields

Bit	Name	Description	Access	Reset
31:0	`hcdma0`	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab0

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00518`
`usb1`	`0xFFB40000`	`0xFFB40518`

Offset: `0x518`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab0 RW 0x0															

hcdmab0 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab0	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar1

Host Channel 1 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00520
usb1	0xFFB40000	0xFFB40520

Offset: 0x520

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>chena
RO 0x0</td><td>chdis
RO 0x0</td><td>Reserved</td><td colspan="7">devaddr
RW 0x0</td><td colspan="2">ec
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>lspddev
RW 0x0</td><td>Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>epdir
RW 0x0</td><td colspan="4">epnum
RW 0x0</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

hcchar1 Fields

Bit	Name	Description	Access	Reset
31	`chena`	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	`chdis`	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	`devaddr`	This field selects the specific device serving as the data source or sink.	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>21:20</td><td>ec</td><td>When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Reserved This field yields undefined result</td></tr>
<tr><td>0x1</td><td>1 transaction</td></tr>
<tr><td>0x2</td><td>2 transactions to be issued for this endpoint per microframe</td></tr>
<tr><td>0x3</td><td>3 transactions to be issued for this endpoint per microframe</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>19:18</td><td>eptype</td><td>Indicates the transfer type selected.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Control</td></tr>
<tr><td>0x1</td><td>Isochronous</td></tr>
<tr><td>0x2</td><td>Bulk</td></tr>
<tr><td>0x3</td><td>Interrupt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>17</td><td>lspddev</td><td>This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Not Communicating with low speed device</td></tr>
<tr><td>0x1</td><td>Communicating with low speed device</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`epdir`	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	`RW`	`0x0`
14:11	`epnum`	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	`RW`	`0x0`
10:0	`mps`	Indicates the maximum packet size of the associated endpoint.	`RW`	`0x0`

hcsplt1

Channel_number 1

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00524`
`usb1`	`0xFFB40000`	`0xFFB40524`

Offset: `0x524`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt1 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint1

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00528
usb1	0xFFB40000	0xFFB40528

Offset: 0x528

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint1 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	`bblerr`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	`xacterr`	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	`nyet`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	`ack`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	`nak`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	`stall`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	`ahberr`	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	`chhltd`	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk1

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0052C
usb1	0xFFB40000	0xFFB4052C

Offset: 0x52C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>frm_lst_rollintrmsk
RW
0x0</td><td>Reserved</td><td>bnaintrmsk
RW
0x0</td><td colspan="8">Reserved</td><td>ahberrmsk
RW
0x0</td><td>chhltdmsk
RW
0x0</td><td>xfercomplmsk
RW 0x0</td></tr>
</table>

hcintmsk1 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`

hctsiz1

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00530`
`usb1`	`0xFFB40000`	`0xFFB40530`

Offset: `0x530`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz1 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** — **Description** 0x0 — No ping protocol 0x1 — Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 0x2 — DATA1 0x3 — MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma1

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00534
usb1	0xFFB40000	0xFFB40534

Offset: 0x534

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma1 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma1 RW 0x0															

hcdma1 Fields

Bit	Name	Description	Access	Reset
31:0	`hcdma1`	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab1

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00538
usb1	0xFFB40000	0xFFB40538

Offset: `0x538`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab1 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab1 RW 0x0															

hcdmab1 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab1	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar2

Host Channel 2 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00540
usb1	0xFFB40000	0xFFB40540

Offset: 0x540

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar2 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>chena</td><td>When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Indicates that the descriptor structure is not yet ready</td></tr><tr><td>0x1</td><td>Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>30</td><td>chdis</td><td>The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Transmit/Recieve normal</td></tr><tr><td>0x1</td><td>Stop transmitting/receiving</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>28:22</td><td>devaddr</td><td>This field selects the specific device serving as the data source or sink.</td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt2

Channel_number 2

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00544
usb1	0xFFB40000	0xFFB40544

Offset: 0x544

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>spltena
RW 0x0</td><td colspan="14">Reserved</td><td>compsplt
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">xactpos
RW 0x0</td><td colspan="7">hubaddr
RW 0x0</td><td colspan="7">prtaddr
RW 0x0</td></tr>
</table>

hcsplt2 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transac-tiontranslator.	RW	0x0

hcint2

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00548
usb1	0xFFB40000	0xFFB40548

Offset: 0x548

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint2 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>12</td><td><code>xcs_xact_err</code></td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Excessive Transaction Error</td></tr>
<tr><td>0x1</td><td>Excessive Transaction Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td><code>bnaintr</code></td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No BNA Interrupt</td></tr>
<tr><td>0x1</td><td>BNA Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>10</td><td><code>datatglerr</code></td><td>This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Data Toggle Error</td></tr>
<tr><td>0x1</td><td>Data Toggle Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>9</td><td><code>frmovrun</code></td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Frame Overrun</td></tr>
<tr><td>0x1</td><td>Frame Overrun</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's address register to get the error address. **Value** / **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** / **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk2

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0054C
usb1	0xFFB40000	0xFFB4054C

Offset: 0x54C

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk2 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz2

Buffer DMA Mode.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00550
usb1	0xFFB40000	0xFFB40550

Offset: 0x550

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz2 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma2

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00554
usb1	0xFFB40000	0xFFB40554

Offset: 0x554

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma2 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma2 RW 0x0															

hcdma2 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma2	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab2

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00558
usb1	0xFFB40000	0xFFB40558

Offset: 0x558

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab2 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab2 RW 0x0															

hcdmab2 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab2	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar3

Channel_number: 3.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00560
usb1	0xFFB40000	0xFFB40560

Offset: 0x560

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar3 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled. This field is set by the application and cleared by the OTG host. **Value** / **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0

Bit	Name	Description	Access	Reset
30	`chdis`	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** / **Description** 0x0 No activity 0x1 Stop transmitting/receiving data	RO	0x0
28:22	`devaddr`	This field selects the specific device serving as the data source or sink.	RW	0x0
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be Set to at least 1. **Value** / **Description** 0x0 Reserved This field yields undefined results 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	RW	0x0
19:18	`eptype`	Indicates the transfer type selected. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	`lspddev`	This field is Set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software. **Value** / **Description** 0x0 Communicating with non lowspeed 0x1 Communicating with lowspeed	`RW`	`0x0`
15	`epdir`	Indicates whether the transaction is IN or OUT. **Value** / **Description** 0x0 OUT 0x1 IN	`RW`	`0x0`

Bit	Name	Description	Access	Reset
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt3

Channel_number 3

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00564
usb1	0xFFB40000	0xFFB40564

Offset: 0x564

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt3 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint3

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00568
usb1	0xFFB40000	0xFFB40568

Offset: 0x568

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint3 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Excessive Transaction Error 0x1 — Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No BNA Interrupt 0x1 — BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** — **Description** 0x0 — No Data Toggle Error 0x1 — Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Frame Overrun 0x1 — Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>4</td><td>nak</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No NAK Response Received Interrupt</td></tr><tr><td>0x1</td><td>NAK Response Received Interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>stall</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Stall Interrupt</td></tr><tr><td>0x1</td><td>Stall Interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>ahberr</td><td>This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No AHB error</td></tr><tr><td>0x1</td><td>AHB error during AHB read/write</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>1</td><td>chhltd</td><td>In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Channel not halted</td></tr><tr><td>0x1</td><td>Channel Halted</td></tr></table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk3

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0056C
usb1	0xFFB40000	0xFFB4056C

Offset: 0x56C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk3 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz3

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00570
usb1	0xFFB40000	0xFFB40570

Offset: 0x570

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz3 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma3

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00574
usb1	0xFFB40000	0xFFB40574

Offset: 0x574

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma3 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma3 RW 0x0															

hcdma3 Fields

Bit	Name	Description	Access	Reset
31:0	`hcdma3`	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab3

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00578`
`usb1`	`0xFFB40000`	`0xFFB40578`

Offset: `0x578`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdmab3
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdmab3
RW 0x0</td></tr>
</table>

hcdmab3 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab3	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar4

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00580
usb1	0xFFB40000	0xFFB40580

Offset: 0x580

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab4 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab4 RW 0x0															

hcchar4 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab4	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcsplt4

Channel_number 4

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00584
usb1	0xFFB40000	0xFFB40584

Offset: 0x584

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt4 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0

Bit	Name	Description	Access	Reset
15:14	`xactpos`	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	`hubaddr`	This field holds the device address of the transaction translator's hub.	RW	0x0
6:0	`prtaddr`	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint4

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00588
usb1	0xFFB40000	0xFFB40588

Offset: 0x588

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint4 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0
12	xcs_xact_err	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0

Bit	Name	Description	Access	Reset
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0
8	`bblerr`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0

Bit	Name	Description	Access	Reset
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
3	`stall`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	`RO`	`0x0`
2	`ahberr`	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	`RO`	`0x0`
1	`chhltd`	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	`RO`	`0x0`
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	`RO`	`0x0`

hcintmsk4

This register reflects the mask for Channel 4 interrupt status bits.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0058C
usb1	0xFFB40000	0xFFB4058C

Offset: 0x58C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk4 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** — **Description** 0x0 — Mask 0x1 — No mask	RW	0x0
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** — **Description** 0x0 — Mask 0x1 — No mask	RW	0x0

Bit	Name	Description	Access	Reset
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz4

Buffer DMA Mode Channel 4

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00590
usb1	0xFFB40000	0xFFB40590

Offset: 0x590

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>dopng
RW 0x0</td><td colspan="2">pid
RW 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

hctsiz4 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma4

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00594

Module Instance	Base Address	Register Address
usb1	0xFFB40000	0xFFB40594

Offset: 0x594

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma4 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma4 RW 0x0															

hcdma4 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma4	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab4

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00598
usb1	0xFFB40000	0xFFB40598

Offset: 0x598

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab4 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab4 RW 0x0															

hcdmab4 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab4	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar5

Channel_number: 5.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005A0
usb1	0xFFB40000	0xFFB405A0

Offset: 0x5A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar5 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled. This field is set by the application and cleared by the OTG host. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 No activity 0x1 Stop transmitting/receiving data	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	ec	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. 0x0: Reserved This field yields undefined results. 0x1: transaction 0x2: 2 transactions to be issued for this endpoint permicroframe 0x3: 3 transactions to be issued for this endpoint permicroframeWhen HCSPLTn.SpltEna is Set (1), this field indicates thenumber of immediate retries to be performed for a periodic splittransactions on transaction errors. This field must be Set to atleast 1. **Value** **Description** 0x0 Reserved This field yields undefined results 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	RW	0x0
19:18	eptype	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	lspddev	This field is Set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software. **Value** — **Description** 0x0 — Communicating with non lowspeed 0x1 — Communicating with lowspeed	RW	0x0
15	epdir	Indicates whether the transaction is IN or OUT. **Value** — **Description** 0x0 — OUT 0x1 — IN	RW	0x0

Bit	Name	Description	Access	Reset
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** — **Description** 0x0 — End point 0 0x1 — End point 1 0x2 — End point 2 0x3 — End point 3 0x4 — End point 4 0x5 — End point 5 0x6 — End point 6 0x7 — End point 7 0x8 — End point 8 0x9 — End point 9 0xa — End point 10 0xb — End point 11 0xc — End point 12 0xd — End point 13 0xe — End point 14 0xf — End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt5

Channel_number 5

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005A4
usb1	0xFFB40000	0xFFB405A4

Offset: 0x5A4

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>spltena
RW 0x0</td><td colspan="14">Reserved</td><td>compsplt
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">xactpos
RW 0x0</td><td colspan="7">hubaddr
RW 0x0</td><td colspan="7">prtaddr
RW 0x0</td></tr>
</table>

hcsplt5 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint5

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005A8
usb1	0xFFB40000	0xFFB405A8

Offset: 0x5A8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint5 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	xcs_xact_err	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	datatglerr	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	frmovrun	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>8</td><td>bblerr</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Babble Error</td></tr>
<tr><td>0x1</td><td>Babble Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>7</td><td>xacterr</td><td>Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Transaction Error</td></tr>
<tr><td>0x1</td><td>Transaction Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>6</td><td>nyet</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No NYET Response Received Interrupt</td></tr>
<tr><td>0x1</td><td>NYET Response Received Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>5</td><td>ack</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No ACK Response Received Transmitted Interrupt</td></tr>
<tr><td>0x1</td><td>ACK Response Received Transmitted Interrup</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk5

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005AC
usb1	0xFFB40000	0xFFB405AC

Offset: 0x5AC

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk5 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz5

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005B0
usb1	0xFFB40000	0xFFB405B0

Offset: 0x5B0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz5 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma5

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005B4
usb1	0xFFB40000	0xFFB405B4

Offset: 0x5B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma5 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma5 RW 0x0															

hcdma5 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma5	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab5

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005B8
usb1	0xFFB40000	0xFFB405B8

Offset: 0x5B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab5 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab5 RW 0x0															

hcdmab5 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab5	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar6

Host Channel 6 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005C0
usb1	0xFFB40000	0xFFB405C0

Offset: 0x5C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar6 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** / **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** / **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** — **Description** 0x0 — Reserved This field yields undefined result 0x1 — 1 transaction 0x2 — 2 transactions to be issued for this endpoint per microframe 0x3 — 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** — **Description** 0x0 — Not Communicating with low speed device 0x1 — Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	`epdir`	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	`RW`	`0x0`
14:11	`epnum`	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	`RW`	`0x0`
10:0	`mps`	Indicates the maximum packet size of the associated endpoint.	`RW`	`0x0`

hcsplt6

Channel_number 6

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB005C4`
`usb1`	`0xFFB40000`	`0xFFB405C4`

Offset: `0x5C4`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt6 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint6

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005C8
usb1	0xFFB40000	0xFFB405C8

Offset: 0x5C8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint6 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Excessive Transaction Error 0x1 — Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No BNA Interrupt 0x1 — BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** — **Description** 0x0 — No Data Toggle Error 0x1 — Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Frame Overrun 0x1 — Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	`bblerr`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	`xacterr`	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	`nyet`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	`ack`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk6

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005CC
usb1	0xFFB40000	0xFFB405CC

Offset: 0x5CC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk6 Fields

Bit	Name	Description	Access	Reset
13	`frm_lst_rollintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** — **Description** 0x0 — Mask 0x1 — No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** — **Description** 0x0 — Mask 0x1 — No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** — **Description** 0x0 — Mask 0x1 — No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** — **Description** 0x0 — Mask 0x1 — No mask	`RW`	`0x0`

hctsiz6

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB005D0`
`usb1`	`0xFFB40000`	`0xFFB405D0`

Offset: `0x5D0`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>dopng
RW 0x0</td><td colspan="2">pid
RW 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

hctsiz6 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma6

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB005D4`
`usb1`	`0xFFB40000`	`0xFFB405D4`

Offset: `0x5D4`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdma6
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdma6
RW 0x0</td></tr>
</table>

hcdma6 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma6	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab6

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005D8
usb1	0xFFB40000	0xFFB405D8

Offset: 0x5D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab6 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab6 RW 0x0															

hcdmab6 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab6	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar7

Host Channel 7 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005E0
usb1	0xFFB40000	0xFFB405E0

Offset: 0x5E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar7 Fields

Bit	Name	Description	Access	Reset
31	`chena`	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	`chdis`	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	`devaddr`	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** — **Description** 0x0 — OUT Direction 0x1 — IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** — **Description** 0x0 — End point 0 0x1 — End point 1 0x2 — End point 2 0x3 — End point 3 0x4 — End point 4 0x5 — End point 5 0x6 — End point 6 0x7 — End point 7 0x8 — End point 8 0x9 — End point 9 0xa — End point 10 0xb — End point 11 0xc — End point 12 0xd — End point 13 0xe — End point 14 0xf — End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt7

Channel_number 7

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005E4
usb1	0xFFB40000	0xFFB405E4

Offset: 0x5E4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt7 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint7

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005E8
usb1	0xFFB40000	0xFFB405E8

Offset: 0x5E8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint7 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** / **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	`bblerr`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	`xacterr`	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	`nyet`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	`ack`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk7

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005EC
usb1	0xFFB40000	0xFFB405EC

Offset: 0x5EC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk7 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz7

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005F0
usb1	0xFFB40000	0xFFB405F0

Offset: `0x5F0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz7 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma7

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB005F4`
`usb1`	`0xFFB40000`	`0xFFB405F4`

Offset: `0x5F4`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdma7
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdma7
RW 0x0</td></tr>
</table>

hcdma7 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma7	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab7

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB005F8
usb1	0xFFB40000	0xFFB405F8

Offset: 0x5F8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab7 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab7 RW 0x0															

hcdmab7 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab7	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar8

Host Channel 8 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00600
usb1	0xFFB40000	0xFFB40600

Offset: 0x600

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar8 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>chena</td><td>When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Indicates that the descriptor structure is not yet ready</td></tr>
<tr><td>0x1</td><td>Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>30</td><td>chdis</td><td>The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Transmit/Recieve normal</td></tr>
<tr><td>0x1</td><td>Stop transmitting/receiving</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>28:22</td><td>devaddr</td><td>This field selects the specific device serving as the data source or sink.</td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
21:20	ec	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	RW	0x0
19:18	eptype	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0
17	lspddev	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	RW	0x0

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt8

Channel_number 8

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00604
usb1	0xFFB40000	0xFFB40604

Offset: 0x604

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>spltena
RW 0x0</td><td colspan="14">Reserved</td><td>compsplt
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">xactpos
RW 0x0</td><td colspan="7">hubaddr
RW 0x0</td><td colspan="7">prtaddr
RW 0x0</td></tr>
</table>

hcsplt8 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint8

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00608
usb1	0xFFB40000	0xFFB40608

Offset: 0x608

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint8 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	`RO`	`0x0`
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	`RO`	`0x0`
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	`RO`	`0x0`
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	`RO`	`0x0`

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	`nak`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	`RO`	`0x0`
3	`stall`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	`RO`	`0x0`
2	`ahberr`	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	`RO`	`0x0`
1	`chhltd`	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	`RO`	`0x0`

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk8

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0060C
usb1	0xFFB40000	0xFFB4060C

Offset: 0x60C

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk8 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`

hctsiz8

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00610`
`usb1`	`0xFFB40000`	`0xFFB40610`

Offset: `0x610`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz8 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma8

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00614`
`usb1`	`0xFFB40000`	`0xFFB40614`

Offset: `0x614`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma8 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma8 RW 0x0															

hcdma8 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma8	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab8

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00618
usb1	0xFFB40000	0xFFB40618

Offset: 0x618

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdmab8
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdmab8
RW 0x0</td></tr>
</table>

hcdmab8 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab8	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar9

Host Channel 9 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00620
usb1	0xFFB40000	0xFFB40620

Offset: 0x620

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar9 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	ec	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	RW	0x0
19:18	eptype	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0
17	lspddev	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	RW	0x0

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt9

Channel_number 9

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00624
usb1	0xFFB40000	0xFFB40624

Offset: 0x624

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt9 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint9

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00628
usb1	0xFFB40000	0xFFB40628

Offset: 0x628

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint9 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	`nak`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	`RO`	`0x0`
3	`stall`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	`RO`	`0x0`
2	`ahberr`	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	`RO`	`0x0`
1	`chhltd`	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	`RO`	`0x0`

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk9

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0062C
usb1	0xFFB40000	0xFFB4062C

Offset: 0x62C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk9 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz9

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00630
usb1	0xFFB40000	0xFFB40630

Offset: 0x630

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz9 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma9

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00634
usb1	0xFFB40000	0xFFB40634

Offset: 0x634

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma9 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma9 RW 0x0															

hcdma9 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma9	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab9

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00638
usb1	0xFFB40000	0xFFB40638

Offset: 0x638

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab9 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab9 RW 0x0															

hcdmab9 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab9	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar10

Host Channel 1 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00640
usb1	0xFFB40000	0xFFB40640

Offset: 0x640

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar10 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** — **Description** 0x0 — OUT Direction 0x1 — IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** — **Description** 0x0 — End point 0 0x1 — End point 1 0x2 — End point 2 0x3 — End point 3 0x4 — End point 4 0x5 — End point 5 0x6 — End point 6 0x7 — End point 7 0x8 — End point 8 0x9 — End point 9 0xa — End point 10 0xb — End point 11 0xc — End point 12 0xd — End point 13 0xe — End point 14 0xf — End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt10

Channel_number 1

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00644
usb1	0xFFB40000	0xFFB40644

Offset: 0x644

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt10 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** — **Description** 0x0 — Split not enabled 0x1 — Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** — **Description** 0x0 — No split transaction 0x1 — Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** — **Description** 0x0 — Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 — End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 — Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 — All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint10

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00648
usb1	0xFFB40000	0xFFB40648

Offset: 0x648

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint10 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** / **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>8</td><td>bblerr</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Babble Error</td></tr>
<tr><td>0x1</td><td>Babble Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>7</td><td>xacterr</td><td>Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Transaction Error</td></tr>
<tr><td>0x1</td><td>Transaction Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>6</td><td>nyet</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No NYET Response Received Interrupt</td></tr>
<tr><td>0x1</td><td>NYET Response Received Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>5</td><td>ack</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No ACK Response Received Transmitted Interrupt</td></tr>
<tr><td>0x1</td><td>ACK Response Received Transmitted Interrup</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No NAK Response Received Interrupt 0x1 — NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Stall Interrupt 0x1 — Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** — **Description** 0x0 — No AHB error 0x1 — AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** — **Description** 0x0 — Channel not halted 0x1 — Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk10

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0064C
usb1	0xFFB40000	0xFFB4064C

Offset: `0x64C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk10 Fields

Bit	Name	Description	Access	Reset
13	`frm_lst_rollintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`

hctsiz10

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00650`
`usb1`	`0xFFB40000`	`0xFFB40650`

Offset: `0x650`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz10 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma10

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00654
usb1	0xFFB40000	0xFFB40654

Offset: 0x654

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdma10
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdma10
RW 0x0</td></tr>
</table>

hcdma10 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma10	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab10

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00658
usb1	0xFFB40000	0xFFB40658

Offset: 0x658

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab10 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab10 RW 0x0															

hcdmab10 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab10	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar11

Host Channel 11 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00660
usb1	0xFFB40000	0xFFB40660

Offset: 0x660

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar11 Fields

Bit	Name	Description	Access	Reset
31	`chena`	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	`chdis`	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	`devaddr`	This field selects the specific device serving as the data source or sink.	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>21:20</td><td>ec</td><td>When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Reserved This field yields undefined result</td></tr>
<tr><td>0x1</td><td>1 transaction</td></tr>
<tr><td>0x2</td><td>2 transactions to be issued for this endpoint per microframe</td></tr>
<tr><td>0x3</td><td>3 transactions to be issued for this endpoint per microframe</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>19:18</td><td>eptype</td><td>Indicates the transfer type selected.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Control</td></tr>
<tr><td>0x1</td><td>Isochronous</td></tr>
<tr><td>0x2</td><td>Bulk</td></tr>
<tr><td>0x3</td><td>Interrupt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>17</td><td>lspddev</td><td>This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Not Communicating with low speed device</td></tr>
<tr><td>0x1</td><td>Communicating with low speed device</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`epdir`	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	`RW`	`0x0`
14:11	`epnum`	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	`RW`	`0x0`
10:0	`mps`	Indicates the maximum packet size of the associated endpoint.	`RW`	`0x0`

HCSPLT11

Channel number 11.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00664`
`usb1`	`0xFFB40000`	`0xFFB40664`

Offset: `0x664`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>spltena
RW 0x0</td><td colspan="14">Reserved</td><td>compsplt
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">xactpos
RW 0x0</td><td colspan="7">hubaddr
RW 0x0</td><td colspan="7">prtaddr
RW 0x0</td></tr>
</table>

HCSPLT11 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transac-tiontranslator.	RW	0x0

hcint11

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00668
usb1	0xFFB40000	0xFFB40668

Offset: 0x668

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint11 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Excessive Transaction Error 0x1 — Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No BNA Interrupt 0x1 — BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** — **Description** 0x0 — No Data Toggle Error 0x1 — Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Frame Overrun 0x1 — Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Babble Error 0x1 — Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Transaction Error 0x1 — Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No NYET Response Received Interrupt 0x1 — NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No ACK Response Received Transmitted Interrupt 0x1 — ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk11

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0066C
usb1	0xFFB40000	0xFFB4066C

Offset: `0x66C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk11 Fields

Bit	Name	Description	Access	Reset
13	`frm_lst_rollintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz11

Buffer DMA Mode

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00670
usb1	0xFFB40000	0xFFB40670

Offset: 0x670

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz11 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma11

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00674`
`usb1`	`0xFFB40000`	`0xFFB40674`

Offset: `0x674`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdma11
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdma11
RW 0x0</td></tr>
</table>

hcdma11 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma11	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab11

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00678
usb1	0xFFB40000	0xFFB40678

Offset: 0x678

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab11 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab11 RW 0x0															

hcdmab11 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab11	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar12

Host Channel 1 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00680
usb1	0xFFB40000	0xFFB40680

Offset: 0x680

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar12 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** — **Description** 0x0 — Reserved This field yields undefined result 0x1 — 1 transaction 0x2 — 2 transactions to be issued for this endpoint per microframe 0x3 — 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** — **Description** 0x0 — Not Communicating with low speed device 0x1 — Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** — **Description** 0x0 — OUT Direction 0x1 — IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** — **Description** 0x0 — End point 0 0x1 — End point 1 0x2 — End point 2 0x3 — End point 3 0x4 — End point 4 0x5 — End point 5 0x6 — End point 6 0x7 — End point 7 0x8 — End point 8 0x9 — End point 9 0xa — End point 10 0xb — End point 11 0xc — End point 12 0xd — End point 13 0xe — End point 14 0xf — End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt12

Channel_number 1

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00684
usb1	0xFFB40000	0xFFB40684

Offset: 0x684

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt12 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transac-tiontranslator.	RW	0x0

hcint12

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00688
usb1	0xFFB40000	0xFFB40688

Offset: 0x688

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint12 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** / **Description** 0x0 No Descriptor rollover interrupt 0x1 Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	`xfercompl`	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk12

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0068C
usb1	0xFFB40000	0xFFB4068C

Offset: `0x68C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk12 Fields

Bit	Name	Description	Access	Reset
13	`frm_lst_rollintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`

hctsiz12

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00690`
`usb1`	`0xFFB40000`	`0xFFB40690`

Offset: `0x690`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz12 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma12

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00694
usb1	0xFFB40000	0xFFB40694

Offset: 0x694

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma12 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma12 RW 0x0															

hcdma12 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma12	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab12

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00698
usb1	0xFFB40000	0xFFB40698

Offset: 0x698

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab12 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab12 RW 0x0															

hcdmab12 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab12	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar13

Host Channel 13 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006A0
usb1	0xFFB40000	0xFFB406A0

Offset: 0x6A0

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
chena RO 0x0	chdis RO 0x0	Reserved	devaddr RW 0x0							ec RW 0x0		eptype RW 0x0		lspddev RW 0x0	Reserved
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
epdir RW 0x0	epnum RW 0x0				mps RW 0x0										

hcchar13 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** **Description** 0x0 Reserved This field yields undefined result 0x1 1 transaction 0x2 2 transactions to be issued for this endpoint per microframe 0x3 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** **Description** 0x0 Not Communicating with low speed device 0x1 Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** / **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** / **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt13

Channel_number 13.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006A4
usb1	0xFFB40000	0xFFB406A4

Offset: 0x6A4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt13 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint13

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006A8
usb1	0xFFB40000	0xFFB406A8

Offset: 0x6A8

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint13 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No Excessive Transaction Error 0x1 Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** **Description** 0x0 No BNA Interrupt 0x1 BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** **Description** 0x0 No Data Toggle Error 0x1 Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Frame Overrun 0x1 Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Babble Error 0x1 — Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Transaction Error 0x1 — Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No NYET Response Received Interrupt 0x1 — NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No ACK Response Received Transmitted Interrupt 0x1 — ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk13

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006AC
usb1	0xFFB40000	0xFFB406AC

Offset: 0x6AC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk13 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	`RW`	`0x0`

hctsiz13

Buffer DMA Mode

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB006B0`
`usb1`	`0xFFB40000`	`0xFFB406B0`

Offset: `0x6B0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz13 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** — **Description** 0x0 — No ping protocol 0x1 — Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 0x2 — DATA1 0x3 — MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma13

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB006B4`
`usb1`	`0xFFB40000`	`0xFFB406B4`

Offset: `0x6B4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma13 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma13 RW 0x0															

hcdma13 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma13	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab13

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006B8
usb1	0xFFB40000	0xFFB406B8

Offset: 0x6B8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab13 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab13 RW 0x0															

hcdmab13 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab13	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar14

Host Channel 1 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006C0
usb1	0xFFB40000	0xFFB406C0

Offset: 0x6C0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>chena
RO 0x0</td><td>chdis
RO 0x0</td><td>Reserved</td><td colspan="7">devaddr
RW 0x0</td><td colspan="2">ec
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>lspddev
RW 0x0</td><td>Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>epdir
RW 0x0</td><td colspan="4">epnum
RW 0x0</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

hcchar14 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/ Gather mode is enabled. **Value** — **Description** 0x0 — Indicates that the descriptor structure is not yet ready 0x1 — Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/ receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** — **Description** 0x0 — Transmit/Recieve normal 0x1 — Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	ec	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** — **Description** 0x0 — Reserved This field yields undefined result 0x1 — 1 transaction 0x2 — 2 transactions to be issued for this endpoint per microframe 0x3 — 3 transactions to be issued for this endpoint per microframe	RW	0x0
19:18	eptype	Indicates the transfer type selected. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0
17	lspddev	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 0x3 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** — **Description** 0x0 — Not Communicating with low speed device 0x1 — Communicating with low speed device	RW	0x0

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt14

Channel_number 14

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006C4
usb1	0xFFB40000	0xFFB406C4

Offset: 0x6C4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt14 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transac-tiontranslator.	RW	0x0

hcint14

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006C8
usb1	0xFFB40000	0xFFB406C8

Offset: 0x6C8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint14 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

Bit	Name	Description	Access	Reset
12	`xcs_xact_err`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Excessive Transaction Error 0x1 — Excessive Transaction Error	RO	0x0
11	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No BNA Interrupt 0x1 — BNA Interrupt	RO	0x0
10	`datatglerr`	This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. **Value** — **Description** 0x0 — No Data Toggle Error 0x1 — Data Toggle Error	RO	0x0
9	`frmovrun`	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No Frame Overrun 0x1 — Frame Overrun	RO	0x0

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>4</td><td>nak</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No NAK Response Received Interrupt</td></tr>
<tr><td>0x1</td><td>NAK Response Received Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>stall</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Stall Interrupt</td></tr>
<tr><td>0x1</td><td>Stall Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>ahberr</td><td>This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No AHB error</td></tr>
<tr><td>0x1</td><td>AHB error during AHB read/write</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>1</td><td>chhltd</td><td>In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Channel not halted</td></tr>
<tr><td>0x1</td><td>Channel Halted</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** / **Description** 0x0 No transfer 0x1 Transfer completed normally without any errors	RO	0x0

hcintmsk14

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006CC
usb1	0xFFB40000	0xFFB406CC

Offset: 0x6CC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk14 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	`bnaintrmsk`	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	`ahberrmsk`	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	`chhltdmsk`	Channel Halted. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	`xfercomplmsk`	Transfer complete. **Value** **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz14

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006D0
usb1	0xFFB40000	0xFFB406D0

Offset: `0x6D0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz14 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic). The width of this counter is specified as 19 bits.	`RW`	`0x0`

hcdma14

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB006D4`
`usb1`	`0xFFB40000`	`0xFFB406D4`

Offset: `0x6D4`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma14 RW 0x0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma14 RW 0x0															

hcdma14 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma14	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab14

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006D8
usb1	0xFFB40000	0xFFB406D8

Offset: 0x6D8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdmab14 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdmab14 RW 0x0															

hcdmab14 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab14	These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

hcchar15

Host Channel 15 Characteristics Register

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006E0
usb1	0xFFB40000	0xFFB406E0

Offset: 0x6E0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>chena
RO 0x0</td><td>chdis
RO 0x0</td><td>Reserved</td><td colspan="7">devaddr
RW 0x0</td><td colspan="2">ec
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>lspddev
RW 0x0</td><td>Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>epdir
RW 0x0</td><td colspan="4">epnum
RW 0x0</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

hcchar15 Fields

Bit	Name	Description	Access	Reset
31	chena	When Scatter/Gather mode is disabled This field is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled When Scatter/Gather mode is enabled. **Value** **Description** 0x0 Indicates that the descriptor structure is not yet ready 0x1 Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor	RO	0x0
30	chdis	The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel Disabled interrupt before treating the channel as disabled. **Value** **Description** 0x0 Transmit/Recieve normal 0x1 Stop transmitting/receiving	RO	0x0
28:22	devaddr	This field selects the specific device serving as the data source or sink.	RW	0x0

Bit	Name	Description	Access	Reset
21:20	`ec`	When the Split Enable bit of the Host Channel-n Split Control register (HCSPLTn.SpltEna) is reset (0), this field indicates to the host the number of transactions that must be executed per microframe for this periodic endpoint. for non periodic transfers, this field is used only in DMA mode, and specifies the number packets to be fetched for this channel before the internal DMA engine changes arbitration. When HCSPLTn.SpltEna is Set (1), this field indicates the number of immediate retries to be performed for a periodic split transactions on transaction errors. This field must be set to at least 1. **Value** — **Description** 0x0 — Reserved This field yields undefined result 0x1 — 1 transaction 0x2 — 2 transactions to be issued for this endpoint per microframe 0x3 — 3 transactions to be issued for this endpoint per microframe	`RW`	`0x0`
19:18	`eptype`	Indicates the transfer type selected. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`
17	`lspddev`	This field is set by the application to indicate that this channel is communicating to a low-speed device. The application must program this bit when a low speed device is connected to the host through an FS HUB. The HS OTG Host core uses this field to drive the XCVR_SELECT signal to 2'b11 while communicating to the LS Device through the FS hub. In a peer to peer setup, the HS OTG Host core ignores this bit even if it is set by the application software **Value** — **Description** 0x0 — Not Communicating with low speed device 0x1 — Communicating with low speed device	`RW`	`0x0`

Bit	Name	Description	Access	Reset
15	epdir	Indicates whether the transaction is IN or OUT. **Value** / **Description** 0x0 OUT Direction 0x1 IN Direction	RW	0x0
14:11	epnum	Indicates the endpoint number on the device serving as the data source or sink. **Value** / **Description** 0x0 End point 0 0x1 End point 1 0x2 End point 2 0x3 End point 3 0x4 End point 4 0x5 End point 5 0x6 End point 6 0x7 End point 7 0x8 End point 8 0x9 End point 9 0xa End point 10 0xb End point 11 0xc End point 12 0xd End point 13 0xe End point 14 0xf End point 15	RW	0x0
10:0	mps	Indicates the maximum packet size of the associated endpoint.	RW	0x0

hcsplt15

Channel_number 15.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006E4
usb1	0xFFB40000	0xFFB406E4

Offset: 0x6E4

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spltena RW 0x0	Reserved														compsplt RW 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
xactpos RW 0x0		hubaddr RW 0x0							prtaddr RW 0x0						

hcsplt15 Fields

Bit	Name	Description	Access	Reset
31	spltena	The application sets this field to indicate that this channel is enabled to perform split transactions. **Value** **Description** 0x0 Split not enabled 0x1 Split enabled	RW	0x0
16	compsplt	The application sets this field to request the OTG host to perform a complete split transaction. **Value** **Description** 0x0 No split transaction 0x1 Split transaction	RW	0x0
15:14	xactpos	This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction. **Value** **Description** 0x0 Mid. This is the middle payload of this transaction (which is larger than 188 bytes) 0x1 End. This is the last payload of this transaction (which is larger than 188 bytes) 0x2 Begin. This is the first data payload of this transaction (which is larger than 188 bytes) 0x3 All. This is the entire data payload is of this transaction (which is less than or equal to 188 bytes)	RW	0x0
13:7	hubaddr	This field holds the device address of the transaction translator's hub.	RW	0x0

Bit	Name	Description	Access	Reset
6:0	prtaddr	This field is the port number of the recipient transactiontranslator.	RW	0x0

hcint15

This register indicates the status of a channel with respect to USB- and AHB-related events. The application must read this register when the Host Channels Interrupt bit of the Core Interrupt register (GINTSTS.HChInt) is set. Before the application can read this register, it must first read the Host All Channels Interrupt (HAINT) register to get the exact channel number for the Host Channel-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006E8
usb1	0xFFB40000	0xFFB406E8

Offset: 0x6E8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		desc_lst_rollintr RO 0x0	xcs_xact_err RO 0x0	bnaintr RO 0x0	datatglerr RO 0x0	frmovrun RO 0x0	bblerr RO 0x0	xacterr RO 0x0	nyet RO 0x0	ack RO 0x0	nak RO 0x0	stall RO 0x0	ahberr RO 0x0	chhltd RO 0x0	xfercompl RO 0x0

hcint15 Fields

Bit	Name	Description	Access	Reset
13	desc_lst_rollintr	Descriptor rollover interrupt (DESC_LST_ROLLIntr) This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel's descriptor list rolls over. for non Scatter/Gather DMA mode, this bit is reserved. **Value** — **Description** 0x0 — No Descriptor rollover interrupt 0x1 — Descriptor rollover interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>12</td><td>xcs_xact_err</td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR will not be generated for Isochronous channels.for non Scatter/Gather DMA mode, this bit is reserved.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Excessive Transaction Error</td></tr><tr><td>0x1</td><td>Excessive Transaction Error</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>bnaintr</td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process. BNA will not be generated for Isochronous channels. for non Scatter/Gather DMA mode, this bit is reserved.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No BNA Interrupt</td></tr><tr><td>0x1</td><td>BNA Interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>10</td><td>datatglerr</td><td>This bit can be set only by the core and the application should write 1 to clear it. In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Data Toggle Error</td></tr><tr><td>0x1</td><td>Data Toggle Error</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>9</td><td>frmovrun</td><td>In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Frame Overrun</td></tr><tr><td>0x1</td><td>Frame Overrun</td></tr></table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
8	bblerr	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core..This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Babble Error 0x1 Babble Error	RO	0x0
7	xacterr	Indicates one of the following errors occurred on the USB.-CRC check failure -Timeout -Bit stuff error -False EOP In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Transaction Error 0x1 Transaction Error	RO	0x0
6	nyet	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NYET Response Received Interrupt 0x1 NYET Response Received Interrupt	RO	0x0
5	ack	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No ACK Response Received Transmitted Interrupt 0x1 ACK Response Received Transmitted Interrup	RO	0x0

Bit	Name	Description	Access	Reset
4	nak	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core.This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No NAK Response Received Interrupt 0x1 NAK Response Received Interrupt	RO	0x0
3	stall	In Scatter/Gather DMA mode, the interrupt due to this bit is masked in the core. This bit can be set only by the core and the application should write 1 to clear it. **Value** **Description** 0x0 No Stall Interrupt 0x1 Stall Interrupt	RO	0x0
2	ahberr	This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address. **Value** **Description** 0x0 No AHB error 0x1 AHB error during AHB read/write	RO	0x0
1	chhltd	In non Scatter/Gather DMA mode, it indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application or because of a completed transfer. In Scatter/gather DMA mode, this indicates that transfer completed due to any of the following . EOL being set in descriptor . AHB error . Excessive transaction errors . Babble . Stall **Value** **Description** 0x0 Channel not halted 0x1 Channel Halted	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Transfer completed normally without any errors. This bit can be set only by the core and the application should write 1 to clear it. **Value** — **Description** 0x0 — No transfer 0x1 — Transfer completed normally without any errors	RO	0x0

hcintmsk15

This register reflects the mask for each channel status described in the previous section.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006EC
usb1	0xFFB40000	0xFFB406EC

Offset: 0x6EC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		frm_lst_rollintrmsk RW 0x0	Reserved	bnaintrmsk RW 0x0	Reserved								ahberrmsk RW 0x0	chhltdmsk RW 0x0	xfercomplmsk RW 0x0

hcintmsk15 Fields

Bit	Name	Description	Access	Reset
13	frm_lst_rollintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** — **Description** 0x0 — Mask 0x1 — No mask	RW	0x0

Bit	Name	Description	Access	Reset
11	bnaintrmsk	This bit is valid only when Scatter/Gather DMA mode is enabled. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
2	ahberrmsk	In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in HCINTn. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
1	chhltdmsk	Channel Halted. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0
0	xfercomplmsk	Transfer complete. **Value** / **Description** 0x0 Mask 0x1 No mask	RW	0x0

hctsiz15

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006F0
usb1	0xFFB40000	0xFFB406F0

Offset: 0x6F0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
dopng RW 0x0	pid RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

hctsiz15 Fields

Bit	Name	Description	Access	Reset
31	dopng	This bit is used only for OUT transfers.Setting this field to 1 directs the host to do PING protocol. Do not Set this bit for IN transfers. If this bit is set for IN transfers it disables the channel. **Value** **Description** 0x0 No ping protocol 0x1 Ping protocol	RW	0x0
30:29	pid	The application programs this field with the type of PID to use forthe initial transaction. The host maintains this field for the rest of the transfer. **Value** **Description** 0x0 DATA0 0x1 DATA2 0x2 DATA1 0x3 MDATA (non-control)/SETUP (control)	RW	0x0
28:19	pktcnt	This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN). The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion. The width of this counter is specified as 10 bits.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	xfersize	for an OUT, this field is the number of data bytes the host sends during the transfer. for an IN, this field is the buffer size that the application has Reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).The width of this counter is specified as 19 bits.	RW	0x0

hcdma15

This register is used by the OTG host in the internal DMA mode to maintain the current buffer pointer for IN/OUT transactions. The starting DMA address must be DWORD-aligned.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006F4
usb1	0xFFB40000	0xFFB406F4

Offset: 0x6F4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
hcdma15 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
hcdma15 RW 0x0															

hcdma15 Fields

Bit	Name	Description	Access	Reset
31:0	hcdma15	Non-Isochronous: This field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value. This field holds the address of the 2*(nTD+1) bytes of locations in which the isochronous descriptors are present where N is based on nTD as per Table below [31:N] Base Address [N-1:3] Offset [2:0] 000 HS ISOC FS ISOC nTD N nTD N 7 6 1 4 15 7 3 5 31 8 7 6 63 9 15 7 127 10 31 8 255 11 63 9 [N-1:3] (Isoc):[8:3] (Non Isoc): Current Transfer Desc(CTD): Non Isochronous: This value is in terms of number of descriptors. The values can be from 0 to 63. 0 - 1 descriptor. 63 - 64 descriptors. This field indicates the current descriptor processed in the list. This field is updated both by application and the core. for example, if the application enables the channel after programming CTD=5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of (8bytes*5=) 40(decimal) to DMAAddr. Isochronous: CTD for isochronous is based on the current frame/microframe value. Need to be set to zero by application.	RW	0x0

hcdmab15

These registers are present only in case of Scatter/Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB006F8
usb1	0xFFB40000	0xFFB406F8

Offset: 0x6F8

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">hcdmab15
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">hcdmab15
RW 0x0</td></tr>
</table>

hcdmab15 Fields

Bit	Name	Description	Access	Reset
31:0	hcdmab15	These registers are present only in case of Scatter/ Gather DMA. These registers are implemented in RAM instead of flop-based implementation. Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode. Otherwise this field is reserved.	RW	0x0

Device Mode Registers Register Descriptions

These registers must be programmed every time the USB OTG Controller changes to Device mode.

Offset: 0x800

dcfg on page 18-369
This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

dctl on page 18-372

dsts on page 18-377
This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (DAINT) register.

diepmsk on page 18-379
This register works with each of the Device IN Endpoint Interrupt (DIEPINTn) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the DIEPINTn register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

doepmsk on page 18-381
This register works with each of the Device OUT Endpoint Interrupt (DOEPINTn) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the DOEPINTn register can be masked by writing into the corresponding bit in this register. Status bits are masked by default

daint on page 18-383
When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively). This is shown in Figure 5-2. There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. for a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (DIEPINTn/DOEPINTn).

daintmsk on page 18-389
The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (DAINT) register bit corresponding to that interrupt is still set.

dvbusdis on page 18-394
This register specifies the VBUS discharge time after VBUS pulsing during SRP.

dvbuspulse on page 18-395
This register specifies the VBUS pulsing time during SRP.

dthrctl on page 18-395
Thresholding is not supported in Slave mode and so this register must not be programmed in Slave mode. for threshold support, the AHB must be run at 60 MHz or higher.

diepempmsk on page 18-398
This register is used to control the IN endpoint FIFO empty interrupt generation (DIEPINTn.TxfEmp).

diepctl0 on page 18-401
This register covers Device Control IN Endpoint 0.

diepint0 on page 18-404
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz0 on page 18-408
The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/ DOEPCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 1 to 15. When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.

diepdma0 on page 18-409
DMA Addressing.

dtxfsts0 on page 18-410
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab0 on page 18-410
Endpoint 16.

diepctl1 on page 18-411
Endpoint_number: 1

diepint1 on page 18-417
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz1 on page 18-421
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma1 on page 18-422
DMA Addressing.

dtxfsts1 on page 18-423
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab1 on page 18-424
DMA Buffer Address.

diepctl2 on page 18-424
Endpoint_number: 2

diepint2 on page 18-430
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz2 on page 18-434
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma2 on page 18-435
DMA Addressing.

DTXFSTS2 on page 18-436
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab2 on page 18-437
DMA Buffer Address.

diepctl3 on page 18-437
Endpoint_number: 3

diepint3 on page 18-443
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz3 on page 18-447
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma3 on page 18-448
DMA Addressing.

dtxfsts3 on page 18-449
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab3 on page 18-450
DMA Buffer Address.

diepctl4 on page 18-450
Endpoint_number: 4

diepint4 on page 18-456
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz4 on page 18-460
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma4 on page 18-461
DMA Addressing.

dtxfsts4 on page 18-462
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab4 on page 18-463
DMA Buffer Address.

diepctl5 on page 18-463
Endpoint_number: 5

diepint5 on page 18-469
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz5 on page 18-473
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma5 on page 18-474
DMA Addressing.

dtxfsts5 on page 18-475
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab5 on page 18-476
Device IN Endpoint 1 Buffer Address.

diepctl6 on page 18-476
Endpoint_number: 6

diepint6 on page 18-482
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz6 on page 18-486
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma6 on page 18-487
DMA Addressing.

dtxfsts6 on page 18-488
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab6 on page 18-489
DMA Buffer Address.

diepctl7 on page 18-489
Endpoint_number: 7

diepint7 on page 18-495
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz7 on page 18-499
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma7 on page 18-500
DMA Addressing.

dtxfsts7 on page 18-501
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab7 on page 18-502
DMA Buffer Address.

diepctl8 on page 18-502
Endpoint_number: 8

diepint8 on page 18-508
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz8 on page 18-512
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma8 on page 18-513
DMA Addressing.

dtxfsts8 on page 18-514
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab8 on page 18-515
DMA Buffer Address.

diepctl9 on page 18-515
Endpoint_number: 9

diepint9 on page 18-521
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz9 on page 18-525
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma9 on page 18-526
DMA Addressing.

dtxfsts9 on page 18-527
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab9 on page 18-528
DMA Buffer Address.

diepctl10 on page 18-528
Endpoint_number: 10

diepint10 on page 18-534
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz10 on page 18-538
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma10 on page 18-539
DMA Addressing.

dtxfsts10 on page 18-540
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab10 on page 18-541
DMA Buffer Address.

diepctl11 on page 18-541
Endpoint_number: 11

diepint11 on page 18-547
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz11 on page 18-551
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma11 on page 18-552
DMA Addressing.

dtxfsts11 on page 18-553
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab11 on page 18-554
DMA Buffer Address.

diepctl12 on page 18-554
Endpoint_number: 12

diepint12 on page 18-560
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz12 on page 18-564
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma12 on page 18-565
DMA Addressing.

dtxfsts12 on page 18-566
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab12 on page 18-567
DMA Buffer Address.

diepctl13 on page 18-567
Endpoint_number: 13

diepint13 on page 18-573
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz13 on page 18-577
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma13 on page 18-578
DMA Addressing.

dtxfsts13 on page 18-579
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab13 on page 18-580
DMA Buffer Address.

diepctl14 on page 18-580
Endpoint_number: 14

diepint14 on page 18-586
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz14 on page 18-590
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma14 on page 18-591
DMA Addressing.

dtxfsts14 on page 18-592
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab14 on page 18-593
DMA Buffer Address.

diepctl15 on page 18-593
Endpoint_number: 15

diepint15 on page 18-599
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

dieptsiz15 on page 18-603
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

diepdma15 on page 18-604
DMA Addressing.

dtxfsts15 on page 18-605
This register contains the free space information for the Device IN endpoint TxFIFO.

diepdmab15 on page 18-606
DMA Buffer Address.

doepctl0 on page 18-606
This is Control OUT Endpoint 0 Control register.

doepint0 on page 18-609
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz0 on page 18-613
The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/ DOEPCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 1 to 15. When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.

doepdma0 on page 18-614
DMA Addressing.

doepdmab0 on page 18-615
DMA Buffer Address.

doepctl1 on page 18-616
Out Endpoint 1.

doepint1 on page 18-621
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz1 on page 18-625
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma1 on page 18-626
DMA Addressing.

doepdmab1 on page 18-627
DMA Buffer Address.

DOEPCTL2 on page 18-628
Out Endpoint 2.

doepint2 on page 18-633
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz2 on page 18-637
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma2 on page 18-638
DMA Addressing.

doepdmab2 on page 18-639
DMA Buffer Address.

DOEPCTL3 on page 18-640
Out Endpoint 3.

doepint3 on page 18-645
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz3 on page 18-649
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma3 on page 18-650
DMA OUT Address.

doepdmab3 on page 18-651
DMA Buffer Address.

doepctl4 on page 18-652
Out Endpoint 4.

Doepint4 on page 18-657
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz4 on page 18-661
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma4 on page 18-662
DMA OUT Address.

doepdmab4 on page 18-663
DMA Buffer Address.

doepctl5 on page 18-664
Out Endpoint 5.

doepint5 on page 18-669
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz5 on page 18-673
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma5 on page 18-674
DMA OUT Address.

doepdmab5 on page 18-675
DMA Buffer Address.

doepctl6 on page 18-676
Out Endpoint 6.

doepint6 on page 18-681
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz6 on page 18-685
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma6 on page 18-686
DMA OUT Address.

doepdmab6 on page 18-687
DMA Buffer Address.

doepctl7 on page 18-688
Endpoint_number: 7

doepint7 on page 18-694
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz7 on page 18-698
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma7 on page 18-699
DMA OUT Address.

doepdmab7 on page 18-699
DMA Buffer Address.

doepctl8 on page 18-700
Out Endpoint 8.

doepint8 on page 18-705
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz8 on page 18-709
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma8 on page 18-710
DMA OUT Address.

doepdmab8 on page 18-711
DMA Buffer Address.

doepctl9 on page 18-712
Out Endpoint 9.

doepint9 on page 18-717
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz9 on page 18-721
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma9 on page 18-722
DMA OUT Address.

doepdmab9 on page 18-723
DMA Buffer Address.

doepctl10 on page 18-724
Out Endpoint 10.

doepint10 on page 18-729
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz10 on page 18-733
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma10 on page 18-734
DMA OUT Address.

doepdmab10 on page 18-735
DMA Buffer Address.

doepctl11 on page 18-736
Out Endpoint 11.

doepint11 on page 18-741
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz11 on page 18-745
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma11 on page 18-746
DMA OUT Address.

doepdmab11 on page 18-747
DMA Buffer Address.

doepctl12 on page 18-748
Out Endpoint 12.

doepint12 on page 18-753
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz12 on page 18-757
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma12 on page 18-758
DMA OUT Address.

doepdmab12 on page 18-759
DMA Buffer Address.

doepctl13 on page 18-760
Out Endpoint 13.

doepint13 on page 18-765
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz13 on page 18-769
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma13 on page 18-770
DMA OUT Address.

doepdmab13 on page 18-771
DMA Buffer Address.

doepctl14 on page 18-772
Out Endpoint 14.

doepint14 on page 18-777
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz14 on page 18-781
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma14 on page 18-782
DMA OUT Address.

doepdmab14 on page 18-783
DMA Buffer Address.

doepctl15 on page 18-784
Out Endpoint 15.

doepint15 on page 18-789
This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

doeptsiz15 on page 18-793
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

doepdma15 on page 18-794
DMA OUT Address.

doepdmab15 on page 18-795
DMA Buffer Address.

dcfg

This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00800`
`usb1`	`0xFFB40000`	`0xFFB40800`

Offset: `0x800`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="6">resvalid
RW 0x2</td><td colspan="2">perschintvl
RW 0x0</td><td>descdma
RW
0x0</td><td colspan="7">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td>endevoutnak
RW
0x0</td><td colspan="2">perfrint
RW 0x0</td><td colspan="7">devaddr
RW 0x0</td><td>ena32khzsusp
RW
0x0</td><td>nzstsouthshk
RW
0x0</td><td colspan="2">devspd
RW 0x0</td></tr>
</table>

dcfg Fields

Bit	Name	Description	Access	Reset
31:26	`resvalid`	This field is effective only when DCFG.Ena32KHzSusp is set. It will control the resume period when the core resumes from suspend. The core counts for ResValid number of clock cycles to detect a valid resume when this is set	RW	0x2

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>25:24</td><td>perschintvl</td><td>PerSchIntvl must be programmed only for Scatter/Gather DMAmode. Description: This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25,50 or 75% of (micro)frame. When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data . When no periodic endpoints are active, Then the internal DMA engine services non-periodic endpoints, ignoring this field. After the specified time within a (micro)frame, the DMA switches to fetching for non-periodic endpoints. 2'b00: 25% of (micro)frame. 2'b01: 50% of (micro)frame. 2'b10: 75% of (micro)frame. 2'b11: Reserved.Reset: 2'b00Access: read-write
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>25% of (micro)frame</td></tr>
<tr><td>0x1</td><td>50% of (micro)frame</td></tr>
<tr><td>0x2</td><td>75% of (micro)frame</td></tr>
<tr><td>0x3</td><td>Reserved</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>23</td><td>descdma</td><td>When the Scatter/Gather DMA option selected during configuration of the RTL, the application can Set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset.The following combinations are available for programming: GAHBCFG.DMAEn=0,DCFG.DescDMA=0 => Slave mode GAHBCFG.DMAEn=0,DCFG.DescDMA=1 => Invalid GAHBCFG.DMAEn=1,DCFG.DescDMA=0 => Buffered DMA mode GAHBCFG.DMAEn=1,DCFG.DescDMA=1 => Scatter/Gather DMA mode
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disable Scatter gather DMA</td></tr>
<tr><td>0x1</td><td>Enable Scatter gather DMA</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
13	`endevoutnak`	This bit enables setting NAK for Bulk OUT endpoints after the transfer is completed for Device mode Descriptor DMA It is one time programmable after reset like any other DCFG register bits. **Value** — **Description** 0x0 — The core does not set NAK after Bulk OUT transfer complete 0x1 — The core sets NAK after Bulk OUT transfer complete	RW	0x0
12:11	`perfrint`	Indicates the time within a (micro)frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine If all the isochronous traffic for that (micro)frame is complete. 0x0: 80% of the (micro) frame interval 0x1: 85% 0x2: 90% 0x3: 95% **Value** — **Description** 0x0 — 80% of the (micro)frame interval 0x1 — 85% of the (micro)frame interval 0x2 — 90% of the (micro)frame interval 0x3 — 95% of the (micro)frame interval	RW	0x0
10:4	`devaddr`	The application must program this field after every SetAddress control command.	RW	0x0
3	`ena32khzsusp`	When the USB 1.1 Full-Speed Serial Transceiver Interface is chosen and this bit is set, the core expects the 48-MHz PHY clock to be switched to 32 KHz during a suspend. This bit can only be set if USB 1.1 Full-Speed Serial Transceiver Interface has been selected. If USB 1.1 Full-Speed Serial Transceiver Interface has not been selected, this bit must be zero. **Value** — **Description** 0x0 — USB 1.1 Full-Speed Serial Transceiver not selected 0x1 — USB 1.1 Full-Speed Serial Transceiver Interface selected	RW	0x0

Bit	Name	Description	Access	Reset
2	nzstsouthshk	The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage. 1: Send a STALL handshake on a nonzero-length statusOUT transaction and do not send the received OUT packet tothe application. 0: Send the received OUT packet to the application (zerolengthor nonzero-length) and send a handshake based onthe NAK and STALL bits for the endpoint in the DeviceEndpoint Control register. **Value** / **Description** 0x0 Send the received OUT packet to the application zerolength 0x1 Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application	RW	0x0
1:0	devspd	Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. **Value** / **Description** 0x0 High speed USB 2.0 PHY clock is 30 MHz or 60 MHz 0x1 Full speed USB 2.0 PHY clock is 30 MHz or 60 MHz 0x2 Low speed USB 1.1 transceiver clock is 6 MHz 0x3 Full speed USB 1.1 transceiver clock is 48 MHz	RW	0x0

dctl

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00804
usb1	0xFFB40000	0xFFB40804

Offset: 0x804

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															nakonbble RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ignrfrmnum RW 0x0	gmc RW 0x0		Reserved	pwronprgdone RW 0x0	cgoutnak WO 0x0	sgoutnak WO 0x0	CGNPInNak WO 0x0	sgnpinnak WO 0x0	tstctl RW 0x0			goutnaksts RO 0x0	gnpinnaksts RO 0x0	sftdiscon RW 0x0	rmtwkupsig RW 0x0

dctl Fields

Bit	Name	Description	Access	Reset
16	nakonbble	Set NAK automatically on babble (NakOnBble). The core sets NAK automatically for the endpoint on which babble is received. **Value** **Description** 0x0 Disable NAK on Babble Error 0x1 NAK on Babble Error	RW	0x0

Bit	Name	Description	Access	Reset
15	ignrfrmnum	Do NOT program IgnrFrmNum bit to 1'b1 when the core is operating in threshold mode. When Scatter/ Gather DMA mode is enabled this feature is not applicable to High Speed, High bandwidth transfers. When this bit is enabled, there must be only one packet per descriptor. In Scatter/Gather DMA mode, if this bit is enabled, the packets are not flushed when a ISOC IN token is received for an elapsed frame. When Scatter/Gather DMA mode is disabled, this field is used by the application to enable periodic transfer interrupt. The application can program periodic endpoint transfers for multiple (micro) frames. 0: periodic transfer interrupt feature is disabled, application needs to program transfers for periodic endpoints every (micro)frame 1: periodic transfer interrupt feature is enabled, application can program transfers for multiple (micro)frames for periodic endpoints. In non Scatter/Gather DMA mode the application will receive transfer complete interrupt after transfers for multiple (micro)frames are completed. **Value** **Description** 0x0 The core transmits the packets only in the frame number in which they are intended to be transmitted 0x1 The core ignores the frame number, sending packets immediately as the packets are ready	RW	0x0
14:13	gmc	GMC must be programmed only once after initialization.Applicable only for Scatter/Gather DMA mode. This indicates the number of packets to be serviced for that end point before moving to the next end point. It is only for non-periodic end points. When Scatter/Gather DMA mode is disabled, this field isreserved. and reads 0. **Value** **Description** 0x0 Invalid 0x1 1 packet 0x2 2 packets 0x3 3 packets	RW	0x0

Bit	Name	Description	Access	Reset
11	pwronprgdone	The application uses this bit to indicate that register-programming is completed after a wake-up from Power Downmode. **Value** — **Description** 0x0 — Power-On Programming not done 0x1 — Power-On Programming Done	RW	0x0
10	cgoutnak	A write to this field clears the Global OUT NAK. **Value** — **Description** 0x0 — Disable Clear Global OUT NAK 0x1 — Clear Global OUT NAK	WO	0x0
9	sgoutnak	A write to this field sets the Global OUT NAK.The application uses this bit to send a NAK handshake on all OUT endpoints. The application must Set the this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register GINTSTS.GOUTNakEff) is cleared. **Value** — **Description** 0x0 — Disable Global OUT NAK 0x1 — Global OUT NAK	WO	0x0
8	CGNPInNak	A write to this field clears the Global Non-periodic IN NAK. **Value** — **Description** 0x0 — Disable Global Non-periodic IN NAK 0x1 — Clear Global Non-periodic IN NAK	WO	0x0

Bit	Name	Description	Access	Reset
7	`sgnpinnak`	A write to this field sets the Global Non-periodic IN NAK. The application uses this bit to send a NAK handshake on all nonperiodic IN endpoints. The core can also Set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must Set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (GINTSTS.GINNakEff) is cleared **Value** **Description** 0x0 Disable Global Non-periodic IN NAK 0x1 Global Non-periodic IN NAK	`WO`	`0x0`
6:4	`tstctl`	Others: Reserved. **Value** **Description** 0x0 Test mode disabled 0x1 Test_J mode 0x2 Test_K mode 0x3 Test_SE0_NAK mode 0x4 Test_Packet mode 0x5 Test_force_Enable	`RW`	`0x0`
3	`goutnaksts`	Reports NAK status. All isochronous OUT packets aredropped. **Value** **Description** 0x0 A handshake is sent based on the FIFO Status and the NAK and STALL bit settings. 0x1 No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.	`RO`	`0x0`
2	`gnpinnaksts`	Defines IN NAK conditions. **Value** **Description** 0x0 A handshake is sent out based on the data availability in the transmit FIFO 0x1 A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
1	sftdiscon	The application uses this bit to signal the otg core to do a soft disconnect. As long as this bit is Set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit. There is a minimum duration for which the core must keep this bit set. When this bit is cleared after a soft disconnect, the core drives the phy_opmode_o signal on the ULPI, which generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.; **Value** — **Description** 0x0 — Normal operation 0x1 — The core drives the phy_opmode_o signal on the ULPI	RW	0x0
0	rmtwkupsig	When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must Set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 115 ms after setting it. Remote Wakeup Signaling (RmtWkUpSig) When LPM is enabled, In L1 state the behavior of this bit is as follows: When the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Sleep state. As specified in the LPM specification, the hardware will automatically clear this bit after a time of 50us (TL1DevDrvResume) after set by application. Application should not set this bit when GLPMCFG bRemoteWake from the previous LPM transaction was zero. **Value** — **Description** 0x0 — No exit suspend state 0x1 — Exit Suspend State	RW	0x0

dsts

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (DAINT) register.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00808

Module Instance	Base Address	Register Address
usb1	0xFFB40000	0xFFB40808

Offset: 0x808

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										soffn RO 0x0					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
soffn RO 0x0								Reserved				errticerr RO 0x0	enumspd RO 0x1		suspsts RO 0x0

dsts Fields

Bit	Name	Description	Access	Reset
21:8	soffn	When the core is operating at high speed, this field contains a microframe number. When the core is operating at full or low speed, this field contains a Frame number.	RO	0x0
3	errticerr	The core sets this bit to report any erratic errors (phy_rxvalid_i/phy_rxvldh_i or phy_rxactive_i is asserted for at least 2 ms, due to PHY error) seen on the UTMI+ . Due to erratic errors, the otg core goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (GINTSTS.ErlySusp). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover. **Value** **Description** 0x0 No Erratic Error 0x1 Erratic Error	RO	0x0

Bit	Name	Description	Access	Reset
2:1	`enumspd`	Indicates the speed at which the otg core has come up after speed detection through a chirp sequence. **Value** **Description** 0x0 High speed (PHY clock is running at 30 or 60 MHz) 0x1 Full speed (PHY clock is running at 30 or 60 MHz) 0x2 Low speed (PHY clock is running at 6 MHz) 0x3 Full speed (PHY clock is running at 48 MHz)	`RO`	`0x1`
0	`suspsts`	In Device mode, this bit is Set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal for an extended period of time. The core comes out of the suspend: -When there is any activity on the phy_line_state_i signal -When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig). **Value** **Description** 0x0 No suspend state 0x1 Suspend state	`RO`	`0x0`

diepmsk

This register works with each of the Device IN Endpoint Interrupt (DIEPINTn) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the DIEPINTn register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00810`
`usb1`	`0xFFB40000`	`0xFFB40810`

Offset: `0x810`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		nakmsk RW 0x0	Reserved			bnainintrmsk RW 0x0	txfifoundrnmsk RW 0x0	Reserved	inepnakeffmsk RW 0x0	intknepmismsk RW 0x0	intkntxfempmsk RW 0x0	timeoutmsk RW 0x0	ahberrmsk RW 0x0	epdisbldmsk RW 0x0	xfercomplmsk RW 0x0

diepmsk Fields

Bit	Name	Description	Access	Reset
13	nakmsk	**Value** **Description** 0x0 Mask NAK Interrupt 0x1 No Mask NAK Interrupt	RW	0x0
9	bnainintrmsk	**Value** **Description** 0x0 Mask BNA Interrupt 0x1 No Mask BNA Interrupt	RW	0x0
8	txfifoundrnmsk	**Value** **Description** 0x0 Mask Fifo Underrun Interrupt 0x1 No Mask Fifo Underrun Interrupt	RW	0x0
6	inepnakeffmsk	**Value** **Description** 0x0 Mask IN Endpoint NAK Effective Interrupt 0x1 No Mask IN Endpoint NAK Effective Interrupt	RW	0x0
5	intknepmismsk	**Value** **Description** 0x0 Mask IN Token received with EP Mismatch Interrupt 0x1 No Mask IN Token received with EP Mismatch Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
4	intkntxfempmsk	**Value** **Description** 0x0 Mask IN Token Received When TxFIFO Empty Interrupt 0x1 No Mask IN Token Received When TxFIFO Empty Interrupt	RW	0x0
3	timeoutmsk	Non-isochronous endpoints **Value** **Description** 0x0 Mask Timeout Condition Interrupt 0x1 No Mask Timeout Condition Interrupt	RW	0x0
2	ahberrmsk	**Value** **Description** 0x0 Mask AHB Error Interrupt 0x1 No Mask AHB Error Interrupt	RW	0x0
1	epdisbldmsk	**Value** **Description** 0x0 Mask Endpoint Disabled Interrupt 0x1 No Mask Endpoint Disabled Interrupt	RW	0x0
0	xfercomplmsk	**Value** **Description** 0x0 Mask Transfer Completed Interrupt 0x1 No Mask Transfer Completed Interrupt	RW	0x0

doepmsk

This register works with each of the Device OUT Endpoint Interrupt (DOEPINTn) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the DOEPINTn register can be masked by writing into the corresponding bit in this register. Status bits are masked by default

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00814
usb1	0xFFB40000	0xFFB40814

Offset: 0x814

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetmsk RW 0x0	nakmsk RW 0x0	bbleerrmsk RW 0x0	Reserved		bnaoutintrmsk RW 0x0	outpkterrmsk RW 0x0	Reserved	back2backsetup RW 0x0	Reserved	outtknepdismsk RW 0x0	setupmsk RW 0x0	ahberrmsk RW 0x0	epdisbldmsk RW 0x0	xfercomplmsk RW 0x0

doepmsk Fields

Bit	Name	Description	Access	Reset
14	nyetmsk	**Value** **Description** 0x0 Mask NYET Interrupt 0x1 No Mask NYET Interrupt	RW	0x0
13	nakmsk	**Value** **Description** 0x0 Mask NAK Interrupt 0x1 No Mask NAK Interrupt	RW	0x0
12	bbleerrmsk	**Value** **Description** 0x0 Mask Babble Error Interrupt 0x1 No Mask Babble Error Interrupt	RW	0x0
9	bnaoutintrmsk	**Value** **Description** 0x0 Mask BNA Interrupt 0x1 No Mask BNA Interrupt	RW	0x0
8	outpkterrmsk	**Value** **Description** 0x0 Mask OUT Packet Error Interrupt 0x1 No Mask OUT Packet Error Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
6	`back2backsetup`	Applies to control OUT endpoints only. **Value** / **Description** 0x0 Mask Back-to-Back SETUP Packets Received Interrupt 0x1 No Mask Back-to-Back SETUP Packets Received Interrupt	RW	0x0
4	`outtknepdismsk`	Applies to control OUT endpoints only. **Value** / **Description** 0x0 Mask OUT Token Received when Endpoint Disabled Interrupt 0x1 No Mask OUT Token Received when Endpoint Disabled Interrupt	RW	0x0
3	`setupmsk`	Applies to control endpoints only. **Value** / **Description** 0x0 Mask SETUP Phase Done Interrupt 0x1 No Mask SETUP Phase Done Interrupt	RW	0x0
2	`ahberrmsk`	**Value** / **Description** 0x0 Mask AHB Error Interrupt 0x1 No Mask AHB Error Interrupt	RW	0x0
1	`epdisbldmsk`	**Value** / **Description** 0x0 Mask Endpoint Disabled Interrupt 0x1 No Mask Endpoint Disabled Interrupt	RW	0x0
0	`xfercomplmsk`	**Value** / **Description** 0x0 Mask Transfer Completed Interrupt 0x1 No Mask Transfer Completed Interrupt	RW	0x0

daint

When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively). This is shown in Figure 5-2.

There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. for a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (DIEPINTn/DOEPINTn).

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00818
usb1	0xFFB40000	0xFFB40818

Offset: 0x818

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
outepint15 RO 0x0	outepint14 RO 0x0	outepint13 RO 0x0	outepint12 RO 0x0	outepint11 RO 0x0	outepint10 RO 0x0	outepint9 RO 0x0	outepint8 RO 0x0	outepint7 RO 0x0	outepint6 RO 0x0	outepint5 RO 0x0	outepint4 RO 0x0	outepint3 RO 0x0	outepint2 RO 0x0	outepint1 RO 0x0	outepint0 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
inepint15 RO 0x0	inepint14 RO 0x0	inepint13 RO 0x0	inepint12 RO 0x0	inepint11 RO 0x0	inepint10 RO 0x0	inepint9 RO 0x0	inepint8 RO 0x0	inepint7 RO 0x0	inepint6 RO 0x0	inepint5 RO 0x0	inepint4 RO 0x0	inepint3 RO 0x0	inepint2 RO 0x0	inepint1 RO 0x0	inepint0 RO 0x0

daint Fields

Bit	Name	Description	Access	Reset
31	outepint15	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 15 Interrupt	RO	0x0
30	outepint14	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 14 Interrupt	RO	0x0
29	outepint13	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 13 Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
28	`outepint12`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 12 Interrupt	RO	0x0
27	`outepint11`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 11 Interrupt	RO	0x0
26	`outepint10`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 10 Interrupt	RO	0x0
25	`outepint9`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 9 Interrupt	RO	0x0
24	`outepint8`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 8 Interrupt	RO	0x0
23	`outepint7`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 7 Interrupt	RO	0x0
22	`outepint6`	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 6 Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
21	outepint5	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 5 Interrupt	RO	0x0
20	outepint4	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 4 Interrupt	RO	0x0
19	outepint3	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 3 Interrupt	RO	0x0
18	outepint2	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 2 Interrupt	RO	0x0
17	outepint1	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 1 Interrupt	RO	0x0
16	outepint0	**Value** **Description** 0x0 No Interrupt 0x1 OUT Endpoint 0 Interrupt	RO	0x0
15	inepint15	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 15 Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
14	inepint14	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 14 Interrupt	RO	0x0
13	inepint13	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 13 Interrupt	RO	0x0
12	inepint12	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 12 Interrupt	RO	0x0
11	inepint11	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 11 Interrupt	RO	0x0
10	inepint10	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 10 Interrupt	RO	0x0
9	inepint9	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 9 Interrupt	RO	0x0
8	inepint8	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 8 Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
7	inepint7	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 7 Interrupt	RO	0x0
6	inepint6	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 6 Interrupt	RO	0x0
5	inepint5	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 5 Interrupt	RO	0x0
4	inepint4	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 4 Interrupt	RO	0x0
3	inepint3	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 3 Interrupt	RO	0x0
2	inepint2	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 2 Interrupt	RO	0x0
1	inepint1	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 1 Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	inepint0	**Value** **Description** 0x0 No Interrupt 0x1 IN Endpoint 0 Interrupt	RO	0x0

daintmsk

The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (DAINT) register bit corresponding to that interrupt is still set.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0081C
usb1	0xFFB40000	0xFFB4081C

Offset: 0x81C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
outepmsk15 RW 0x0	OutEPMsk14 RW 0x0	outepmsk13 RW 0x0	outepmsk12 RW 0x0	outepmsk11 RW 0x0	outepmsk10 RW 0x0	outepmsk9 RW 0x0	outepmsk8 RW 0x0	outepmsk7 RW 0x0	outepmsk6 RW 0x0	outepmsk5 RW 0x0	outepmsk4 RW 0x0	OutEPMsk3 RW 0x0	outepmsk2 RW 0x0	outepmsk1 RW 0x0	outepmsk0 RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
InEpMsk15 RW 0x0	inepmsk14 RW 0x0	InEpMsk13 RW 0x0	inepmsk12 RW 0x0	inepmsk11 RW 0x0	inepmsk10 RW 0x0	inepmsk9 RW 0x0	inepmsk8 RW 0x0	inepmsk7 RW 0x0	inepmsk6 RW 0x0	inepmsk5 RW 0x0	inepmsk4 RW 0x0	inepmsk3 RW 0x0	inepmsk2 RW 0x0	inepmsk1 RW 0x0	inepmsk0 RW 0x0

daintmsk Fields

Bit	Name	Description	Access	Reset
31	outepmsk15	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 15 Interrupt mask	RW	0x0

Bit	Name	Description	Access	Reset
30	`OutEPMsk14`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 14 Interrupt mask	RW	0x0
29	`outepmsk13`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 13 Interrupt mask	RW	0x0
28	`outepmsk12`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 12 Interrupt mask	RW	0x0
27	`outepmsk11`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 11 Interrupt mask	RW	0x0
26	`outepmsk10`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 10 Interrupt mask	RW	0x0
25	`outepmsk9`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 9 Interrupt mask	RW	0x0
24	`outepmsk8`	OUT Endpoint 8 Interrupt mask Bit **Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 8 Interrupt mask	RW	0x0

Bit	Name	Description	Access	Reset
23	`outepmsk7`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 7 Interrupt mask	RW	0x0
22	`outepmsk6`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 6 Interrupt mask	RW	0x0
21	`outepmsk5`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 5 Interrupt mask	RW	0x0
20	`outepmsk4`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 4 Interrupt mask	RW	0x0
19	`OutEPMsk3`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 3 Interrupt mask	RW	0x0
18	`outepmsk2`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 2 Interrupt mask	RW	0x0
17	`outepmsk1`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 1 Interrupt mask	RW	0x0

Bit	Name	Description	Access	Reset
16	`outepmsk0`	**Value** **Description** 0x1 No Interrupt mask 0x0 OUT Endpoint 0 Interrupt mask	RW	0x0
15	`InEpMsk15`	IN Endpoint 15 Interrupt mask Bit **Value** **Description** 0x0 No Interrupt mask 0x1 IN Endpoint 0 Interrupt mask	RW	0x0
14	`inepmsk14`	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 14 Interrupt mask	RW	0x0
13	`InEpMsk13`	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 13 Interrupt mask	RW	0x0
12	`inepmsk12`	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 12 Interrupt mask	RW	0x0
11	`inepmsk11`	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 11 Interrupt mask	RW	0x0
10	`inepmsk10`	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 10 Interrupt mask	RW	0x0

Bit	Name	Description	Access	Reset
9	`inepmsk9`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 0 Interrupt mask	RW	0x0
8	`inepmsk8`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 8 Interrupt mask	RW	0x0
7	`inepmsk7`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 7 Interrupt mask	RW	0x0
6	`inepmsk6`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 6 Interrupt mask	RW	0x0
5	`inepmsk5`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 5 Interrupt mask	RW	0x0
4	`inepmsk4`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 4 Interrupt mask	RW	0x0
3	`inepmsk3`	**Value** — **Description** 0x1 — No Interrupt mask 0x0 — IN Endpoint 3 Interrupt mask	RW	0x0

Bit	Name	Description	Access	Reset
2	inepmsk2	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 2 Interrupt mask	RW	0x0
1	inepmsk1	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 1 Interrupt mask	RW	0x0
0	inepmsk0	**Value** **Description** 0x1 No Interrupt mask 0x0 IN Endpoint 0 Interrupt mask	RW	0x0

dvbusdis

This register specifies the VBUS discharge time after VBUS pulsing during SRP.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00828
usb1	0xFFB40000	0xFFB40828

Offset: 0x828

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dvbusdis RW 0x17D7															

dvbusdis Fields

Bit	Name	Description	Access	Reset
15:0	dvbusdis	This value equals: VBUS discharge time in PHY clocks/1,024 The value you use depends whether the PHY is operating at 30 MHz (16-bit data width) or 60 MHz (8-bit data width). Depending on your VBUS load, this value can need adjustment.	RW	0x17D7

dvbuspulse

This register specifies the VBUS pulsing time during SRP.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0082C
usb1	0xFFB40000	0xFFB4082C

Offset: 0x82C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				dvbuspulse RW 0x5B8											

dvbuspulse Fields

Bit	Name	Description	Access	Reset
11:0	dvbuspulse	Specifies the VBUS pulsing time during SRP. This value equals: VBUS pulsing time in PHY clocks/1,024 The value you use depends whether the PHY is operating at 30MHz (16-bit data width) or 60 MHz (8-bit data width).	RW	0x5B8

dthrctl

Thresholding is not supported in Slave mode and so this register must not be programmed in Slave mode. for threshold support, the AHB must be run at 60 MHz or higher.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00830
usb1	0xFFB40000	0xFFB40830

Offset: 0x830

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				arbprken RW 0x1	Reserved	rxthrlen RW 0x8									rxthren RW 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			ahbthrratio RW 0x0		txthrlen RW 0x8									isothren RW 0x0	nonisothren RW 0x0

dthrctl Fields

Bit	Name	Description	Access	Reset
27	arbprken	This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is Set to one, Then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By Default the parking is enabled. **Value** **Description** 0x0 Disable DMA arbiter parking 0x1 Enable DMA arbiter parking for IN endpoints	RW	0x1
25:17	rxthrlen	This field specifies Receive thresholding size in DWORDS.This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).	RW	0x8
16	rxthren	When this bit is Set, the core enables thresholding in the receive direction. **Value** **Description** 0x0 Disable thresholding 0x1 Enable thresholding in the receive direction	RW	0x0

Bit	Name	Description	Access	Reset
12:11	`ahbthrratio`	These bits define the ratio between the AHB threshold and the MAC threshold for the transmit path only. The AHB threshold always remains less than or equal to the USB threshold, because this does not increase overhead. Both the AHB and the MAC threshold must be DWORD-aligned. The application needs to program TxThrLen and the AHBThrRatio to make the AHB Threshold value DWORD aligned. If the AHB threshold value is not DWORD aligned, the core might not behave correctly. When programming the TxThrLen and AHBThrRatio, the application must ensure that the minimum AHB threshold value does not go below 8 DWORDS to meet the USB turnaround time requirements. **Value** **Description** 0x0 AHB threshold = MAC threshold 0x1 AHB threshold = MAC threshold /2 0x2 AHB threshold = MAC threshold /4 0x3 AHB threshold = MAC threshold /	`RW`	`0x0`
10:2	`txthrlen`	This field specifies Transmit thresholding size in DWORDS. This also forms the MAC threshold and specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmit on the USB. The threshold length has to be at least eight DWORDS when the value of AHBThrRatio is 0. In case the AHBThrRatio is non zero the application needs to ensure that the AHB Threshold value does not go below the recommended eight DWORD. This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (GAHBCFG.HBstLen).	`RW`	`0x8`
1	`isothren`	When this bit is Set, the core enables thresholding for isochronous IN endpoints. **Value** **Description** 0x0 No thresholding 0x1 Enables thresholding	`RW`	`0x0`

Bit	Name	Description	Access	Reset
0	nonisothren	When this bit is Set, the core enables thresholding for Non Isochronous IN endpoints. **Value** **Description** 0x0 No thresholding 0x1 Enable thresholding	RW	0x0

diepempmsk

This register is used to control the IN endpoint FIFO empty interrupt generation (DIEPINTn.TxfEmp).

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00834
usb1	0xFFB40000	0xFFB40834

Offset: 0x834

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfempmsk15 RW 0x0	ineptxfempmsk14 RW 0x0	ineptxfempmsk13 RW 0x0	ineptxfempmsk12 RW 0x0	ineptxfempmsk11 RW 0x0	ineptxfempmsk10 RW 0x0	ineptxfempmsk9 RW 0x0	ineptxfempmsk8 RW 0x0	ineptxfempmsk7 RW 0x0	ineptxfempmsk6 RW 0x0	ineptxfempmsk5 RW 0x0	ineptxfempmsk4 RW 0x0	ineptxfempmsk3 RW 0x0	ineptxfempmsk2 RW 0x0	ineptxfempmsk1 RW 0x0	ineptxfempmsk0 RW 0x0

diepempmsk Fields

Bit	Name	Description	Access	Reset
15	ineptxfempmsk15	This bit acts as mask bits for DIEPINT15. **Value** **Description** 0x0 Mask End point 15 interrupt 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
14	`ineptxfempmsk14`	This bit acts as mask bits for DIEPINT14. **Value** **Description** 0x0 Mask End point 14 interrupt 0x1 No mask	RW	0x0
13	`ineptxfempmsk13`	This bit acts as mask bits for DIEPINT13. **Value** **Description** 0x0 Mask End point 12 interrupt 0x1 No mask	RW	0x0
12	`ineptxfempmsk12`	This bit acts as mask bits for DIEPINT12. **Value** **Description** 0x0 Mask End point 12 interrupt 0x1 No mask	RW	0x0
11	`ineptxfempmsk11`	This bit acts as mask bits for DIEPINT11. **Value** **Description** 0x0 Mask End point 11 interrupt 0x1 No mask	RW	0x0
10	`ineptxfempmsk10`	This bit acts as mask bits for DIEPINT10. **Value** **Description** 0x0 Mask End point 10 interrupt 0x1 No mask	RW	0x0
9	`ineptxfempmsk9`	This bit acts as mask bits for DIEPINT9. **Value** **Description** 0x0 Mask End point 9 interrupt 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
8	`ineptxfempmsk8`	This bit acts as mask bits for DIEPINT8. **Value** **Description** 0x0 Mask End point 8 interrupt 0x1 No mask	RW	0x0
7	`ineptxfempmsk7`	This bit acts as mask bits for DIEPINT7. **Value** **Description** 0x0 Mask End point 7 interrupt 0x1 No mask	RW	0x0
6	`ineptxfempmsk6`	This bit acts as mask bits for DIEPINT6. **Value** **Description** 0x0 Mask End point 6 interrupt 0x1 No mask	RW	0x0
5	`ineptxfempmsk5`	This bit acts as mask bits for DIEPINT5. **Value** **Description** 0x0 Mask End point 5 interrupt 0x1 No mask	RW	0x0
4	`ineptxfempmsk4`	This bit acts as mask bits for DIEPINT4. **Value** **Description** 0x0 Mask End point 4 interrupt 0x1 No mask	RW	0x0
3	`ineptxfempmsk3`	This bit acts as mask bits for DIEPINT3. **Value** **Description** 0x0 Mask End point 3 interrupt 0x1 No mask	RW	0x0

Bit	Name	Description	Access	Reset
2	ineptxfempmsk2	This bit acts as mask bits for DIEPINT2. **Value** **Description** 0x0 Mask End point 2 interrupt 0x1 No mask	RW	0x0
1	ineptxfempmsk1	This bit acts as mask bits for DIEPINT1. **Value** **Description** 0x0 Mask End point 1 interrupt 0x1 No mask	RW	0x0
0	ineptxfempmsk0	This bit acts as mask bits for DIEPINT0. **Value** **Description** 0x0 Mask End point 0 interrupt 0x1 No mask	RW	0x0

diepctl0

This register covers Device Control IN Endpoint 0.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00900
usb1	0xFFB40000	0xFFB40900

Offset: 0x900

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	Reserved		snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RO 0x0		naksts RO 0x0	Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RO 0x1	Reserved													mps RW 0x0	

diepctl0 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>epena</td><td>When Scatter/Gather DMA mode is enabled, for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. When Scatter/Gather DMA mode is disabled such as in bufferpointer based DMA mode this bit indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting the following interrupts on this endpoint: -Endpoint Disabled -Transfer Completed<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No action</td></tr><tr><td>0x1</td><td>Endpoint Enabled</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>30</td><td>epdis</td><td>The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must Set this bit only If Endpoint Enable is already Set for this endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No action</td></tr><tr><td>0x1</td><td>Stop transmitting data on endpoint</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No action</td></tr><tr><td>0x1</td><td>Set NAK</td></tr></table></td><td>WO</td><td>0x0</td></tr>
<tr><td>26</td><td>cnak</td><td>A write to this bit clears the NAK bit for the endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No action</td></tr><tr><td>0x1</td><td>Clear NAK</td></tr></table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25:22	`txfnum`	for Shared FIFO operation, this value is always Set to 0, indicating that control IN endpoint 0 data is always written in the Non-Periodic Transmit FIFO. for Dedicated FIFO operation, this value is Set to the FIFO number that is assigned to IN Endpoint 0.	RW	0x0
21	`stall`	The application can only Set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Nonperiodic IN NAK, or Global OUT NAK is Set along with this bit, the STALL bit takes priority. **Value** — **Description** 0x0 — No Stall 0x1 — Stall Handshake	RO	0x0
19:18	`eptype`	Hardcoded to 00 for control. **Value** — **Description** 0x0 — Endpoint Control 0	RO	0x0
17	`naksts`	When this bit is Set, either by the application or core, the core stops transmitting data, even If there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
15	`usbactep`	This bit is always SET to 1, indicating that control endpoint 0 is always active in all configurations and interfaces. **Value** — **Description** 0x1 — Control endpoint is always active	RO	0x1

Bit	Name	Description	Access	Reset
1:0	mps	Applies to IN and OUT endpoints.The application must program this field with the maximum packet size for the current logical endpoint. **Value** **Description** 0x0 64 bytes 0x1 32 bytes 0x2 16 bytes 0x3 8 bytes	RW	0x0

diepint0

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00908
usb1	0xFFB40000	0xFFB40908

Offset: 0x908

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint0 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td><code>nyetintrpt</code></td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td><code>nakintrpt</code></td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td><code>bbleerr</code></td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td><code>pktdrpsts</code></td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`txfifoundrn`	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	`txfemp`	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	`inepnakeff`	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	`RO`	`0x0`
0	`xfercompl`	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	`RO`	`0x0`

dieptsiz0

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/ DOEPCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 1 to 15. When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00910`
`usb1`	`0xFFB40000`	`0xFFB40910`

Offset: `0x910`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="11">Reserved</td><td colspan="2">pktcnt
RW 0x0</td><td colspan="3">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="9">Reserved</td><td colspan="7">xfersize
RW 0x0</td></tr>
</table>

dieptsiz0 Fields

Bit	Name	Description	Access	Reset
20:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
6:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma0

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00914
usb1	0xFFB40000	0xFFB40914

Offset: `0x914`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">diepdma0
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">diepdma0
RW 0x0</td></tr>
</table>

diepdma0 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma0	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/ Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts0

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00918
usb1	0xFFB40000	0xFFB40918

Offset: 0x918

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts0 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab0

Endpoint 16.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0091C
usb1	0xFFB40000	0xFFB4091C

Offset: 0x91C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab0 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab0 RO 0x0															

diepdmab0 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab0	Used with Scatter/Gather DMA.	RO	0x0

diepctl1

Endpoint_number: 1

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00920
usb1	0xFFB40000	0xFFB40920

Offset: 0x920

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl1 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	WO	0x0
25:22	txfnum	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0

Bit	Name	Description	Access	Reset
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0

Bit	Name	Description	Access	Reset
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint1

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00928
usb1	0xFFB40000	0xFFB40928

Offset: 0x928

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint1 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0

Bit	Name	Description	Access	Reset
7	`txfemp`	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** — **Description** 0x0 — No interrupt 0x1 — Transmit FIFO Empty interrupt	RO	0x1
6	`inepnakeff`	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** — **Description** 0x0 — No interrupt 0x1 — IN Endpoint NAK Effective interrupt	RO	0x0
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** — **Description** 0x0 — No interrupt 0x1 — IN Token Received with EP Mismatch interrupt	RO	0x0

Bit	Name	Description	Access	Reset
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/ Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz1

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00930
usb1	0xFFB40000	0xFFB40930

Offset: `0x930`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz1 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** – **Description** 0x1 – 1 packet 0x2 – 2 packets 0x3 – 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma1

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00934
usb1	0xFFB40000	0xFFB40934

Offset: 0x934

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma1 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma1 RW 0x0															

diepdma1 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma1	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts1

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00938
usb1	0xFFB40000	0xFFB40938

Offset: 0x938

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts1 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab1

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0093C
usb1	0xFFB40000	0xFFB4093C

Offset: 0x93C

Access: RO

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">diepdmab1
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">diepdmab1
RO 0x0</td></tr>
</table>

diepdmab1 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab1	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl2

Endpoint_number: 2

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00940
usb1	0xFFB40000	0xFFB40940

Offset: 0x940

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl2 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	WO	0x0

Bit	Name	Description	Access	Reset
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** **Description** 0x0 Disables Set DATA0 PID 0x1 Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** **Description** 0x0 No Set NAK 0x1 Set NAK	WO	0x0
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	`naksts`	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	`dpid`	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint2

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00948
usb1	0xFFB40000	0xFFB40948

Offset: 0x948

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint2 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	`RO`	`0x0`
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	`RO`	`0x0`
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	`RO`	`0x0`
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	`RO`	`0x0`

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/ Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz2

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00950
usb1	0xFFB40000	0xFFB40950

Offset: 0x950

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">mc
RW 0x0</td><td colspan="10">PktCnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

dieptsiz2 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30:29</td><td>mc</td><td>for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>1 packet</td></tr>
<tr><td>0x2</td><td>2 packets</td></tr>
<tr><td>0x3</td><td>3 packets</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>28:19</td><td>PktCnt</td><td>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
<tr><td>18:0</td><td>xfersize</td><td>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
</table>

diepdma2

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00954
usb1	0xFFB40000	0xFFB40954

Offset: 0x954

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma2 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma2 RW 0x0															

diepdma2 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma2	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

DTXFSTS2

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00958
usb1	0xFFB40000	0xFFB40958

Offset: 0x958

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

DTXFSTS2 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab2

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0095C
usb1	0xFFB40000	0xFFB4095C

Offset: 0x95C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab2 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab2 RO 0x0															

diepdmab2 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab2	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl3

Endpoint_number: 3

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00960
usb1	0xFFB40000	0xFFB40960

Offset: 0x960

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl3 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** **Description** 0x0 Disables Set DATA0 PID 0x1 Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** **Description** 0x0 No Set NAK 0x1 Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint3

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00968`
`usb1`	`0xFFB40000`	`0xFFB40968`

Offset: `0x968`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint3 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** — **Description** 0x0 — No interrupt 0x1 — BNA interrupt	RO	0x0
8	`txfifoundrn`	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — Fifo Underrun interrupt	RO	0x0
7	`txfemp`	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** — **Description** 0x0 — No interrupt 0x1 — Transmit FIFO Empty interrupt	RO	0x1
6	`inepnakeff`	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** — **Description** 0x0 — No interrupt 0x1 — IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

dieptsiz3

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00970
usb1	0xFFB40000	0xFFB40970

Offset: 0x970

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz3 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma3

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00974
usb1	0xFFB40000	0xFFB40974

Offset: 0x974

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma3 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma3 RW 0x0															

diepdma3 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma3	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts3

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00978
usb1	0xFFB40000	0xFFB40978

Offset: 0x978

Access: RO

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ineptxfspcavail
RO 0x2000</td></tr>
</table>

dtxfsts3 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab3

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0097C
usb1	0xFFB40000	0xFFB4097C

Offset: 0x97C

Access: RO

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">diepdmab3
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">diepdmab3
RO 0x0</td></tr>
</table>

diepdmab3 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab3	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl4

Endpoint_number: 4

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00980
usb1	0xFFB40000	0xFFB40980

Offset: 0x980

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">txfnum
RW 0x0</td><td>stall
RO 0x0</td><td>Reserved</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

diepctl4 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	WO	0x0

Bit	Name	Description	Access	Reset
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** / **Description** 0x0 Disables Set DATA0 PID 0x1 Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** / **Description** 0x0 No Set NAK 0x1 Set NAK	WO	0x0
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>25:22</td><td>txfnum</td><td>Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.</td><td>RW</td><td>0x0</td></tr>
<tr><td>21</td><td>stall</td><td>Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>STALL All Tokens not active</td></tr>
<tr><td>0x1</td><td>STALL All Tokens active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>19:18</td><td>eptype</td><td>This is the transfer type supported by this logical endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Control</td></tr>
<tr><td>0x1</td><td>Isochronous</td></tr>
<tr><td>0x2</td><td>Bulk</td></tr>
<tr><td>0x3</td><td>Interrupt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
17	`naksts`	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	`dpid`	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint4

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00988
usb1	0xFFB40000	0xFFB40988

Offset: 0x988

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint4 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** — **Description** 0x0 — No interrupt 0x1 — BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** — **Description** 0x0 — No interrupt 0x1 — Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** — **Description** 0x0 — No interrupt 0x1 — IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** / **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0
0	`xfercompl`	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

dieptsiz4

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00990
usb1	0xFFB40000	0xFFB40990

Offset: `0x990`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz4 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** **Description** 0x1 1 packet 0x2 2 packets 0x3 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma4

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00994
usb1	0xFFB40000	0xFFB40994

Offset: 0x994

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma4 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma4 RW 0x0															

diepdma4 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma4	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts4

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00998
usb1	0xFFB40000	0xFFB40998

Offset: 0x998

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts4 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab4

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB0099C
usb1	0xFFB40000	0xFFB4099C

Offset: 0x99C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab4 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab4 RO 0x0															

diepdmab4 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab4	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl5

Endpoint_number: 5

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009A0
usb1	0xFFB40000	0xFFB409A0

Offset: 0x9A0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl5 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	`WO`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>26</td><td>cnak</td><td>A write to this bit clears the NAK bit for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Clear NAK</td></tr>
<tr><td>0x1</td><td>Clear NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint5

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB009A8`
`usb1`	`0xFFB40000`	`0xFFB409A8`

Offset: `0x9A8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint5 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz5

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009B0
usb1	0xFFB40000	0xFFB409B0

Offset: 0x9B0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">mc
RW 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

dieptsiz5 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30:29</td><td>mc</td><td>for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>1 packet</td></tr>
<tr><td>0x2</td><td>2 packets</td></tr>
<tr><td>0x3</td><td>3 packets</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>28:19</td><td>pktcnt</td><td>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
<tr><td>18:0</td><td>xfersize</td><td>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
</table>

diepdma5

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009B4
usb1	0xFFB40000	0xFFB409B4

Offset: 0x9B4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma5 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma5 RW 0x0															

diepdma5 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma5	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts5

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009B8
usb1	0xFFB40000	0xFFB409B8

Offset: 0x9B8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts5 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab5

Device IN Endpoint 1 Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009BC
usb1	0xFFB40000	0xFFB409BC

Offset: 0x9BC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab5 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab5 RO 0x0															

diepdmab5 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab5	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl6

Endpoint_number: 6

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009C0
usb1	0xFFB40000	0xFFB409C0

Offset: 0x9C0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl6 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** / **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint6

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009C8
usb1	0xFFB40000	0xFFB409C8

Offset: `0x9C8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint6 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** / **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz6

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009D0
usb1	0xFFB40000	0xFFB409D0

Offset: 0x9D0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz6 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** **Description** 0x1 1 packet 0x2 2 packets 0x3 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma6

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009D4
usb1	0xFFB40000	0xFFB409D4

Offset: 0x9D4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma6 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma6 RW 0x0															

diepdma6 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma6	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts6

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009D8
usb1	0xFFB40000	0xFFB409D8

Offset: 0x9D8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts6 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab6

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009DC
usb1	0xFFB40000	0xFFB409DC

Offset: 0x9DC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab6 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab6 RO 0x0															

diepdmab6 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab6	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl7

Endpoint_number: 7

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009E0
usb1	0xFFB40000	0xFFB409E0

Offset: 0x9E0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl7 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint7

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009E8
usb1	0xFFB40000	0xFFB409E8

Offset: 0x9E8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint7 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** / **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** / **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** / **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** / **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	`RO`	`0x0`
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	`RO`	`0x0`
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	`RO`	`0x0`
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	`RO`	`0x0`

Bit	Name	Description	Access	Reset
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0
0	`xfercompl`	Applies to IN and OUT endpoints. When Scatter/ Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

dieptsiz7

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB009F0`
`usb1`	`0xFFB40000`	`0xFFB409F0`

Offset: `0x9F0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz7 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma7

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009F4
usb1	0xFFB40000	0xFFB409F4

Offset: 0x9F4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma7 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma7 RW 0x0															

diepdma7 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma7	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts7

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009F8
usb1	0xFFB40000	0xFFB409F8

Offset: 0x9F8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts7 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab7

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB009FC
usb1	0xFFB40000	0xFFB409FC

Offset: 0x9FC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab7 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab7 RO 0x0															

diepdmab7 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab7	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl8

Endpoint_number: 8

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A00
usb1	0xFFB40000	0xFFB40A00

Offset: 0xA00

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl8 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr><tr><td>0x1</td><td>Endpoint Data PID to DATA0)</td></tr></table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Set NAK</td></tr><tr><td>0x1</td><td>Set NAK</td></tr></table></td><td>WO</td><td>0x0</td></tr>
<tr><td>26</td><td>cnak</td><td>A write to this bit clears the NAK bit for the endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Clear NAK</td></tr><tr><td>0x1</td><td>Clear NAK</td></tr></table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint8

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A08
usb1	0xFFB40000	0xFFB40A08

Offset: 0xA08

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint8 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>9</td><td>bnaintr</td><td>This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No interrupt</td></tr><tr><td>0x1</td><td>BNA interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>8</td><td>txfifoundrn</td><td>Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No interrupt</td></tr><tr><td>0x1</td><td>Fifo Underrun interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>7</td><td>txfemp</td><td>This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)).<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No interrupt</td></tr><tr><td>0x1</td><td>Transmit FIFO Empty interrupt</td></tr></table></td><td>RO</td><td>0x1</td></tr>
<tr><td>6</td><td>inepnakeff</td><td>Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No interrupt</td></tr><tr><td>0x1</td><td>IN Endpoint NAK Effective interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz8

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A10
usb1	0xFFB40000	0xFFB40A10

Offset: 0xA10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz8 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma8

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A14
usb1	0xFFB40000	0xFFB40A14

Offset: 0xA14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma8 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma8 RW 0x0															

diepdma8 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma8	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts8

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A18
usb1	0xFFB40000	0xFFB40A18

Offset: 0xA18

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts8 Fields

Bit	Name	Description	Access	Reset
15:0	`ineptxfspcavail`	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab8

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A1C
usb1	0xFFB40000	0xFFB40A1C

Offset: `0xA1C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab8 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab8 RO 0x0															

diepdmab8 Fields

Bit	Name	Description	Access	Reset
31:0	`diepdmab8`	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl9

Endpoint_number: 9

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A20
usb1	0xFFB40000	0xFFB40A20

Offset: 0xA20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl9 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	WO	0x0

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** **Description** 0x0 Disables Set DATA0 PID 0x1 Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** **Description** 0x0 No Set NAK 0x1 Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** / **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint9

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00A28`
`usb1`	`0xFFB40000`	`0xFFB40A28`

Offset: `0xA28`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint9 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz9

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A30
usb1	0xFFB40000	0xFFB40A30

Offset: 0xA30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz9 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma9

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A34
usb1	0xFFB40000	0xFFB40A34

Offset: 0xA34

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma9 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma9 RW 0x0															

diepdma9 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma9	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts9

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A38
usb1	0xFFB40000	0xFFB40A38

Offset: 0xA38

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts9 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab9

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A3C
usb1	0xFFB40000	0xFFB40A3C

Offset: 0xA3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab9 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab9 RO 0x0															

diepdmab9 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab9	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl10

Endpoint_number: 10

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A40
usb1	0xFFB40000	0xFFB40A40

Offset: 0xA40

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">txfnum
RW 0x0</td><td>stall
RO 0x0</td><td>Reserved</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

diepctl10 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint10

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A48
usb1	0xFFB40000	0xFFB40A48

Offset: 0xA48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint10 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** / **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** / **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** / **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** / **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	intknepmis	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	intkntxfemp	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	timeout	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** / **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz10

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A50
usb1	0xFFB40000	0xFFB40A50

Offset: 0xA50

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">mc
RW 0x0</td><td colspan="10">PktCnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

dieptsiz10 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** **Description** 0x1 1 packet 0x2 2 packets 0x3 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma10

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A54
usb1	0xFFB40000	0xFFB40A54

Offset: 0xA54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma10 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma10 RW 0x0															

diepdma10 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma10	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts10

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A58
usb1	0xFFB40000	0xFFB40A58

Offset: 0xA58

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts10 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab10

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A5C
usb1	0xFFB40000	0xFFB40A5C

Offset: 0xA5C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab10 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab10 RO 0x0															

diepdmab10 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab10	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl11

Endpoint_number: 11

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A60
usb1	0xFFB40000	0xFFB40A60

Offset: 0xA60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl11 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30</td><td>epdis</td><td>Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Endpoint Disable</td></tr>
<tr><td>0x1</td><td>Endpoint Disable</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>26</td><td>cnak</td><td>A write to this bit clears the NAK bit for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Clear NAK</td></tr>
<tr><td>0x1</td><td>Clear NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** / **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint11

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00A68`
`usb1`	`0xFFB40000`	`0xFFB40A68`

Offset: `0xA68`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint11 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	`xfercompl`	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz11

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00A70`
`usb1`	`0xFFB40000`	`0xFFB40A70`

Offset: `0xA70`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz11 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** **Description** 0x1 1 packet 0x2 2 packets 0x3 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma11

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A74
usb1	0xFFB40000	0xFFB40A74

Offset: 0xA74

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma11 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma11 RW 0x0															

diepdma11 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma11	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts11

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A78
usb1	0xFFB40000	0xFFB40A78

Offset: 0xA78

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts11 Fields

Bit	Name	Description	Access	Reset
15:0	`ineptxfspcavail`	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab11

DMA Buffer Address.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00A7C`
`usb1`	`0xFFB40000`	`0xFFB40A7C`

Offset: `0xA7C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab11 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab11 RO 0x0															

diepdmab11 Fields

Bit	Name	Description	Access	Reset
31:0	`diepdmab11`	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl12

Endpoint_number: 12

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A80
usb1	0xFFB40000	0xFFB40A80

Offset: 0xA80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl12 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30</td><td>epdis</td><td>Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Endpoint Disable</td></tr>
<tr><td>0x1</td><td>Endpoint Disable</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>26</td><td>cnak</td><td>A write to this bit clears the NAK bit for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Clear NAK</td></tr>
<tr><td>0x1</td><td>Clear NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** / **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint12

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00A88`
`usb1`	`0xFFB40000`	`0xFFB40A88`

Offset: `0xA88`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint12 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** — **Description** 0x0 — No interrupt 0x1 — BNA interrupt	RO	0x0
8	`txfifoundrn`	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — Fifo Underrun interrupt	RO	0x0
7	`txfemp`	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** — **Description** 0x0 — No interrupt 0x1 — Transmit FIFO Empty interrupt	RO	0x1
6	`inepnakeff`	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** — **Description** 0x0 — No interrupt 0x1 — IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** / **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** / **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz12

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A90
usb1	0xFFB40000	0xFFB40A90

Offset: 0xA90

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">mc
RW 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

dieptsiz12 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma12

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A94
usb1	0xFFB40000	0xFFB40A94

Offset: 0xA94

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma12 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma12 RW 0x0															

diepdma12 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma12	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts12

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A98
usb1	0xFFB40000	0xFFB40A98

Offset: 0xA98

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts12 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab12

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00A9C
usb1	0xFFB40000	0xFFB40A9C

Offset: 0xA9C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab12 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab12 RO 0x0															

diepdmab12 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab12	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl13

Endpoint_number: 13

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AA0
usb1	0xFFB40000	0xFFB40AA0

Offset: 0xAA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl13 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** **Description** 0x0 Disables Set DATA0 PID 0x1 Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** **Description** 0x0 No Set NAK 0x1 Set NAK	WO	0x0
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0

Bit	Name	Description	Access	Reset
25:22	txfnum	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint13

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AA8
usb1	0xFFB40000	0xFFB40AA8

Offset: 0xAA8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint13 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	`RO`	`0x0`
8	`txfifoundrn`	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	`RO`	`0x0`
7	`txfemp`	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	`RO`	`0x1`
6	`inepnakeff`	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	`RO`	`0x0`

Bit	Name	Description	Access	Reset
5	intknepmis	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	intkntxfemp	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	timeout	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. Value / Description 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. Value / Description 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz13

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AB0
usb1	0xFFB40000	0xFFB40AB0

Offset: 0xAB0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz13 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma13

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AB4
usb1	0xFFB40000	0xFFB40AB4

Offset: 0xAB4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma13 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma13 RW 0x0															

diepdma13 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma13	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts13

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AB8
usb1	0xFFB40000	0xFFB40AB8

Offset: 0xAB8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts13 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab13

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00ABC
usb1	0xFFB40000	0xFFB40ABC

Offset: 0xABC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab13 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab13 RO 0x0															

diepdmab13 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab13	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl14

Endpoint_number: 14

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AC0
usb1	0xFFB40000	0xFFB40AC0

Offset: 0xAC0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved					mps RW 0x0									

diepctl14 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** / **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** / **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint14

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00AC8`
`usb1`	`0xFFB40000`	`0xFFB40AC8`

Offset: `0xAC8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint14 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

Bit	Name	Description	Access	Reset
5	`intknepmis`	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received with EP Mismatch interrupt	RO	0x0
4	`intkntxfemp`	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received. **Value** **Description** 0x0 No interrupt 0x1 IN Token Received Interrupt	RO	0x0
3	`timeout`	In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Timeout interrupy	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0

Bit	Name	Description	Access	Reset
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

dieptsiz14

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AD0
usb1	0xFFB40000	0xFFB40AD0

Offset: `0xAD0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz14 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30:29</td><td>mc</td><td>for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp)
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x1</td><td>1 packet</td></tr>
<tr><td>0x2</td><td>2 packets</td></tr>
<tr><td>0x3</td><td>3 packets</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>28:19</td><td>PktCnt</td><td>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
<tr><td>18:0</td><td>xfersize</td><td>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.</td><td>RW</td><td>0x0</td></tr>
</table>

diepdma14

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AD4
usb1	0xFFB40000	0xFFB40AD4

Offset: 0xAD4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma14 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma14 RW 0x0															

diepdma14 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma14	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts14

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AD8
usb1	0xFFB40000	0xFFB40AD8

Offset: 0xAD8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts14 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab14

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00ADC
usb1	0xFFB40000	0xFFB40ADC

Offset: 0xADC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab14 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab14 RO 0x0															

diepdmab14 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab14	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

diepctl15

Endpoint_number: 15

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AE0
usb1	0xFFB40000	0xFFB40AE0

Offset: 0xAE0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	txfnum RW 0x0				stall RO 0x0	Reserved	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

diepctl15 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	`RO`	`0x0`
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	`WO`	`0x0`

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
25:22	txfnum	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	RW	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

diepint15

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00AE8`
`usb1`	`0xFFB40000`	`0xFFB40AE8`

Offset: `0xAE8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	txfifoundrn RO 0x0	txfemp RO 0x1	inepnakeff RO 0x0	intknepmis RO 0x0	intkntxfemp RO 0x0	timeout RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

diepint15 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	txfifoundrn	Applies to IN endpoints Only. The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. **Value** **Description** 0x0 No interrupt 0x1 Fifo Underrun interrupt	RO	0x0
7	txfemp	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (GAHBCFG.NPTxFEmpLvl)). **Value** **Description** 0x0 No interrupt 0x1 Transmit FIFO Empty interrupt	RO	0x1
6	inepnakeff	Applies to periodic IN endpoints only. This bit can be cleared when the application clears the IN endpoint NAK by writing to DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit Set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit Set by the application has taken effect in the core.This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit. **Value** **Description** 0x0 No interrupt 0x1 IN Endpoint NAK Effective interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>5</td><td>intknepmis</td><td>Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>IN Token Received with EP Mismatch interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>4</td><td>intkntxfemp</td><td>Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>IN Token Received Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>timeout</td><td>In shared TX FIFO mode, applies to non-isochronous IN endpoints only. In dedicated FIFO mode, applies only to Control IN endpoints. In Scatter/Gather DMA mode, the TimeOUT interrupt is notasserted. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Timeout interrupy</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>ahberr</td><td>Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Interrupt</td></tr>
<tr><td>0x1</td><td>AHB Error interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	`xfercompl`	Applies to IN and OUT endpoints. When Scatter/Gather DMA mode is enabled - for IN endpoint this field indicates that the requested data from the descriptor is moved from external system memory to internal FIFO. - for OUT endpoint this field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** / **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

dieptsiz15

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00AF0`
`usb1`	`0xFFB40000`	`0xFFB40AF0`

Offset: `0xAF0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	mc RW 0x0		PktCnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

dieptsiz15 Fields

Bit	Name	Description	Access	Reset
30:29	mc	for periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints. for non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core must fetchfor an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (DIEPCTLn.NextEp) **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
28:19	PktCnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	0x0

diepdma15

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AF4
usb1	0xFFB40000	0xFFB40AF4

Offset: 0xAF4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdma15 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdma15 RW 0x0															

diepdma15 Fields

Bit	Name	Description	Access	Reset
31:0	diepdma15	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/ Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

dtxfsts15

This register contains the free space information for the Device IN endpoint TxFIFO.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AF8
usb1	0xFFB40000	0xFFB40AF8

Offset: 0xAF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ineptxfspcavail RO 0x2000															

dtxfsts15 Fields

Bit	Name	Description	Access	Reset
15:0	ineptxfspcavail	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. 16'h0: Endpoint TxFIFO is full 16'h1: 1 word available 16'h2: 2 words available 16'hn: n words available (where 0 n 32,768) 16'h8000: 32,768 words available Others: Reserved	RO	0x2000

diepdmab15

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00AFC
usb1	0xFFB40000	0xFFB40AFC

Offset: 0xAFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
diepdmab15 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
diepdmab15 RO 0x0															

diepdmab15 Fields

Bit	Name	Description	Access	Reset
31:0	diepdmab15	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl0

This is Control OUT Endpoint 0 Control register.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B00
usb1	0xFFB40000	0xFFB40B00

Offset: 0xB00

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td colspan="2">Reserved</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">Reserved</td><td>stall
RO 0x0</td><td>snp
RW 0x0</td><td colspan="2">eptype
RO 0x0</td><td>naksts
RO 0x0</td><td>Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RO 0x1</td><td colspan="13">Reserved</td><td colspan="2">mps
RO 0x0</td></tr>
</table>

doepctl0 Fields

Bit	Name	Description	Access	Reset
31	epena	When Scatter/Gather DMA mode is enabled, for OUT endpoints this bit indicates that the descriptor structure and data buffer to receive data is setup. When Scatter/Gather DMA mode is disabled(such as for buffer-pointer based DMA mode)this bit indicates that the application has allocated the memory to start receiving data from the USB.The core clears this bit before setting any of the following interrupts on this endpoint: SETUP Phase Done Endpoint Disabled Transfer Completed In DMA mode, this bit must be Set for the core to transfer SETUP data packets into memory. **Value** — **Description** 0x0 — No action 0x1 — Endpoint Enabled	RO	0x0
30	epdis	The application cannot disable control OUT endpoint 0. **Value** — **Description** 0x0 — No Endpoint disable	RO	0x0

Bit	Name	Description	Access	Reset
27	snak	A write to this bit sets the NAK bit for the endpoint.Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint. **Value** / **Description** 0x0 / No action 0x1 / Set NAK	WO	0x0
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 / No action 0x1 / Clear NAK	WO	0x0
21	stall	The application can only Set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is Set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 / No Stall 0x1 / Stall Handshake	RO	0x0
20	snp	This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 / Snoop Mode disabled 0x1 / Snoop Mode enabled	RW	0x0
19:18	eptype	Hardcoded to 0 for control. **Value** / **Description** 0x0 / Endpoint Control 0	RO	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit, the core stops receiving data, even If there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
15	usbactep	This bit is always Set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces. **Value** / **Description** 0x1 USB Active Endpoint 0	RO	0x1
1:0	mps	The maximum packet size for control OUT endpoint 0 is thesame as what is programmed in control IN Endpoint 0. **Value** / **Description** 0x0 64 bytes 0x1 32 bytes 0x2 16 bytes 0x3 8 bytes	RO	0x0

doepint0

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B08
usb1	0xFFB40000	0xFFB40B08

Offset: 0xB08

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint0 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0

Bit	Name	Description	Access	Reset
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** / **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** / **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** / **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0

Bit	Name	Description	Access	Reset
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** — **Description** 0x0 — No Status Phase Received for Control Write 0x1 — Status Phase Received for Control Write	`RO`	`0x0`
4	`outtknepdis`	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** — **Description** 0x0 — No OUT Token Received When Endpoint Disabled 0x1 — OUT Token Received When Endpoint Disabled	`RO`	`0x0`
3	`setup`	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** — **Description** 0x0 — No SETUP Phase Done 0x1 — SETUP Phase Done	`RO`	`0x0`

Bit	Name	Description	Access	Reset
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0
0	`xfercompl`	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** / **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz0

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/ DOEPCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit. Nonzero endpoints use the registers for endpoints 1 to 15. When Scatter/Gather DMA mode is enabled, this register must not be programmed by the application. If the application reads this register when Scatter/Gather DMA mode is enabled, the core returns all zeros.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B10`
`usb1`	`0xFFB40000`	`0xFFB40B10`

Offset: 0xB10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	supcnt RW 0x0		Reserved									pktcnt RW 0x0	Reserved		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									xfersize RW 0x0						

doeptsiz0 Fields

Bit	Name	Description	Access	Reset
30:29	supcnt	SETUP Packet Count (SUPCnt)This field specifies the number of back-to-back SETUP datapackets the endpoint can receive. **Value** — **Description** 0x1 — 1 packet 0x2 — 2 packets 0x3 — 3 packets	RW	0x0
19	pktcnt	This field is decremented to zero after a packet is written into the RxFIFO.	RW	0x0
6:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma0

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B14
usb1	0xFFB40000	0xFFB40B14

Offset: 0xB14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma0 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma0 RW 0x0															

doepdma0 Fields

Bit	Name	Description	Access	Reset
31:0	`doepdma0`	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab0

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B1C
usb1	0xFFB40000	0xFFB40B1C

Offset: 0xB1C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab0 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab0 RO 0x0															

doepdmab0 Fields

Bit	Name	Description	Access	Reset
31:0	`doepdmab0`	Used with Scatter/Gather DMA.	RO	0x0

doepctl1

Out Endpoint 1.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B20
usb1	0xFFB40000	0xFFB40B20

Offset: 0xB20

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl1 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled - Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0

Bit	Name	Description	Access	Reset
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	RO	0x0
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0

Bit	Name	Description	Access	Reset
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`

Bit	Name	Description	Access	Reset
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. h 1'b0'b0: Even (micro)frame 1: Odd (micro)frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint1

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B28`
`usb1`	`0xFFB40000`	`0xFFB40B28`

Offset: `0xB28`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint1 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>4</td><td>outtknepdis</td><td>Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No OUT Token Received When Endpoint Disabled</td></tr>
<tr><td>0x1</td><td>OUT Token Received When Endpoint Disabled</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>setup</td><td>Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No SETUP Phase Done</td></tr>
<tr><td>0x1</td><td>SETUP Phase Done</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>ahberr</td><td>Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Interrupt</td></tr>
<tr><td>0x1</td><td>AHB Error interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>1</td><td>epdisbld</td><td>Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Interrupt</td></tr>
<tr><td>0x1</td><td>Endpoint Disabled Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz1

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B30
usb1	0xFFB40000	0xFFB40B30

Offset: `0xB30`

Access: `RW`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rxdpid RO 0x0		pktcnt RW 0x0										xfersize RW 0x0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

doeptsiz1 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 or 1 packet 0x2 — DATA1 or 2 packets 0x3 — MDATA or 3 packets	`RO`	`0x0`
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	`RW`	`0x0`
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	`RW`	`0x0`

doepdma1

DMA Addressing.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B34`
`usb1`	`0xFFB40000`	`0xFFB40B34`

Offset: `0xB34`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma1 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma1 RW 0x0															

doepdma1 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma1	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab1

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B3C
usb1	0xFFB40000	0xFFB40B3C

Offset: 0xB3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab1 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab1 RO 0x0															

doepdmab1 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab1	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

DOEPCTL2

Out Endpoint 2.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B40
usb1	0xFFB40000	0xFFB40B40

Offset: 0xB40

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">Reserved</td><td>stall
RO 0x0</td><td>snp
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

DOEPCTL2 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Enables Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. Value — Description 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. Value — Description 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`
20	`snp`	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. Value — Description 0x0 — Disable Snoop Mode 0x1 — Enable Snoop Mode	`RW`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. Value — Description 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint2

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B48
usb1	0xFFB40000	0xFFB40B48

Offset: 0xB48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint2 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	`outtknepdis`	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** — **Description** 0x0 — No OUT Token Received When Endpoint Disabled 0x1 — OUT Token Received When Endpoint Disabled	RO	0x0
3	`setup`	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** — **Description** 0x0 — No SETUP Phase Done 0x1 — SETUP Phase Done	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** — **Description** 0x0 — No Interrupt 0x1 — AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz2

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B50
usb1	0xFFB40000	0xFFB40B50

Offset: 0xB50

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz2 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** / **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma2

DMA Addressing.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B54
usb1	0xFFB40000	0xFFB40B54

Offset: `0xB54`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma2 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma2 RW 0x0															

doepdma2 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma2	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab2

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B5C
usb1	0xFFB40000	0xFFB40B5C

Offset: 0xB5C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab2 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab2 RO 0x0															

doepdmab2 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab2	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

DOEPCTL3

Out Endpoint 3.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B60
usb1	0xFFB40000	0xFFB40B60

Offset: 0xB60

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">Reserved</td><td>stall
RO 0x0</td><td>snp
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

DOEPCTL3 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	WO	0x0
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** **Description** 0x0 Disables Set DATA0 PID 0x1 Enables Endpoint Data PID to DATA0)	WO	0x0
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** **Description** 0x0 No Set NAK 0x1 Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	`WO`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	`RO`	`0x0`
20	`snp`	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	`RW`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table>
</td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table>
</td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint3

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B68`
`usb1`	`0xFFB40000`	`0xFFB40B68`

Offset: `0xB68`

Access: `RO`

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint3 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz3

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B70
usb1	0xFFB40000	0xFFB40B70

Offset: 0xB70

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz3 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** / **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma3

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B74
usb1	0xFFB40000	0xFFB40B74

Offset: 0xB74

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma3 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma3 RW 0x0															

doepdma3 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma3	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab3

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B7C
usb1	0xFFB40000	0xFFB40B7C

Offset: 0xB7C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab3 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab3 RO 0x0															

doepdmab3 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab3	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl4

Out Endpoint 4.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B80
usb1	0xFFB40000	0xFFB40B80

Offset: 0xB80

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl4 Fields

Bit	Name	Description	Access	Reset
31	`epena`	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	WO	0x0
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	WO	0x0
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
20	`snp`	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** — **Description** 0x0 — Disable Snoop Mode 0x1 — Enable Snoop Mode	RW	0x0
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

Doepint4

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B88`
`usb1`	`0xFFB40000`	`0xFFB40B88`

Offset: `0xB88`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

Doepint4 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz4

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B90
usb1	0xFFB40000	0xFFB40B90

Offset: 0xB90

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz4 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	`RO`	`0x0`
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	`RW`	`0x0`
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	`RW`	`0x0`

doepdma4

DMA OUT Address.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00B94`
`usb1`	`0xFFB40000`	`0xFFB40B94`

Offset: `0xB94`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma4 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma4 RW 0x0															

doepdma4 Fields

Bit	Name	Description	Access	Reset
31:0	`doepdma4`	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab4

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00B9C
usb1	0xFFB40000	0xFFB40B9C

Offset: `0xB9C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab4 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab4 RO 0x0															

doepdmab4 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab4	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl5

Out Endpoint 5.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BA0
usb1	0xFFB40000	0xFFB40BA0

Offset: 0xBA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved					mps RW 0x0									

doepctl5 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. Value / Description 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. Value / Description 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. Value / Description 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. Value / Description 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint5

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BA8
usb1	0xFFB40000	0xFFB40BA8

Offset: 0xBA8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint5 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** / **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** / **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz5

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BB0
usb1	0xFFB40000	0xFFB40BB0

Offset: `0xBB0`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz5 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30:29</td><td>rxdpid</td><td>Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>DATA0</td></tr><tr><td>0x1</td><td>DATA2 or 1 packet</td></tr><tr><td>0x2</td><td>DATA1 or 2 packets</td></tr><tr><td>0x3</td><td>MDATA or 3 packets</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>28:19</td><td>pktcnt</td><td>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.</td><td>RW</td><td>0x0</td></tr>
<tr><td>18:0</td><td>xfersize</td><td>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.</td><td>RW</td><td>0x0</td></tr>
</table>

doepdma5

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BB4
usb1	0xFFB40000	0xFFB40BB4

Offset: 0xBB4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma5 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma5 RW 0x0															

doepdma5 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma5	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab5

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BBC
usb1	0xFFB40000	0xFFB40BBC

Offset: 0xBBC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab5 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab5 RO 0x0															

doepdmab5 Fields

Bit	Name	Description	Access	Reset
31:0	`doepdmab5`	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl6

Out Endpoint 6.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BC0
usb1	0xFFB40000	0xFFB40BC0

Offset: `0xBC0`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl6 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled - Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** — **Description** 0x0 — Endpoint Enable inactive 0x1 — Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** — **Description** 0x0 — No Endpoint Disable 0x1 — Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	`WO`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	`RO`	`0x0`
20	`snp`	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	`RW`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint6

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00BC8`
`usb1`	`0xFFB40000`	`0xFFB40BC8`

Offset: `0xBC8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint6 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** — **Description** 0x0 — No interrupt 0x1 — BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** — **Description** 0x0 — No OUT Packet Error 0x1 — OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** — **Description** 0x0 — No Back-to-Back SETUP Packets Received 0x1 — Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** — **Description** 0x0 — No Status Phase Received for Control Write 0x1 — Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz6

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BD0
usb1	0xFFB40000	0xFFB40BD0

Offset: 0xBD0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz6 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 or 1 packet 0x2 — DATA1 or 2 packets 0x3 — MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma6

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BD4
usb1	0xFFB40000	0xFFB40BD4

Offset: 0xBD4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma6 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma6 RW 0x0															

doepdma6 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma6	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab6

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BDC
usb1	0xFFB40000	0xFFB40BDC

Offset: 0xBDC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab6 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab6 RO 0x0															

doepdmab6 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab6	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl7

Endpoint_number: 7

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BE0
usb1	0xFFB40000	0xFFB40BE0

Offset: 0xBE0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">txfnum
RW 0x0</td><td>stall
RO 0x0</td><td>Reserved</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

doepctl7 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled - Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Endpoint Data PID to DATA0)	WO	0x0
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	`WO`	`0x0`
25:22	`txfnum`	Shared FIFO Operation-non-periodic endpoints must set this bit to zero. Periodic endpoints must map this to the corresponding Periodic TxFIFO number. 4'h0: Non-Periodic TxFIFO Others: Specified Periodic TxFIFO.number An interrupt IN endpoint can be configured as a non-periodic endpoint for applications such as mass storage. The core treats an IN endpoint as a non-periodic endpoint if the TxFNum field is set to 0. Configuring an interrupt IN endpoint as a non-periodic endpoint saves the extra periodic FIFO area. Dedicated FIFO Operation-these bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number. This field is valid only for IN endpoints.	`RW`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	`RO`	`0x0`

Bit	Name	Description	Access	Reset
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0

Bit	Name	Description	Access	Reset
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** / **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** / **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint7

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BE8
usb1	0xFFB40000	0xFFB40BE8

Offset: 0xBE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint7 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** / **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0

Bit	Name	Description	Access	Reset
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0

Bit	Name	Description	Access	Reset
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** — **Description** 0x0 — No SETUP Phase Done 0x1 — SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** — **Description** 0x0 — No Interrupt 0x1 — AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz7

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BF0
usb1	0xFFB40000	0xFFB40BF0

Offset: 0xBF0

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rxdpid RO 0x0		pktcnt RW 0x0										xfersize RW 0x0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

doeptsiz7 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0

Bit	Name	Description	Access	Reset
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	`RW`	`0x0`

doepdma7

DMA OUT Address.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00BF4`
`usb1`	`0xFFB40000`	`0xFFB40BF4`

Offset: `0xBF4`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">doepdma7
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">doepdma7
RW 0x0</td></tr>
</table>

doepdma7 Fields

Bit	Name	Description	Access	Reset
31:0	`doepdma7`	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	`RW`	`0x0`

doepdmab7

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00BFC
usb1	0xFFB40000	0xFFB40BFC

Offset: 0xBFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab7 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab7 RO 0x0															

doepdmab7 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab7	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl8

Out Endpoint 8.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C00
usb1	0xFFB40000	0xFFB40C00

Offset: 0xC00

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl8 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled - Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Enables Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** — **Description** 0x0 — No Clear NAK 0x1 — Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — STALL All Tokens not active 0x1 — STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** — **Description** 0x0 — Disable Snoop Mode 0x1 — Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** — **Description** 0x0 — Control 0x1 — Isochronous 0x2 — Bulk 0x3 — Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	`naksts`	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** — **Description** 0x0 — The core is transmitting non-NAK handshakes based on the FIFO status 0x1 — The core is transmitting NAK handshakes on this endpoint	`RO`	`0x0`
16	`dpid`	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** — **Description** 0x0 — Endpoint Data PID not active 0x1 — Endpoint Data PID active	`RO`	`0x0`

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint8

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C08
usb1	0xFFB40000	0xFFB40C08

Offset: 0xC08

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint8 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>9</td><td>bnaintr</td><td>This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BNA interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>8</td><td>outpkterr</td><td>Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No OUT Packet Error</td></tr>
<tr><td>0x1</td><td>OUT Packet Error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>6</td><td>back2backsetup</td><td>Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt,
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Back-to-Back SETUP Packets Received</td></tr>
<tr><td>0x1</td><td>Back-to-Back SETUP Packets Received</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>5</td><td>stsphsercvd</td><td>This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Status Phase Received for Control Write</td></tr>
<tr><td>0x1</td><td>Status Phase Received for Control Write</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
4	`outtknepdis`	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	`setup`	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz8

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C10
usb1	0xFFB40000	0xFFB40C10

Offset: 0xC10

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz8 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma8

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C14
usb1	0xFFB40000	0xFFB40C14

Offset: 0xC14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma8 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma8 RW 0x0															

doepdma8 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma8	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab8

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C1C
usb1	0xFFB40000	0xFFB40C1C

Offset: 0xC1C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab8 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab8 RO 0x0															

doepdmab8 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab8	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl9

Out Endpoint 9.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C20
usb1	0xFFB40000	0xFFB40C20

Offset: 0xC20

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl9 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint9

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C28
usb1	0xFFB40000	0xFFB40C28

Offset: 0xC28

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint9 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0: No interrupt 0x1: NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0: No interrupt 0x1: NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0: No interrupt 0x1: BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0: No interrupt 0x1: Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** / **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** / **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz9

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C30
usb1	0xFFB40000	0xFFB40C30

Offset: 0xC30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rxdpid RO 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

doeptsiz9 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 or 1 packet 0x2 — DATA1 or 2 packets 0x3 — MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma9

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C34
usb1	0xFFB40000	0xFFB40C34

Offset: 0xC34

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma9 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma9 RW 0x0															

doepdma9 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma9	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab9

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C3C
usb1	0xFFB40000	0xFFB40C3C

Offset: 0xC3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab9 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab9 RO 0x0															

doepdmab9 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab9	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl10

Out Endpoint 10.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C40
usb1	0xFFB40000	0xFFB40C40

Offset: 0xC40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl10 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled - Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr><tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr></table></td><td>WO</td><td>0x0</td></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr><tr><td>0x1</td><td>Enables Endpoint Data PID to DATA0)</td></tr></table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Set NAK</td></tr><tr><td>0x1</td><td>Set NAK</td></tr></table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	`naksts`	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	`dpid`	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint10

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C48
usb1	0xFFB40000	0xFFB40C48

Offset: 0xC48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint10 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** **Description** 0x0 No interrupt 0x1 NYET Interrupt	`RO`	`0x0`
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** **Description** 0x0 No interrupt 0x1 NAK Interrupt	`RO`	`0x0`
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** **Description** 0x0 No interrupt 0x1 BbleErr interrupt	`RO`	`0x0`
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	`RO`	`0x0`

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>4</td><td>outtknepdis</td><td>Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No OUT Token Received When Endpoint Disabled</td></tr>
<tr><td>0x1</td><td>OUT Token Received When Endpoint Disabled</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>setup</td><td>Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No SETUP Phase Done</td></tr>
<tr><td>0x1</td><td>SETUP Phase Done</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>ahberr</td><td>Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Interrupt</td></tr>
<tr><td>0x1</td><td>AHB Error interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>1</td><td>epdisbld</td><td>Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Interrupt</td></tr>
<tr><td>0x1</td><td>Endpoint Disabled Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz10

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C50
usb1	0xFFB40000	0xFFB40C50

Offset: 0xC50

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz10 Fields

Bit	Name	Description	Access	Reset
30:29	rxdpid	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** / **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	pktcnt	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	xfersize	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma10

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C54
usb1	0xFFB40000	0xFFB40C54

Offset: 0xC54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma10 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma10 RW 0x0															

doepdma10 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma10	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab10

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C5C
usb1	0xFFB40000	0xFFB40C5C

Offset: 0xC5C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab10 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab10 RO 0x0															

doepdmab10 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab10	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl11

Out Endpoint 11.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C60
usb1	0xFFB40000	0xFFB40C60

Offset: 0xC60

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl11 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	WO	0x0
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint11

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C68
usb1	0xFFB40000	0xFFB40C68

Offset: 0xC68

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint11 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz11

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C70
usb1	0xFFB40000	0xFFB40C70

Offset: 0xC70

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rxdpid RO 0x0		pktcnt RW 0x0										xfersize RW 0x0		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

doeptsiz11 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>30:29</td><td>rxdpid</td><td>Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>DATA0</td></tr>
<tr><td>0x1</td><td>DATA2 or 1 packet</td></tr>
<tr><td>0x2</td><td>DATA1 or 2 packets</td></tr>
<tr><td>0x3</td><td>MDATA or 3 packets</td></tr>
</table>
</td><td>RO</td><td>0x0</td></tr>
<tr><td>28:19</td><td>pktcnt</td><td>Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.</td><td>RW</td><td>0x0</td></tr>
<tr><td>18:0</td><td>xfersize</td><td>Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.</td><td>RW</td><td>0x0</td></tr>
</table>

doepdma11

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C74
usb1	0xFFB40000	0xFFB40C74

Offset: 0xC74

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma11 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma11 RW 0x0															

doepdma11 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma11	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab11

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C7C
usb1	0xFFB40000	0xFFB40C7C

Offset: 0xC7C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab11 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab11 RO 0x0															

doepdmab11 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab11	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl12

Out Endpoint 12.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C80
usb1	0xFFB40000	0xFFB40C80

Offset: 0xC80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl12 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>31</td><td>epena</td><td>Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Enable inactive</td></tr>
<tr><td>0x1</td><td>Endpoint Enable active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>30</td><td>epdis</td><td>Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Endpoint Disable</td></tr>
<tr><td>0x1</td><td>Endpoint Disable</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
29	setd1pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode. **Value** / **Description** 0x0 Disables Set DATA1 PID 0x1 Enables Set DATA1 PID	WO	0x0
28	setd0pid	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** / **Description** 0x0 Disables Set DATA0 PID 0x1 Enables Endpoint Data PID to DATA0)	WO	0x0
27	snak	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** / **Description** 0x0 No Set NAK 0x1 Set NAK	WO	0x0

Bit	Name	Description	Access	Reset
26	`cnak`	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	`WO`	`0x0`
21	`stall`	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	`RO`	`0x0`
20	`snp`	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	`RW`	`0x0`
19:18	`eptype`	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	`RW`	`0x0`

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint12

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C88
usb1	0xFFB40000	0xFFB40C88

Offset: 0xC88

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint12 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** — **Description** 0x0 — No interrupt 0x1 — NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** — **Description** 0x0 — No interrupt 0x1 — BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** — **Description** 0x0 — No interrupt 0x1 — Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz12

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C90
usb1	0xFFB40000	0xFFB40C90

Offset: 0xC90

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz12 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 or 1 packet 0x2 — DATA1 or 2 packets 0x3 — MDATA or 3 packets	RO	0x0
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma12

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C94
usb1	0xFFB40000	0xFFB40C94

Offset: `0xC94`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma12 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma12 RW 0x0															

doepdma12 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma12	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab12

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00C9C
usb1	0xFFB40000	0xFFB40C9C

Offset: 0xC9C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab12 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab12 RO 0x0															

doepdmab12 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab12	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl13

Out Endpoint 13.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CA0
usb1	0xFFB40000	0xFFB40CA0

Offset: 0xCA0

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
epena RO 0x0	epdis RO 0x0	setd1pid WO 0x0	setd0pid WO 0x0	snak WO 0x0	cnak WO 0x0	Reserved				stall RO 0x0	snp RW 0x0	eptype RW 0x0		naksts RO 0x0	dpid RO 0x0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
usbactep RW 0x0	Reserved				mps RW 0x0										

doepctl13 Fields

Bit	Name	Description	Access	Reset
31	`epena`	Applies to IN and OUT endpoints. -When Scatter/ Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/ Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	`epdis`	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Enables Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. **Value** / **Description** 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** / **Description** 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. **Value** / **Description** 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. **Value** / **Description** 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>17</td><td>naksts</td><td>When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>The core is transmitting non-NAK handshakes based on the FIFO status</td></tr>
<tr><td>0x1</td><td>The core is transmitting NAK handshakes on this endpoint</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>16</td><td>dpid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Endpoint Data PID not active</td></tr>
<tr><td>0x1</td><td>Endpoint Data PID active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint13

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CA8
usb1	0xFFB40000	0xFFB40CA8

Offset: 0xCA8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint13 Fields

Bit	Name	Description	Access	Reset
14	`nyetintrpt`	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	`nakintrpt`	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	`bbleerr`	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	`pktdrpsts`	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	bnaintr	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	outpkterr	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	back2backsetup	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	stsphsercvd	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	`outtknepdis`	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** — **Description** 0x0 — No OUT Token Received When Endpoint Disabled 0x1 — OUT Token Received When Endpoint Disabled	RO	0x0
3	`setup`	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** — **Description** 0x0 — No SETUP Phase Done 0x1 — SETUP Phase Done	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** — **Description** 0x0 — No Interrupt 0x1 — AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** — **Description** 0x0 — No Interrupt 0x1 — Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** — **Description** 0x0 — No Interrupt 0x1 — Transfer Completed Interrupt	RO	0x0

doeptsiz13

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CB0
usb1	0xFFB40000	0xFFB40CB0

Offset: 0xCB0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	rxdpid RO 0x0		pktcnt RW 0x0										xfersize RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xfersize RW 0x0															

doeptsiz13 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** — **Description** 0x0 — DATA0 0x1 — DATA2 or 1 packet 0x2 — DATA1 or 2 packets 0x3 — MDATA or 3 packets	RO	0x0
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma13

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CB4
usb1	0xFFB40000	0xFFB40CB4

Offset: `0xCB4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma13 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma13 RW 0x0															

doepdma13 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma13	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab13

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CBC
usb1	0xFFB40000	0xFFB40CBC

Offset: 0xCBC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab13 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab13 RO 0x0															

doepdmab13 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab13	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl14

Out Endpoint 14.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CC0
usb1	0xFFB40000	0xFFB40CC0

Offset: 0xCC0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO
0x0</td><td>setd1pid
WO
0x0</td><td>setd0pid
WO
0x0</td><td>snak
WO
0x0</td><td>cnak
WO
0x0</td><td colspan="4">Reserved</td><td>stall
RO
0x0</td><td>snp
RW
0x0</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO
0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

doepctl14 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>29</td><td>setd1pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/ Gather DMA mode.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA1 PID</td></tr>
<tr><td>0x1</td><td>Enables Set DATA1 PID</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>28</td><td>setd0pid</td><td>Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/ Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/ Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disables Set DATA0 PID</td></tr>
<tr><td>0x1</td><td>Enables Endpoint Data PID to DATA0)</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>27</td><td>snak</td><td>A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Set NAK</td></tr>
<tr><td>0x1</td><td>Set NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
26	cnak	A write to this bit clears the NAK bit for the endpoint. Value / Description 0x0 No Clear NAK 0x1 Clear NAK	WO	0x0
21	stall	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. Value / Description 0x0 STALL All Tokens not active 0x1 STALL All Tokens active	RO	0x0
20	snp	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory. Value / Description 0x0 Disable Snoop Mode 0x1 Enable Snoop Mode	RW	0x0
19:18	eptype	This is the transfer type supported by this logical endpoint. Value / Description 0x0 Control 0x1 Isochronous 0x2 Bulk 0x3 Interrupt	RW	0x0

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: -The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	`usbactep`	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** — **Description** 0x0 — Not Active 0x1 — USB Active Endpoint	RW	0x0
10:0	`mps`	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint14

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
`usb0`	`0xFFB00000`	`0xFFB00CC8`
`usb1`	`0xFFB40000`	`0xFFB40CC8`

Offset: `0xCC8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint14 Fields

Bit	Name	Description	Access	Reset
14	nyetintrpt	The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint. **Value** / **Description** 0x0 No interrupt 0x1 NYET Interrupt	RO	0x0
13	nakintrpt	The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo. **Value** / **Description** 0x0 No interrupt 0x1 NAK Interrupt	RO	0x0
12	bbleerr	The core generates this interrupt when babble is received for the endpoint. **Value** / **Description** 0x0 No interrupt 0x1 BbleErr interrupt	RO	0x0
11	pktdrpsts	This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. **Value** / **Description** 0x0 No interrupt 0x1 Packet Drop Status interrupt	RO	0x0

Bit	Name	Description	Access	Reset
9	`bnaintr`	This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done **Value** / **Description** 0x0 No interrupt 0x1 BNA interrupt	RO	0x0
8	`outpkterr`	Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet. **Value** / **Description** 0x0 No OUT Packet Error 0x1 OUT Packet Error	RO	0x0
6	`back2backsetup`	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt, **Value** / **Description** 0x0 No Back-to-Back SETUP Packets Received 0x1 Back-to-Back SETUP Packets Received	RO	0x0
5	`stsphsercvd`	This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode. **Value** / **Description** 0x0 No Status Phase Received for Control Write 0x1 Status Phase Received for Control Write	RO	0x0

Bit	Name	Description	Access	Reset
4	outtknepdis	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** / **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	setup	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** / **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	ahberr	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	epdisbld	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz14

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CD0
usb1	0xFFB40000	0xFFB40CD0

Offset: 0xCD0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz14 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** / **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma14

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CD4
usb1	0xFFB40000	0xFFB40CD4

Offset: `0xCD4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma14 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma14 RW 0x0															

doepdma14 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma14	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab14

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CDC
usb1	0xFFB40000	0xFFB40CDC

Offset: 0xCDC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab14 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab14 RO 0x0															

doepdmab14 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab14	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

doepctl15

Out Endpoint 15.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CE0
usb1	0xFFB40000	0xFFB40CE0

Offset: 0xCE0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>epena
RO 0x0</td><td>epdis
RO 0x0</td><td>setd1pid
WO 0x0</td><td>setd0pid
WO 0x0</td><td>snak
WO 0x0</td><td>cnak
WO 0x0</td><td colspan="4">Reserved</td><td>stall
RO 0x0</td><td>snp
RW 0x0</td><td colspan="2">eptype
RW 0x0</td><td>naksts
RO 0x0</td><td>dpid
RO 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>usbactep
RW 0x0</td><td colspan="4">Reserved</td><td colspan="11">mps
RW 0x0</td></tr>
</table>

doepctl15 Fields

Bit	Name	Description	Access	Reset
31	epena	Applies to IN and OUT endpoints. -When Scatter/Gather DMA mode is enabled, -for IN endpoints this bit indicates that the descriptor structure and data buffer with data ready to transmit is setup. -for OUT endpoint it indicates that the descriptor structure and data buffer to receive data is setup. -When Scatter/Gather DMA mode is enabled such as for buffer-pointer based DMA mode: - for IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. - for OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. - The core clears this bit before setting any of the following interrupts on this endpoint: -SETUP Phase Done -Endpoint Disabled -Transfer Completed for control endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory. **Value** / **Description** 0x0 Endpoint Enable inactive 0x1 Endpoint Enable active	RO	0x0
30	epdis	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint. **Value** / **Description** 0x0 No Endpoint Disable 0x1 Endpoint Disable	RO	0x0

Bit	Name	Description	Access	Reset
29	`setd1pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Set Odd (micro) frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.This field is not applicable for Scatter/Gather DMA mode. **Value** — **Description** 0x0 — Disables Set DATA1 PID 0x1 — Enables Set DATA1 PID	`WO`	`0x0`
28	`setd0pid`	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. In non-Scatter/Gather DMA mode: Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame. When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is in the transmit descriptor structure. The frame in which to receive data is updated in receive descriptor structure. **Value** — **Description** 0x0 — Disables Set DATA0 PID 0x1 — Enables Endpoint Data PID to DATA0)	`WO`	`0x0`
27	`snak`	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also Set this bit for an endpoint after a SETUP packet is received on that endpoint. **Value** — **Description** 0x0 — No Set NAK 0x1 — Set NAK	`WO`	`0x0`

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>26</td><td><code>cnak</code></td><td>A write to this bit clears the NAK bit for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No Clear NAK</td></tr>
<tr><td>0x1</td><td>Clear NAK</td></tr>
</table></td><td>WO</td><td>0x0</td></tr>
<tr><td>21</td><td><code>stall</code></td><td>Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>STALL All Tokens not active</td></tr>
<tr><td>0x1</td><td>STALL All Tokens active</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>20</td><td><code>snp</code></td><td>Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disable Snoop Mode</td></tr>
<tr><td>0x1</td><td>Enable Snoop Mode</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>19:18</td><td><code>eptype</code></td><td>This is the transfer type supported by this logical endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Control</td></tr>
<tr><td>0x1</td><td>Isochronous</td></tr>
<tr><td>0x2</td><td>Bulk</td></tr>
<tr><td>0x3</td><td>Interrupt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
17	naksts	When either the application or the core sets this bit: - The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. -for non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. -for isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake. **Value** **Description** 0x0 The core is transmitting non-NAK handshakes based on the FIFO status 0x1 The core is transmitting NAK handshakes on this endpoint	RO	0x0
16	dpid	Applies to interrupt/bulk IN and OUT endpoints only. Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID. 0: DATA0 1: DATA1This field is applicable both for Scatter/Gather DMA mode and non-Scatter/Gather DMA mode. Even/Odd (Micro)Frame (EO_FrNum) In non-Scatter/Gather DMA mode: Applies to isochronous IN and OUT endpoints only. Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register. 0: Even (micro)frame 1: Odd (micro) frame When Scatter/Gather DMA mode is enabled, this field is reserved. The frame number in which to send data is provided in the transmit descriptor structure. The frame in which data is received is updated in receive descriptor structure. **Value** **Description** 0x0 Endpoint Data PID not active 0x1 Endpoint Data PID active	RO	0x0

Bit	Name	Description	Access	Reset
15	usbactep	Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit. **Value** **Description** 0x0 Not Active 0x1 USB Active Endpoint	RW	0x0
10:0	mps	Applies to IN and OUT endpoints. The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.	RW	0x0

doepint15

This register indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (GINTSTS.OEPInt or GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CE8
usb1	0xFFB40000	0xFFB40CE8

Offset: 0xCE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	nyetintrpt RO 0x0	nakintrpt RO 0x0	bbleerr RO 0x0	pktdrpsts RO 0x0	Reserved	bnaintr RO 0x0	outpkterr RO 0x0	Reserved	back2backsetup RO 0x0	stsphsercvd RO 0x0	outtknepdis RO 0x0	setup RO 0x0	ahberr RO 0x0	epdisbld RO 0x0	xfercompl RO 0x0

doepint15 Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>14</td><td>nyetintrpt</td><td>The core generates this interrupt when a NYET response is transmitted for a non isochronous OUT endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NYET Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>13</td><td>nakintrpt</td><td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to un-availability of data in the TXFifo.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>NAK Interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>12</td><td>bbleerr</td><td>The core generates this interrupt when babble is received for the endpoint.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>BbleErr interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>11</td><td>pktdrpsts</td><td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>No interrupt</td></tr>
<tr><td>0x1</td><td>Packet Drop Status interrupt</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>9</td><td><code>bnaintr</code></td><td>This bit is valid only when Scatter/Gather DMA mode is This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as Host busy or DMA done<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No interrupt</td></tr><tr><td>0x1</td><td>BNA interrupt</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>8</td><td><code>outpkterr</code></td><td>Applies to OUT endpoints Only This interrupt is asserted when the core detects an overflow or a CRC error for non-Isochronous OUT packet.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No OUT Packet Error</td></tr><tr><td>0x1</td><td>OUT Packet Error</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>6</td><td><code>back2backsetup</code></td><td>Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. for information about handling this interrupt,<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Back-to-Back SETUP Packets Received</td></tr><tr><td>0x1</td><td>Back-to-Back SETUP Packets Received</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>5</td><td><code>stsphsercvd</code></td><td>This interrupt is valid only for Control OUT endpoints and only in Scatter Gather DMA mode. This interrupt is generated only after the core has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a Control Write transfer. The application can use this interrupt to ACK or STALL the Status phase, after it has decoded the data phase. This is applicable only in Case of Scatter Gather DMA mode.<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>No Status Phase Received for Control Write</td></tr><tr><td>0x1</td><td>Status Phase Received for Control Write</td></tr></table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
4	`outtknepdis`	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received. **Value** / **Description** 0x0 No OUT Token Received When Endpoint Disabled 0x1 OUT Token Received When Endpoint Disabled	RO	0x0
3	`setup`	Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet. **Value** / **Description** 0x0 No SETUP Phase Done 0x1 SETUP Phase Done	RO	0x0
2	`ahberr`	Applies to IN and OUT endpoints.This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address. **Value** / **Description** 0x0 No Interrupt 0x1 AHB Error interrupt	RO	0x0
1	`epdisbld`	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request. **Value** / **Description** 0x0 No Interrupt 0x1 Endpoint Disabled Interrupt	RO	0x0

Bit	Name	Description	Access	Reset
0	xfercompl	Applies to IN and OUT endpoints.When Scatter/ Gather DMA mode is enabled This field indicates that the requested data from the internal FIFO is moved to external system memory. This interrupt is generated only when the corresponding endpoint descriptor is closed, and the IOC bit for the corresponding descriptor is Set. When Scatter/Gather DMA mode is disabled, this field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint. **Value** **Description** 0x0 No Interrupt 0x1 Transfer Completed Interrupt	RO	0x0

doeptsiz15

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (DIEPCTLn.EPEna/ DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CF0
usb1	0xFFB40000	0xFFB40CF0

Offset: 0xCF0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>Reserved</td><td colspan="2">rxdpid
RO 0x0</td><td colspan="10">pktcnt
RW 0x0</td><td colspan="3">xfersize
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">xfersize
RW 0x0</td></tr>
</table>

doeptsiz15 Fields

Bit	Name	Description	Access	Reset
30:29	`rxdpid`	Applies to isochronous OUT endpoints only.This is the data PID received in the last packet for this endpoint. Use datax. Applies to control OUT Endpoints only. Use packetx. This field specifies the number of back-to-back SETUP data packets the endpoint can receive. **Value** / **Description** 0x0 DATA0 0x1 DATA2 or 1 packet 0x2 DATA1 or 2 packets 0x3 MDATA or 3 packets	RO	0x0
28:19	`pktcnt`	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.This field is decremented every time a packet (maximum size or short packet) is read from the RxFIFO.	RW	0x0
18:0	`xfersize`	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be Set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the RxFIFO.	RW	0x0

doepdma15

DMA OUT Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CF4
usb1	0xFFB40000	0xFFB40CF4

Offset: `0xCF4`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdma15 RW 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdma15 RW 0x0															

doepdma15 Fields

Bit	Name	Description	Access	Reset
31:0	doepdma15	Holds the start address of the external memory for storing or fetching endpoint data. for control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address. When Scatter/Gather DMA mode is not enabled, the application programs the start address value in this field. When Scatter/Gather DMA mode is enabled, this field indicates the base pointer for the descriptor list.	RW	0x0

doepdmab15

DMA Buffer Address.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00CFC
usb1	0xFFB40000	0xFFB40CFC

Offset: 0xCFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
doepdmab15 RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
doepdmab15 RO 0x0															

doepdmab15 Fields

Bit	Name	Description	Access	Reset
31:0	doepdmab15	Holds the current buffer address. This register is updated as and when the data transfer for the corresponding end point is in progress. This register is present only in Scatter/Gather DMA mode.	RO	0x0

Power and Clock Gating Register Register Descriptions

There is a single register for power and clock gating. It is available in both Host and Device modes.

Offset: 0xe00

pcgcctl on page 18-796
This register is available in Host and Device modes. The application can use this register to control the core's power-down and clock gating features. Because the CSR module is turned off during power-down, this register is implemented in the AHB Slave BIU module.

pcgcctl

This register is available in Host and Device modes. The application can use this register to control the core's power-down and clock gating features. Because the CSR module is turned off during power-down, this register is implemented in the AHB Slave BIU module.

Module Instance	Base Address	Register Address
usb0	0xFFB00000	0xFFB00E00
usb1	0xFFB40000	0xFFB40E00

Offset: 0xE00

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								l1suspended RO 0x0	physleep RO 0x0	Reserved		rstpdwnmodule RW 0x0	Reserved		stoppclk RW 0x0

pcgcctl Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>7</td><td>l1suspended</td><td>Indicates that the PHY is in deep sleep when in L1 state.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Non Deep Sleep</td></tr><tr><td>0x1</td><td>Deep Sleep active</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>6</td><td>physleep</td><td>Indicates that the PHY is in Sleep State.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Phy non-sleep</td></tr><tr><td>0x1</td><td>Phy sleep state</td></tr></table></td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>rstpdwnmodule</td><td>This bit is valid only in Partial Power-Down mode. Theapplication sets this bit when the power is turned off. The application clears this bit after the power is turned on and the PHY clock is up. The R/W of all core registers are possible only when this bit is set to 1b0.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Power is turned on</td></tr><tr><td>0x1</td><td>Power is turned off</td></tr></table></td><td>RW</td><td>0x0</td></tr>
<tr><td>0</td><td>stoppclk</td><td>The application sets this bit to stop the PHY clock (phy_clk) when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.
<table><tr><th>Value</th><th>Description</th></tr><tr><td>0x0</td><td>Disable Stop Pclk</td></tr><tr><td>0x1</td><td>Enable Stop Pclk</td></tr></table></td><td>RW</td><td>0x0</td></tr>
</table>

USB Data FIFO Address Map

These regions, available in both Host and Device modes, are a push/pop FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Module Instance	Base Address
usb0	0xFFB00000
usb1	0xFFB40000

Table 18-4: USB Endpoint FIFO Address Ranges

Name	Description	Start Address Offset	End Address Offset
EP0/HC0 FIFO	This address space is allocated for Endpoint 0/Host Channel 0 push/pop FIFO access.	0x1000	0x1FFF
EP1/HC1 FIFO	This address space is allocated for Endpoint 1/Host Channel 1 push/pop FIFO access.	0x2000	0x2FFF
EP2/HC2 FIFO	This address space is allocated for Endpoint 2/Host Channel 2 push/pop FIFO access.	0x3000	0x3FFF
EP3/HC3 FIFO	This address space is allocated for Endpoint 3/Host Channel 3 push/pop FIFO access.	0x4000	0x4FFF
EP4/HC4 FIFO	This address space is allocated for Endpoint 4/Host Channel 4 push/pop FIFO access.	0x5000	0x5FFF
EP5/HC5 FIFO	This address space is allocated for Endpoint 5/Host Channel 5 push/pop FIFO access.	0x6000	0x6FFF
EP6/HC6 FIFO	This address space is allocated for Endpoint 6/Host Channel 6 push/pop FIFO access.	0x7000	0x7FFF
EP7/HC7 FIFO	This address space is allocated for Endpoint 7/Host Channel 7 push/pop FIFO access.	0x8000	0x8FFF

Name	Description	Start Address Offset	End Address Offset
EP8/HC8 FIFO	This address space is allocated for Endpoint 8/Host Channel 8 push/pop FIFO access.	0x9000	0x9FFF
EP9/HC9 FIFO	This address space is allocated for Endpoint 9/Host Channel 9 push/pop FIFO access.	0xA000	0xAFFF
EP10/HC10 FIFO	This address space is allocated for Endpoint 10/Host Channel 10 push/pop FIFO access.	0xB000	0xBFFF
EP11/HC11 FIFO	This address space is allocated for Endpoint 11/Host Channel 11 push/pop FIFO access.	0xC000	0xCFFF
EP12/HC12 FIFO	This address space is allocated for Endpoint 12/Host Channel 12 push/pop FIFO access.	0xD000	0xDFFF
EP13/HC13 FIFO	This address space is allocated for Endpoint 13/Host Channel 13 push/pop FIFO access.	0xE000	0xEFFF
EP14/HC14 FIFO	This address space is allocated for Endpoint 14/Host Channel 14 push/pop FIFO access.	0xF000	0xFFFF
EP15/HC15 FIFO	This address space is allocated for Endpoint 15/Host Channel 15 push/pop FIFO access.	0x10000	0x10FFF

USB Direct Access FIFO RAM Address Map

This address space provides direct access to the Data FIFO RAM for debugging.

Module Instance	Base Address
`usb0`	`0xFFB00000`
`usb1`	`0xFFB40000`

Table 18-5: USB Direct Access FIFO Address Range

Name	Description	Start Address Offset	End Address Offset
`Direct_FIFO`	This address space is allocated for directly accessing the data FIFO for debugging purposes.	`0x20000`	`0x3FFFF`

Document Revision History

Table 18-6: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added USB OTG Controller address map and register definitions.
February 2014	2014.02.28	Maintenance release.
December 2013	2013.12.30	Maintenance release.
November 2012	1.2	• Described interrupt generation. • Described software initialization in host and device modes. • Described software operation in host and device modes. • Simplified features list. • Simplified hardware description.
June 2012	1.1	Added information about ECCs.
January 2012	1.0	Initial release.

SPI Controller 19

2014.06.30

cv_54019 Subscribe Send Feedback

The hard processor system (HPS) provides two serial peripheral interface (SPI) masters and two SPI slaves. The SPI masters and slaves are instances of the Synopsys® DesignWare® Synchronous Serial Interface (SSI) controller (DW_apb_ssi). †[42]

Features of the SPI Controller

The SPI controller has the following features: †

- Serial master and serial slave controllers – Enable serial communication with serial-master or serial-slave peripheral devices. †
- Serial interface operation – Programmable choice of the following protocols:
  - Motorola SPI protocol
  - Texas Instruments Synchronous Serial Protocol
  - National Semiconductor Microwire
- DMA controller interface integrated with HPS DMA controller
- SPI master supports `rxd` sample delay
- Transmit and receive FIFO buffers are 256 words deep
- SPI master supports up to four slave selects
- Programmable master serial bit rate
- Programmable data item size of 4 to 16 bits

SPI Block Diagram and System Integration

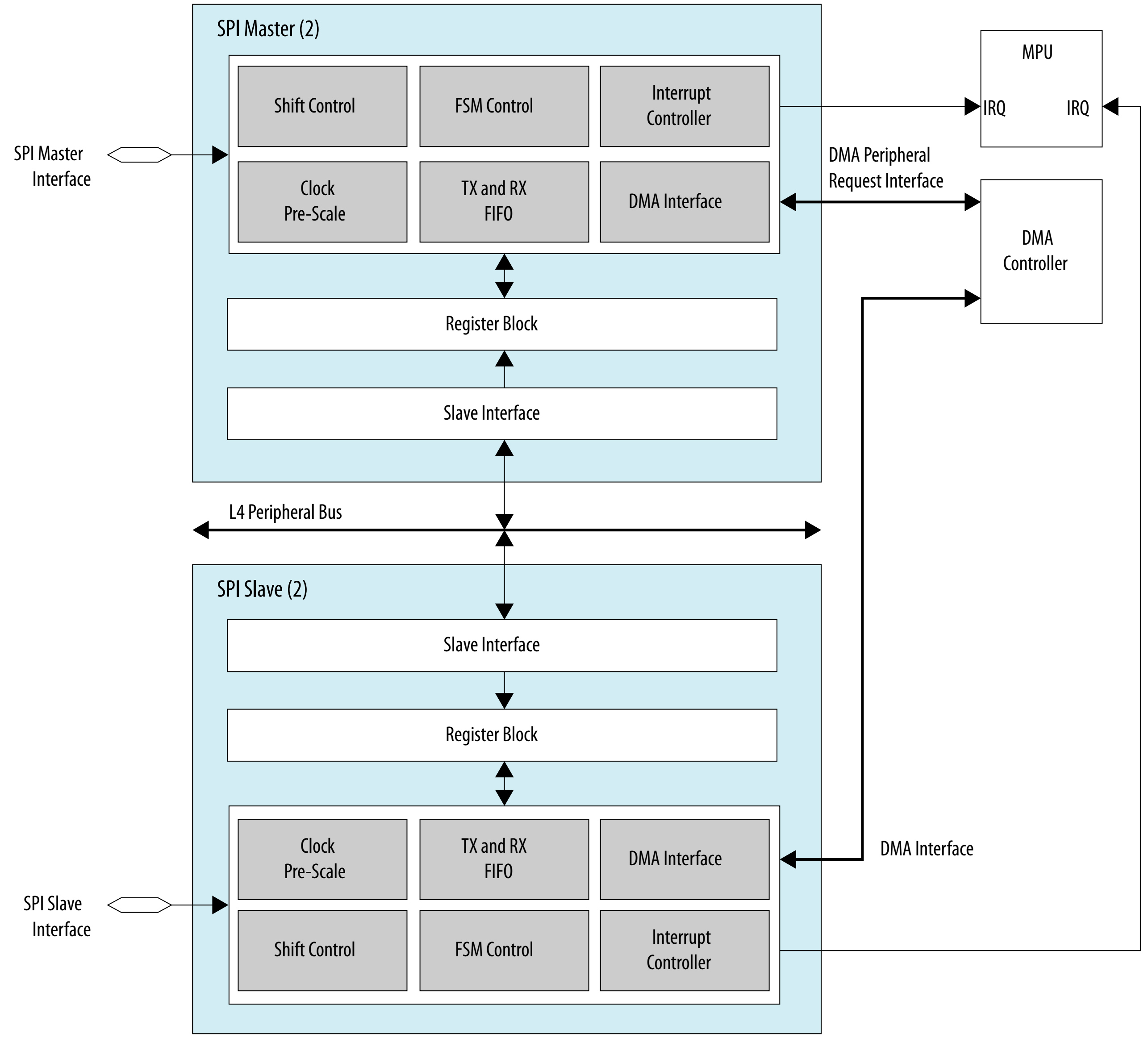
The SPI supports data bus widths of 32 bits. †

[42] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

SPI Block Diagram

Figure 19-1: SPI Block Diagram



The functional groupings of the main interfaces to the SPI block are as follows: †

- System bus interface
- DMA peripheral request interface
- Interrupt interface
- SPI interface

Functional Description of the SPI Controller

Protocol Details and Standards Compliance

This chapter describes the functional operation of the SPI controller. †

The host processor accesses data, control, and status information about the SPI controller through the system bus interface. The SPI also interfaces with the DMA Controller. †

The HPS includes two general-purpose SPI master controllers and two general-purpose SPI slave controllers.

The SPI controller can connect to any other SPI device using any of the following protocols:

- Motorola SPI Protocol †
- Texas Instruments Serial Protocol (SSP) †
- National Semiconductor Microwire Protocol †

SPI Controller Overview

In order for the SPI controller to connect to a serial-master or serial-slave peripheral device, the peripheral must have a least one of the following interfaces: †

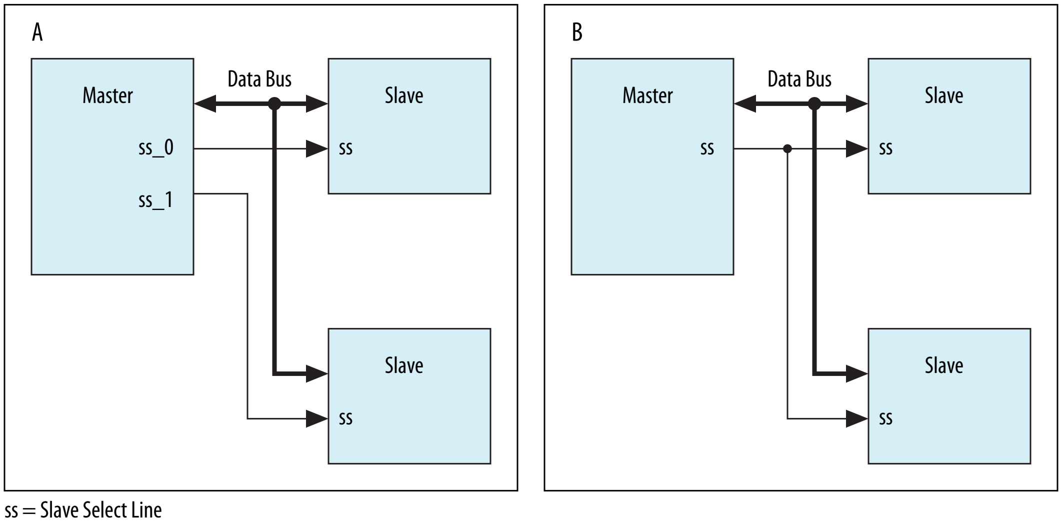
- Motorola SPI protocol – A four-wire, full-duplex serial protocol from Motorola. The slave select line is held high when the SPI controller is idle or disabled. For more information, refer to “Motorola SPI Protocol”. †
- Texas Instruments Serial Protocol (SSP) – A four-wire, full-duplex serial protocol. The slave select line used for SPI and Microwire protocols doubles as the frame indicator for the SSP protocol. For more information, refer to “Texas Instruments Synchronous Serial Protocol (SSP)”. †
- National Semiconductor Microwire – A half-duplex serial protocol, which uses a control word transmitted from the serial master to the target serial slave. For more information, refer to “National Semiconductor Microwire Protocol”. You can program the FRF (frame format) bit field in the Control Register 0 (`CTRLR0`) to select which protocol is used. †

The serial protocols supported by the SPI controller allow for serial slaves to be selected or addressed using hardware. Serial slaves are selected under the control of dedicated hardware select lines. The number of select lines generated from the serial master is equal to the number of serial slaves present on the bus. The serial-master device asserts the select line of the target serial slave before data transfer begins. This architecture is illustrated in part A in the following figure. †

When implemented in software, the input select line for all serial slave devices should originate from a single slave select output on the serial master. In this mode it is assumed that the serial master has only a single slave select output. If there are multiple serial masters in the system, the slave select output from all masters can be logically ANDed to generate a single slave select input for all serial slave devices. †

The main program in the software domain controls selection of the target slave device; this architecture is illustrated in part B of the Hardware/Software Slave Selection diagram Software would control which slave is to respond to the serial transfer request from the master device. †

Figure 19-2: Hardware/Software Slave Selection



Related Information

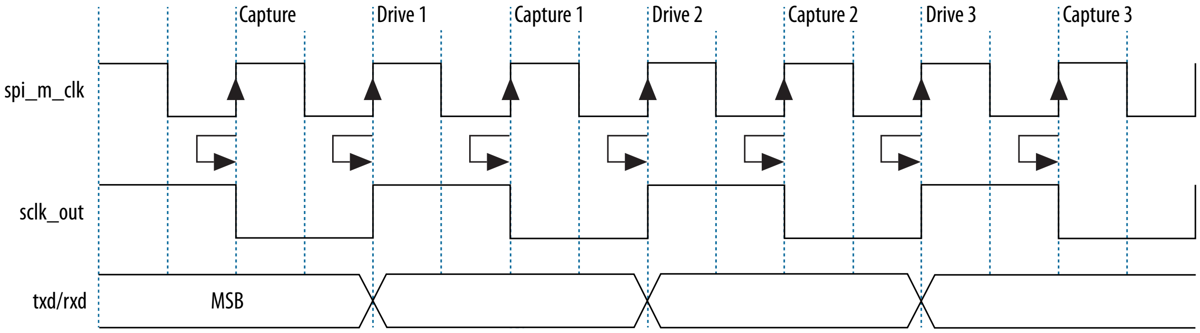
- **Motorola SPI Protocol** on page 19-14
- **Texas Instruments Synchronous Serial Protocol (SSP)** on page 19-16
- **National Semiconductor Microwire Protocol** on page 19-17

Serial Bit-Rate Clocks

SPI Master Bit-Rate Clock

The maximum frequency of the SPI master bit-rate clock (`sclk_out`) is one-half the frequency of SPI master clock (`spi_m_clk`). This allows the shift control logic to capture data on one clock edge of `sclk_out` and propagate data on the opposite edge. The `sclk_out` line toggles only when an active transfer is in progress. At all other times it is held in an inactive state, as defined by the serial protocol under which it operates. †

Figure 19-3: Maximum sclk_out/spi_m_clk Ratio



The frequency of `sclk_out` can be derived from the equation below, where <*SPI clock*> is `spi_m_clk` for the master SPI modules and `l4_main_clk` for the slave SPI modules. †

$F_{sclk\_out} = F_{<SPI\ clock>} / \mathrm{SCKDV}$

`SCKDV` is a bit field in the register `BAUDR`, holding any even value in the range 2 to 65,534. If `SCKDV` is 0, then `sclk_out` is disabled. †

The following equation describes the frequency ratio restrictions between the bit-rate clock `sclk_out` and the SPI master peripheral clock. The SPI master peripheral clock must be at least double the offchip master clock. †

Table 19-1: SPI Master Peripheral Clock

• SPI Master Peripheral Clock
$F_{spi\_m\_clk}$ >= 2 x (maximum $_{Fsclk\_out}$) †

SPI Slave Bit-Rate Clock

The minimum frequency of `l4_main_clk` depends on the operation of the slave peripheral. If the slave device is *receive only*, the minimum frequency of `l4_main_clk` is six times the maximum expected frequency of the bit-rate clock from the master device (`sclk_in`). The `sclk_in` signal is double synchronized to the `l4_main_clk` domain, and then it is edge detected; this synchronization requires three `l4_main_clk` periods. †

If the slave device is *transmit and receive*, the minimum frequency of `l4_main_clk` is eight times the maximum expected frequency of the bit-rate clock from the master device (`sclk_in`). This ensures that data on the master `rxd` line is stable before the master shift control logic captures the data. †

The frequency ratio restrictions between the bit-rate clock `sclk_in` and the SPI slave peripheral clock are as follows: †

- Slave (receive only): $F_{l4\_main\_clk}$ >= 6 x (maximum $_{Fsclk\_in}$) †
- Slave: Fl4_main_clk >= 8 x (maximum Fsclk_in) †

Transmit and Receive FIFO Buffers

There are two 16-bit FIFO buffers, a transmit FIFO buffer and a receive FIFO buffer, with a depth of 256. Data frames that are less than 16 bits in size must be right-justified when written into the transmit FIFO buffer. The shift control logic automatically right-justifies receive data in the receive FIFO buffer. †

Each data entry in the FIFO buffers contains a single data frame. It is impossible to store multiple data frames in a single FIFO buffer location; for example, you may not store two 8-bit data frames in a single FIFO buffer location. If an 8-bit data frame is required, the upper 8-bits of the FIFO buffer entry are ignored or unused when the serial shifter transmits the data. †

The transmit and receive FIFO buffers are cleared when the PI controller is disabled (`SSIENR=0`) or reset.

The transmit FIFO buffer is loaded by write commands to the SPI data register (`DR`). Data are popped (removed) from the transmit FIFO buffer by the shift control logic into the transmit shift register. The transmit FIFO buffer generates a transmit FIFO empty interrupt request when the number of entries in the FIFO buffer is less than or equal to the FIFO buffer threshold value. The threshold value, set through the register `TXFTLR`, determines the level of FIFO buffer entries at which an interrupt is generated. The threshold value allows you to provide early indication to the processor that the transmit FIFO buffer is nearly empty. A Transmit FIFO Overflow Interrupt is generated if you attempt to write data into an already full transmit FIFO buffer. †

Data are popped from the receive FIFO buffer by read commands to the SPI data register (`DR`). The receive FIFO buffer is loaded from the receive shift register by the shift control logic. The receive FIFO buffer generates a receive FIFO full interrupt request when the number of entries in the FIFO buffer is greater than or equal to the FIFO buffer threshold value plus one. The threshold value, set through register `RXFTLR`, determines the level of FIFO buffer entries at which an interrupt is generated. †

The threshold value allows you to provide early indication to the processor that the receive FIFO buffer is nearly full. A Receive FIFO Overrun Interrupt is generated when the receive shift logic attempts to load data into a completely full receive FIFO buffer. However, the newly received data are lost. A Receive FIFO Underflow Interrupt is generated if you attempt to read from an empty receive FIFO buffer. This alerts the processor that the read data are invalid. †

Related Information

Reset Manager on page 3-1
For more information, refer to the*Reset Manager* chapter.

SPI Interrupts

The SPI controller supports combined interrupt requests, which can be masked. The combined interrupt request is the ORed result of all other SPI interrupts after masking. All SPI interrupts have active-high polarity level. The SPI interrupts are described as follows: †

- Transmit FIFO Empty Interrupt – Set when the transmit FIFO buffer is equal to or below its threshold value and requires service to prevent an underrun. The threshold value, set through a software-programmable register, determines the level of transmit FIFO buffer entries at which an interrupt is generated. This interrupt is cleared by hardware when data are written into the transmit FIFO buffer, bringing it over the threshold level. †
- Transmit FIFO Overflow Interrupt – Set when a master attempts to write data into the transmit FIFO buffer after it has been completely filled. When set, new data writes are discarded. This interrupt remains set until you read the transmit FIFO overflow interrupt clear register (`TXOICR`). †
- Receive FIFO Full Interrupt – Set when the receive FIFO buffer is equal to or above its threshold value plus 1 and requires service to prevent an overflow. The threshold value, set through a software-programmable register, determines the level of receive FIFO buffer entries at which an interrupt is generated. This interrupt is cleared by hardware when data are read from the receive FIFO buffer, bringing it below the threshold level. †
- Receive FIFO Overflow Interrupt – Set when the receive logic attempts to place data into the receive FIFO buffer after it has been completely filled. When set, newly received data are discarded. This interrupt remains set until you read the receive FIFO overflow interrupt clear register (`RXOICR`). †
- Receive FIFO Underflow Interrupt – Set when a system bus access attempts to read from the receive FIFO buffer when it is empty. When set, zeros are read back from the receive FIFO buffer. This interrupt remains set until you read the receive FIFO underflow interrupt clear register (`RXUICR`). †
- Combined Interrupt Request – ORed result of all the above interrupt requests after masking. To mask this interrupt signal, you must mask all other SPI interrupt requests. †

Transmit FIFO Overflow, Transmit FIFO Empty, Receive FIFO Full, Receive FIFO Underflow, and Receive FIFO Overflow interrupts can all be masked independently, using the Interrupt Mask Register (`IMR`). †

Interface Pins

Table 19-2: Interface Pins

Signal Name	Signal Width	Direction	Description
	SPI Master		

Signal Name	Signal Width	Direction	Description
CLK	1	Out	Serial clock output from the SPI master
MOSI	1	Out	Transmit data line for the SPI master
MISO	1	In	Receive data line for the SPI master
SS0	1	Out	Slave Select 0 Slave select signal from SPI master
SS1	1	Out	Slave Select 1 Slave select signal from SPI master
	SPI Slave		
CLK	1	In	Serial clock input to the SPI slave
MOSI	1	Out	Receive data line for the SPI slave
MISO	1	In	Transmit data line for the SPI slave
SS0	1	In	Slave select input to the SPI slave

FPGA Routing

Table 19-3: SPI Master Signals for FPGA routing:

Signal Name	Signal Width	Direction	Description
spim_txd	1	Out	Transmit data line for the SPI master
spim_rxd	1	In	Receive data line for the SPI master
spim_ss_in_n	1	In	Master Contention Input

Signal Name	Signal Width	Direction	Description
spim_ssi_oe_n	1	Out	Output enable for the SPI master
spim_ss_0_n	1	Out	Slave Select 0 Slave select signal from SPI master
spim_ss_1_n	1	Out	Slave Select 1 Allows second slave to be connected to this master
spim_ss_2_n	1	Out	Slave Select 2 Allows third slave to be connected to this master
spim_ss_3_n	1	Out	Slave Select 3 Allows fourth slave to be connected to this master

Table 19-4: SPI Slave Signals for FPGA Routing

spis_txd	1	Out	Transmit data line for the SPI slave
spis_rxd	1	In	Receive data line for the SPI slave
spis_ss_in_n	1	Out	Master Contention Input
spis_ssi_oe_n	1	Out	Output enable for the SPI slave
spis_sclk_in	1	In	Serial clock input

Transfer Modes

When transferring data on the serial bus, the SPI controller operates one of several modes. The transfer mode (TMOD) is set by writing to control register 0 (`CTRLR0`).

Note: The transfer mode setting does not affect the duplex of the serial transfer. TMOD is ignored for Microwire transfers, which are controlled by the MWCR register. †

Transmit and Receive

When TMOD = 0, both transmit and receive logic are valid. The data transfer occurs as normal according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO buffer and sent through the `txd` line to the target device, which replies with data on the `rxd` line. The receive data from the target device is moved from the receive shift register into the receive FIFO buffer at the end of each data frame. †

Transmit Only

When TMOD = 1, any receive data are ignored. The data transfer occurs as normal, according to the selected frame format (serial protocol). Transmit data are popped from the transmit FIFO buffer and sent through the `txd` line to the target device, which replies with data on the `rxd` line. At the end of the data frame, the receive shift register does not load its newly received data into the receive FIFO buffer. The data in the receive shift register is overwritten by the next transfer. You should mask interrupts originating from the receive logic when this mode is entered. †

Receive Only

When TMOD = 2, the transmit data are invalid. In the case of the SPI slave, the transmit FIFO buffer is never popped in Receive Only mode. The `txd` output remains at a constant logic level during the transmission. The data transfer occurs as normal according to the selected frame format (serial protocol). The receive data from the target device is moved from the receive shift register into the receive FIFO buffer at the end of each data frame. You should mask interrupts originating from the transmit logic when this mode is entered. †

EEPROM Read

Note: This transfer mode is only valid for serial masters. †

When TMOD = 3, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. This takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). During the transmission of the opcode and address, no data is captured by the receive logic (as long as the SPI master is transmitting data on its `txd` line, data on the `rxd` line is ignored). The SPI master continues to transmit data until the transmit FIFO buffer is empty. You should ONLY have enough data frames in the transmit FIFO buffer to supply the opcode and address to the EEPROM. If more data frames are in the transmit FIFO buffer than are needed, then Read data is lost. †

When the transmit FIFO buffer becomes empty (all control information has been sent), data on the receive line (`rxd`) is valid and is stored in the receive FIFO buffer; the `txd` output is held at a constant logic level. The serial transfer continues until the number of data frames received by the SPI master matches the value of the NDF field in the `CTRLR1` register plus one. †

Note: EEPROM read mode is not supported when the SPI controller is configured to be in the SSP mode. †

SPI Master

The SPI master initiates and controls all serial transfers with serial-slave peripheral devices. †

The serial bit-rate clock, generated and controlled by the SPI controller, is driven out on the `sclk_out` line. When the SPI controller is disabled, no serial transfers can occur and `sclk_out` is held in “inactive” state, as defined by the serial protocol under which it operates. †

Related Information

SPI Block Diagram on page 19-2

RXD Sample Delay

The SPI master device is capable of delaying the default sample time of the `rxd` signal in order to increase the maximum achievable frequency on the serial bus.

Round trip routing delays on the sclk_out signal from the master and the rxd signal from the slave can mean that the timing of the rxd signal, as seen by the master, has moved away from the normal sampling time.

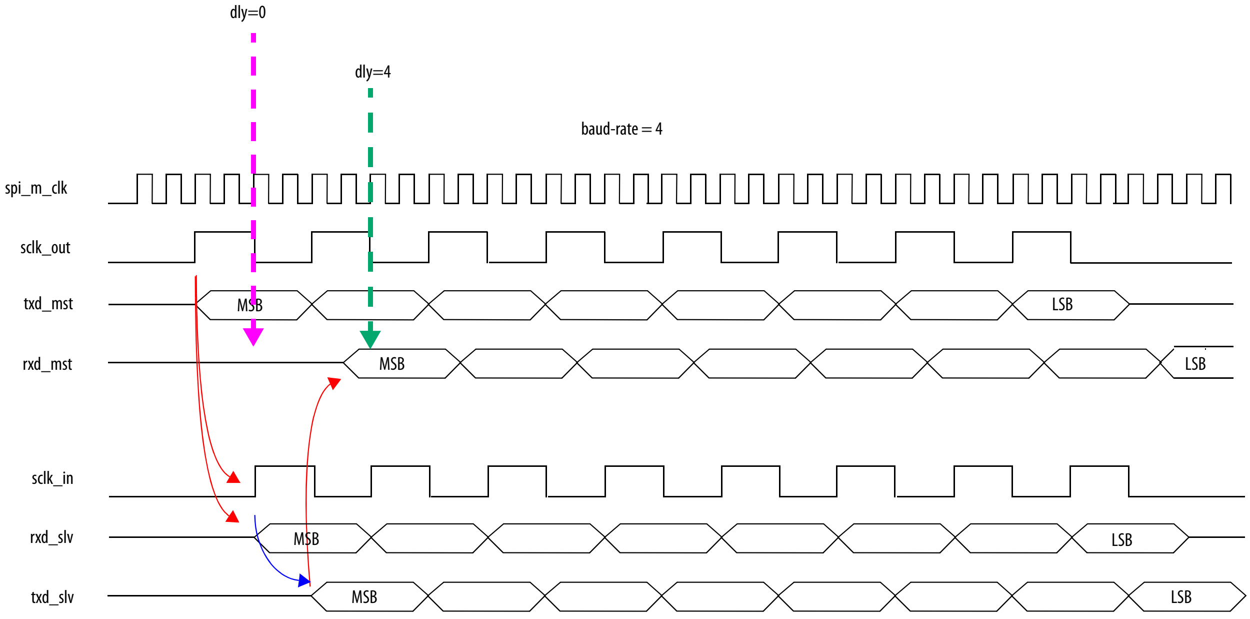
Without the RXD sample delay, you must increase the baud rate for the transfer in order to ensure that the setup times on the rxd signal are within range. This reduces the frequency of the serial interface.

Additional logic is included in the SPI master to delay the default sample time of the rxd signal. This additional logic can help to increase the maximum achievable frequency on the serial bus. †

By writing to the rsd field of the RX Sample Delay Register (rx_sample_dly), you specify an additional amount of delay applied to the rxd sample, in number of l4_main_clk clock cycles, up to 64 cycles. If the rsd field is programmed with a value exceeding 64, zero delay is applied to the rxd sample.

Round trip routing delays on the sclk_out signal from the master and the rxd signal from the slave can mean that the timing of the rxd signal, as seen by the master, has moved away from the normal sampling time. Red arrows indicate routing delay between master and slave devices. Blue arrow indicates sampling delay within slave from receiving sclk_in to driving txd out. †

Figure 19-4: Effects of Round Trip Routing Delays on sclk_out Signal



Red arrows indicate routing dela   y between master and slave devices
Blue arrow indicates sampling delay within slav   e from receiving slk_in to driving txd out

Data Transfers

The SPI master starts data transfers when all the following conditions are met:

- The SPI master is enabled
- There is at least one valid entry in the transmit FIFO buffer
- A slave device is selected

When actively transferring data, the busy flag (BUSY) in the status register (SR) is set. You must wait until the busy flag is cleared before attempting a new serial transfer. †

Note: The BUSY status is not set when the data are written into the transmit FIFO buffer. This bit gets set only when the target slave has been selected and the transfer is underway. After writing data into the transmit FIFO buffer, the shift logic does not begin the serial transfer until a positive edge of the sclk_out signal is present. The delay in waiting for this positive edge depends on the baud rate of the serial transfer. Before polling the BUSY status, you should first poll the Transit FIFO Empty (TFE) status (waiting for 1) or wait for (BAUDR * SPI clock) clock cycles. †

Master SPI and SSP Serial Transfers

"Motorola SPI Protocol" and "Texas Instruments Synchronous Serial Protocol (SSP)" describe the SPI and SSP serial protocols, respectively. †

When the transfer mode is "transmit and receive" or "transmit only" (TMOD = 0 or TMOD = 1, respectively), transfers are terminated by the shift control logic when the transmit FIFO buffer is empty. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level (TXFTLR) can be used to early interrupt (Transmit FIFO Empty Interrupt) the processor indicating that the transmit FIFO buffer is nearly empty. †

When the DMA is used in conjunction with the SPI master, the transmit data level (DMATDLR) can be used to early request the DMA Controller, indicating that the transmit FIFO buffer is nearly empty. The FIFO buffer can then be refilled with data to continue the serial transfer. The user may also write a block of data (at least two FIFO buffer entries) into the transmit FIFO buffer before enabling a serial slave. This ensures that serial transmission does not begin until the number of data frames that make up the continuous transfer are present in the transmit FIFO buffer. †

When the transfer mode is "receive only" (TMOD = 2), a serial transfer is started by writing one "dummy" data word into the transmit FIFO buffer when a serial slave is selected. The txd output from the SPI controller is held at a constant logic level for the duration of the serial transfer. The transmit FIFO buffer is popped only once at the beginning and may remain empty for the duration of the serial transfer. The end of the serial transfer is controlled by the "number of data frames" (NDF) field in control register 1 (CTRLR1). †

If, for example, you want to receive 24 data frames from a serial-slave peripheral, you should program the NDF field with the value 23; the receive logic terminates the serial transfer when the number of frames received is equal to the NDF value plus one. This transfer mode increases the bandwidth of the system bus as the transmit FIFO buffer never needs to be serviced during the transfer. The receive FIFO buffer should be read each time the receive FIFO buffer generates a FIFO full interrupt request to prevent an overflow. †

When the transfer mode is "eeprom_read" (TMOD = 3), a serial transfer is started by writing the opcode and/or address into the transmit FIFO buffer when a serial slave (EEPROM) is selected. The opcode and address are transmitted to the EEPROM device, after which read data is received from the EEPROM device and stored in the receive FIFO buffer. The end of the serial transfer is controlled by the NDF field in the control register 1 (CTRLR1). †

Note: EEPROM read mode is not supported when the SPI controller is configured to be in the SSP mode. †

The receive FIFO threshold level (RXF TLR) can be used to give early indication that the receive FIFO buffer is nearly full. When a DMA is used, the receive data level (DMARDLR) can be used to early request the DMA Controller, indicating that the receive FIFO buffer is nearly full. †

Related Information

- **Motorola SPI Protocol** on page 19-14
- **Texas Instruments Synchronous Serial Protocol (SSP)** on page 19-16
- **SPI Controller Address Map and Register Definitions** on page 19-30

Master Microwire Serial Transfers

“National Semiconductor Microwire Protocol” describes the Microwire serial protocol in detail. †

Microwire serial transfers from the SPI serial master are controlled by the Microwire Control Register (MWCR). The MHS bit field enables and disables the Microwire handshaking interface. The MDD bit field controls the direction of the data frame (the control frame is always transmitted by the master and received by the slave). The MWMOD bit field defines whether the transfer is sequential or nonsequential. †

All Microwire transfers are started by the SPI serial master when there is at least one control word in the transmit FIFO buffer and a slave is enabled. When the SPI master transmits the data frame (MDD =1), the transfer is terminated by the shift logic when the transmit FIFO buffer is empty. When the SPI master receives the data frame (MDD = 1), the termination of the transfer depends on the setting of the MWMOD bit field. If the transfer is nonsequential (MWMOD = 0), it is terminated when the transmit FIFO buffer is empty after shifting in the data frame from the slave. When the transfer is sequential (MWMOD = 1), it is terminated by the shift logic when the number of data frames received is equal to the value in the CTRLR1 register plus one. †

When the handshaking interface on the SPI master is enabled (MHS =1), the status of the target slave is polled after transmission. Only when the slave reports a *ready status* does the SPI master complete the transfer and clear its BUSY status. If the transfer is continuous, the next control/data frame is not sent until the slave device returns a *ready status*. †

Related Information

National Semiconductor Microwire Protocol on page 19-17

SPI Slave

The SPI slave handles serial communication with transfer initiated and controlled by serial master peripheral devices.

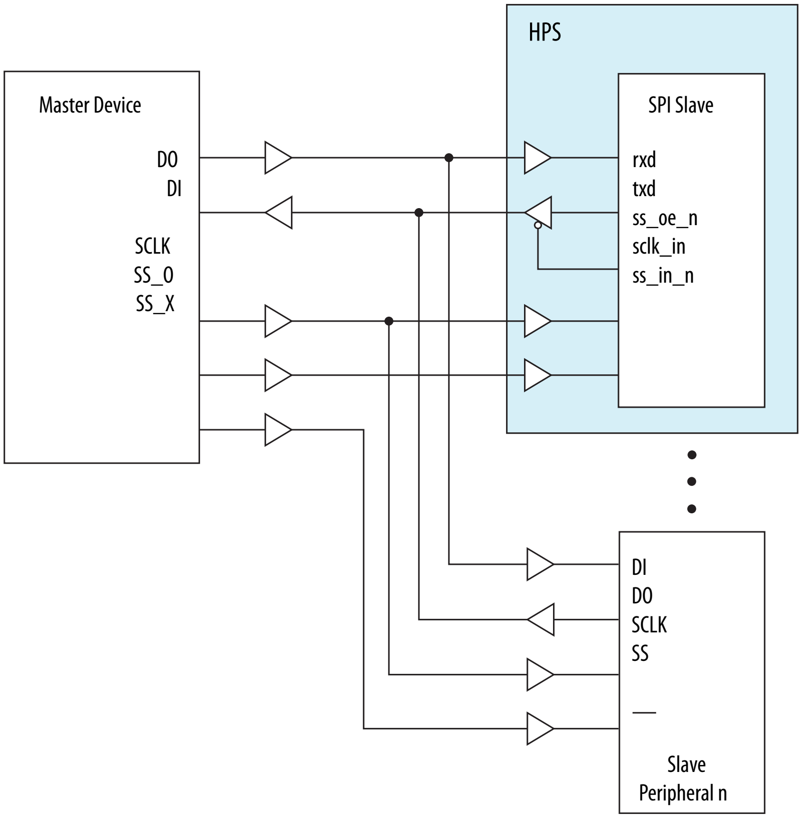
- sclk_in—serial clock to the SPI slave †
- ss_in_n—slave select input to the SPI slave †
- ss_oe_n—output enable for the SPI master or slave †
- txd—transmit data line for the SPI master or slave †
- rxd—receive data line for the SPI master or slave †

When the SPI serial slave is selected, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk_in), driven from the SPI master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge. †

When the SPI serial slave is not selected, it must not interfere with data transfers between the serial-master and other serial-slave devices. When the serial slave is not selected, its txd output is buffered, resulting in a high impedance drive onto the SPI master rxd line. The buffers shown in the SPI Slave diagram are external to SPI controller. spi_oe_n is the SPI slave output enable signal. †

The serial clock that regulates the data transfer is generated by the serial-master device and input to the SPI slave on sclk_in. The slave remains in an idle state until selected by the bus master. When not actively transmitting data, the slave must hold its txd line in a high impedance state to avoid interference with serial transfers to other slave devices. The SPI slave output enable (ss_oe_n) signal is available for use to control the txd output buffer. The slave continues to transfer data to and from the master device as long as it is selected. If the master transmits to all serial slaves, a control bit (SLV_OE) in the SPI control register 0 (CTRLR0) can be programmed to inform the slave if it should respond with data from its txd line. †

Figure 19-5: SPI Slave



Slave SPI and SSP Serial Transfers †

“Motorola SPI Protocol” and the “Texas Instruments Synchronous Serial Protocol (SSP)” contain a description of the SPI and SSP serial protocols, respectively. †

If the SPI slave is *receive only* (TMOD=2), the transmit FIFO buffer need not contain valid data because the data currently in the transmit shift register is resent each time the slave device is selected. The TXE error flag in the status register (SR) is not set when TMOD=2. You should mask the Transmit FIFO Empty Interrupt when this mode is used. †

If the SPI slave transmits data to the master, you must ensure that data exists in the transmit FIFO buffer before a transfer is initiated by the serial-master device. If the master initiates a transfer to the SPI slave when no data exists in the transmit FIFO buffer, an error flag (TXE) is set in the SPI status register, and the previously transmitted data frame is resent on txd. For continuous data transfers, you must ensure that the transmit FIFO buffer does not become empty before all the data have been transmitted. The transmit FIFO threshold level register (TXFTLR) can be used to early interrupt (Transmit FIFO Empty Interrupt) the processor, indicating that the transmit FIFO buffer is nearly empty. When a DMA Controller is used, the DMA transmit data level register (DMATDLR) can be used to early request the DMA Controller, indicating that the transmit FIFO buffer is nearly empty. The FIFO buffer can then be refilled with data to continue the serial transfer. †

The receive FIFO buffer should be read each time the receive FIFO buffer generates a FIFO full interrupt request to prevent an overflow. The receive FIFO threshold level register (RXFTLR) can be used to give early indication that the receive FIFO buffer is nearly full. When a DMA Controller is used, the DMA receive data level register (DMARDLR) can be used to early request the DMA controller, indicating that the receive FIFO buffer is nearly full. †

Related Information

- Motorola SPI Protocol on page 19-14
- Texas Instruments Synchronous Serial Protocol (SSP) on page 19-16

Serial Transfers

“National Semiconductor Microwire Protocol” describes the Microwire serial protocol in detail, including timing diagrams and information about how data are structured in the transmit and receive FIFO buffers before and after a serial transfer. The Microwire protocol operates in much the same way as the SPI protocol. There is no decode of the control frame by the SPI slave device. †

Related Information

National Semiconductor Microwire Protocol on page 19-17

Glue Logic for Master Port ss_in_n

When configured as a master, the SPI has an input, ssi_in_n, which can be used by the deciding logic in a multi-master system to disable one master when another has priority. The polarity of this signal depends on the serial protocol in use, and the protocol is dynamically selectable.

The table below lists the three protocols and the effect of ss_in_n on the ability of the master to transfer data. Note that for the SSP protocol the effect of ss_in_n is inverted with respect to the other protocols.

Protocol	ss_in_n value	Effect on Serial Transfer
Motorola SPI	1	Enabled
	0	Disabled
National Semiconductor Microwire	1	Enabled
	0	Disabled
Texas Instruments Serial Protocol (SSP)	1	Disabled
	0	Enabled

Partner Connection Interfaces

The SPI can connect to any serial-master or serial-slave peripheral device using one of several interfaces.

Motorola SPI Protocol

The inactive state of the serial clock is low. The data frame can be 4 to 16 bits in length. †

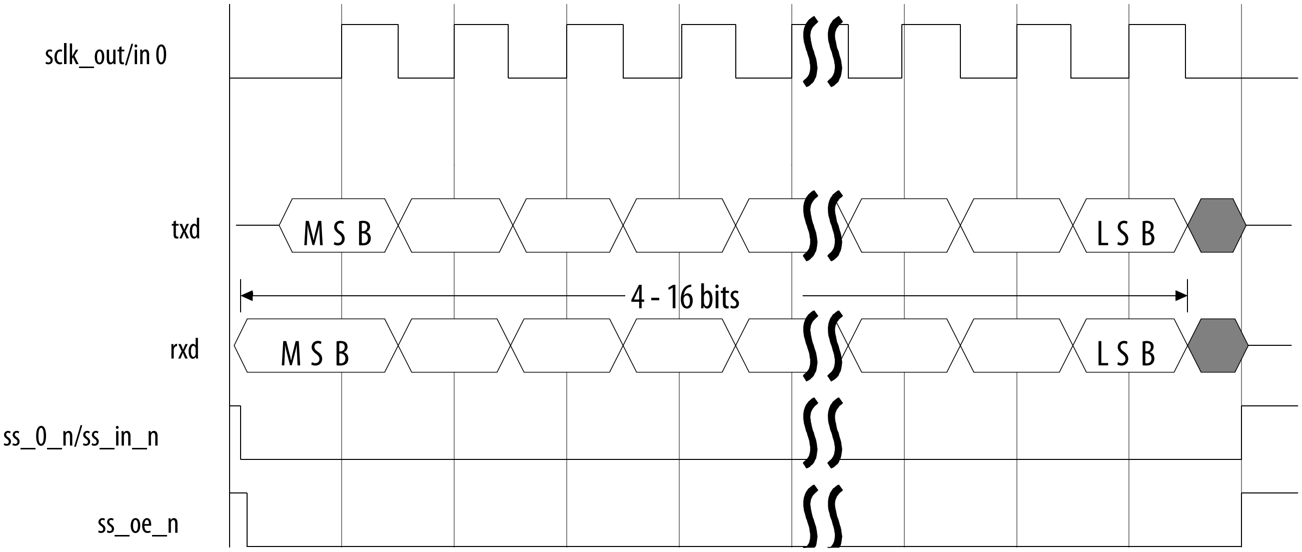
Data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. †

The slave select signal takes effect only when used as slave SPI. For master SPI, the data transmission begins as soon as the output enable signal is deasserted.

The following signals are illustrated in the timing diagrams in this section: †

- `sclk_out`—serial clock from SPI master †
- `sclk_in`—serial clock from SPI slave †
- `ss_0_n`—slave select signal from SPI master †
- `ss_oe_n`—output enable for the SPI master or slave †
- `txd`—transmit data line for the SPI master or slave †
- `rxd`—receive data line for the SPI master or slave †

Figure 19-6: SPI Serial Format



There are four possible transfer modes on the SPI controller for performing SPI serial transactions; refer to “Transfer Modes” . For *transmit and receive transfers* (transfer mode field (9:8) of the Control Register 0 = 0), data transmitted from the SPI controller to the external serial device is written into the transmit FIFO buffer. Data received from the external serial device into the SPI controller is pushed into the receive FIFO buffer. †

Note: For *transmit only* transfers (transfer mode field (9:8) of the Control Register 0 = 1), data transmitted from the SPI controller to the external serial device is written into the transmit FIFO buffer. As the data received from the external serial device is deemed invalid, it is not stored in the SPI receive FIFO buffer. †

For *receive only* transfers (transfer mode field (9:8) of the Control Register 0 = 2), data transmitted from the SPI controller to the external serial device is invalid, so a single dummy word is written into the transmit FIFO buffer to begin the serial transfer. The `txd` output from the SPI controller is held at a constant logic level for the duration of the serial transfer. Data received from the external serial device into the SPI controller is pushed into the receive FIFO buffer. †

For eeprom_read transfers (transfer mode field [9:8] of the Control Register 0 = 3), opcode and/or EEPROM address are written into the transmit FIFO buffer. During transmission of these control frames, received data is not captured by the SPI master. After the control frames have been transmitted, receive data from the EEPROM is stored in the receive FIFO buffer.

SPI Serial Format

Two different modes of continuous data transfers are supported when `SCPH` = 0 and 1.†

- When the clock phase, SCPH = 0, the SPI Controller toggles the slave select signal between frames and the serial clock is held to its default value while the slave select signal is active.†
- When SCPH = 1, the slave select is held active for the duration of the transfer.†

Figure 19-7: Serial Format Continuous Transfers (SCPH =0)

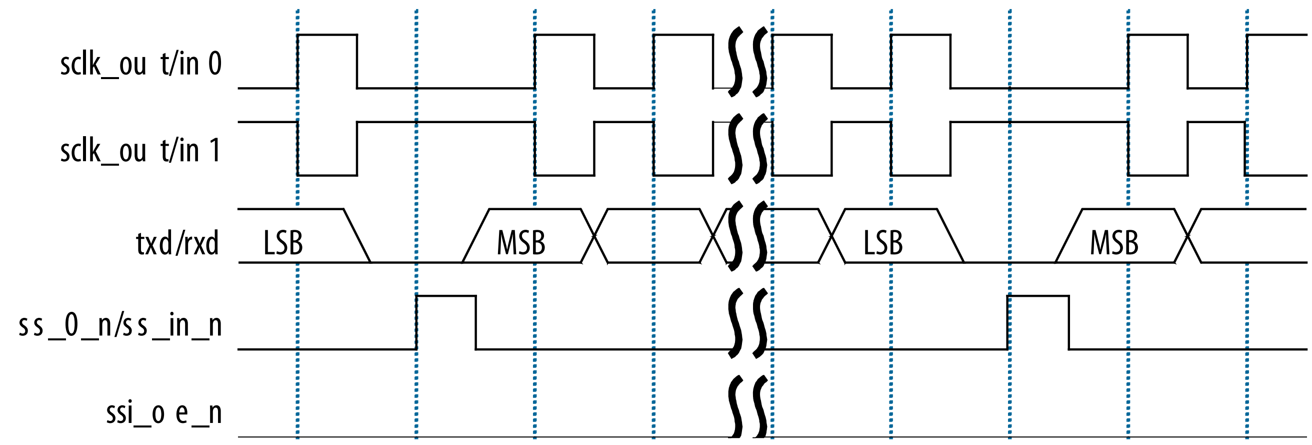
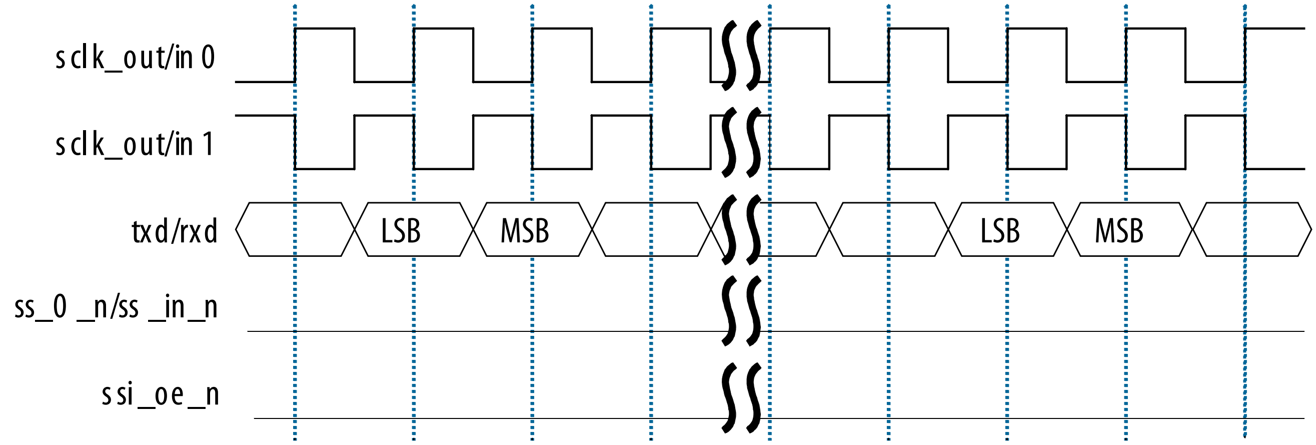


Figure 19-8: Serial Format Continuous Transfers (SCPH =1)



Related Information

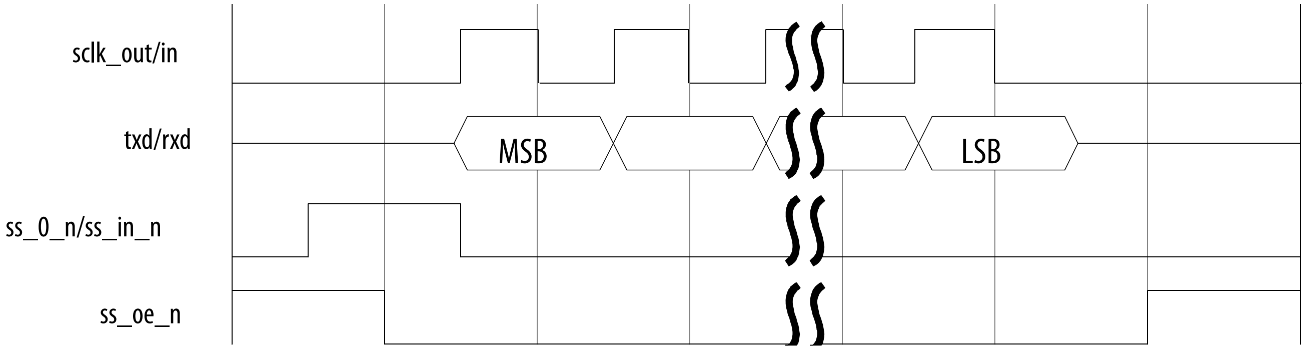
Transfer Modes on page 19-8

Texas Instruments Synchronous Serial Protocol (SSP)

Data transfers begin by asserting the frame indicator line (`ss_0_n`) for one serial clock period. Data to be transmitted are driven onto the `txd` line one serial clock cycle later; similarly data from the slave are driven onto the `rxd` line. Data are propagated on the rising edge of the serial clock (`sclk_out/sclk_in`) and captured on the falling edge. The length of the data frame ranges from 4 to 16 bits.

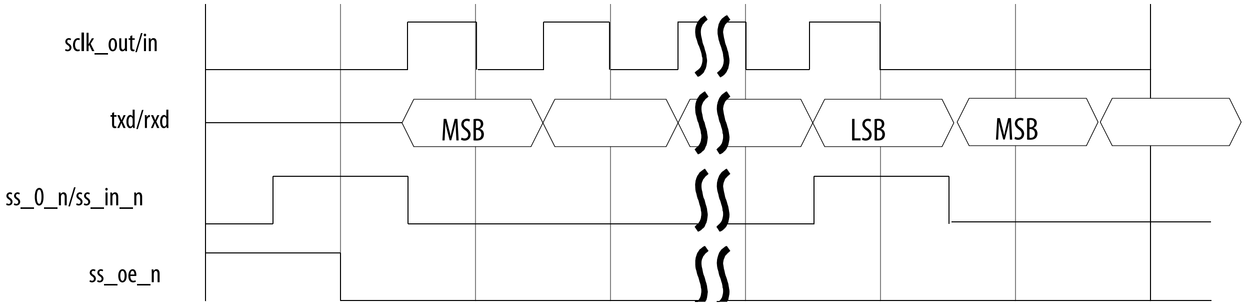
Note: The slave select signal (`ss_0_n`) takes effect only when used as slave SPI. For master SPI, the data transmission begins as soon as the output enable signal is deasserted.

Figure 19-9: SSP Serial Format



Continuous data frames are transferred in the same way as single data frames. The frame indicator is asserted for one clock period during the same cycle as the LSB from the current transfer, indicating that another data frame follows. †

Figure 19-10: SSP Serial Format Continuous Transfer



National Semiconductor Microwire Protocol

For the master SPI, data transmission begins as soon as the output enable signal is deasserted. One-half serial clock (`sclk_out`) period later, the first bit of the control is sent out on the `txd` line. The length of the control word can be in the range 1 to 16 bits and is set by writing bit field CFS (bits 15:12) in `CTRLR0`. The remainder of the control word is transmitted (propagated on the falling edge of `sclk_ou t`) by the SPI serial master. During this transmission, no data are present (high impedance) on the serial master's `rxd` line. †

The direction of the data word is controlled by the MDD bit field (bit 1) in the Microwire Control Register (`MWCR`). When MDD=0, this indicates that the SPI serial master receives data from the external serial slave. One clock cycle after the LSB of the control word is transmitted, the slave peripheral responds with a dummy 0 bit, followed by the data frame, which can be 4 to 16 bits in length. Data are propagated on the falling edge of the serial clock and captured on the rising edge. †

Continuous transfers from the Microwire protocol can be sequential or nonsequential, and are controlled by the MWMOD bit field (bit 0) in the MWCR. †

Nonsequential continuous transfers occur, with the control word for the next transfer following immediately after the LSB of the current data word. †

The only modification needed to perform a continuous nonsequential transfer is to write more control words into the transmit FIFO buffer. †

During sequential continuous transfers, only one control word is transmitted from the SPI master. The transfer is started in the same manner as with nonsequential read operations, but the cycle is continued to read further data. The slave device automatically increments its address pointer to the next location and continues to provide data from that location. Any number of locations can be read in this manner; the SPI master terminates the transfer when the number of words received is equal to the value in the `CTRLR1` register plus one. †

When MDD = 1, this indicates that the SPI serial master transmits data to the external serial slave. Immediately after the LSB of the control word is transmitted, the SPI master begins transmitting the data frame to the slave peripheral. †

Note: The SPI controller does not support continuous sequential Microwire writes, where MDD = 1 and MWMOD = 1. †

Continuous transfers occur with the control word for the next transfer following immediately after the LSB of the current data word.

The Microwire handshaking interface can also be enabled for SPI master write operations to external serial-slave devices. To enable the handshaking interface, you must write 1 into the MHS bit field (bit 2) on the MWCR register. When MHS is set to 1, the SPI serial master checks for a ready status from the slave device before completing the transfer, or transmitting the next control word for continuous transfers. †

After the first data word has been transmitted to the serial-slave device, the SPI master polls the `rxd` input waiting for a ready status from the slave device. Upon reception of the ready status, the SPI master begins transmission of the next control word. After transmission of the last data frame has completed, the SPI master transmits a start bit to clear the ready status of the slave device before completing the transfer. †

In the SPI slave, data transmission begins with the falling edge of the slave select signal (`ss_in_0`). One-half serial clock (`sclk_in`) period later, the first bit of the control is present on the `rxd` line. The length of the control word can be in the range of 1 to 16 bits and is set by writing bit field CFS in the `CTRLR0` register. The CFS bit field must be set to the size of the expected control word from the serial master. The remainder of the control word is received (captured on the rising edge of `sclk_in`) by the SPI serial slave. During this reception, no data are driven (high impedance) on the serial slave's `txd` line. †

The direction of the data word is controlled by the MDD bit field (bit 1) MWCR register. When MDD=0, this indicates that the SPI serial slave is to receive data from the external serial master. Immediately after the control word is transmitted, the serial master begins to drive the data frame onto the SPI slave `rxd` line. Data are propagated on the falling edge of the serial clock and captured on the rising edge. The slave-select signal is held active-low during the transfer and is deasserted one-half clock cycle later after the data are transferred. The SPI slave output enable signal is held inactive for the duration of the transfer. †

When MDD=1, this indicates that the SPI serial slave transmits data to the external serial master. Immediately after the LSB of the control word is transmitted, the SPI slave transmits a dummy 0 bit, followed by the 4- to 16-bit data frame on the `txd` line. †

Continuous transfers for a SPI slave occur in the same way as those specified for the SPI master. The SPI slave does not support the handshaking interface, as there is never a busy period. †

Figure 19-11: Single SPI Serial Master Microwire Serial Transfer (MDD=0)

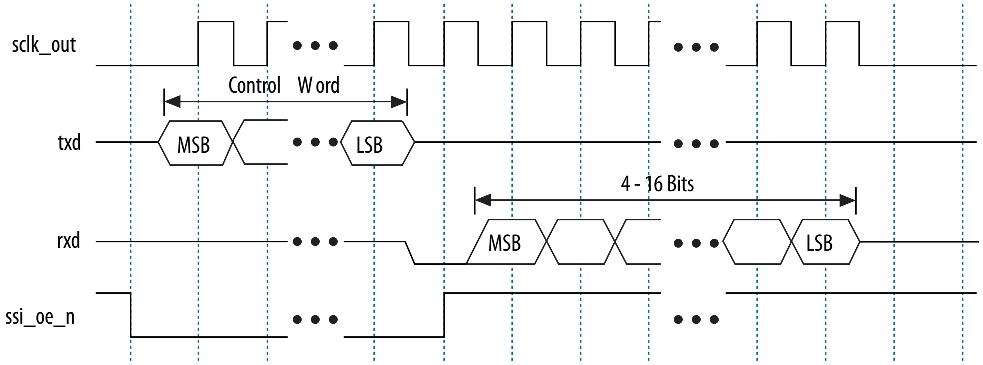
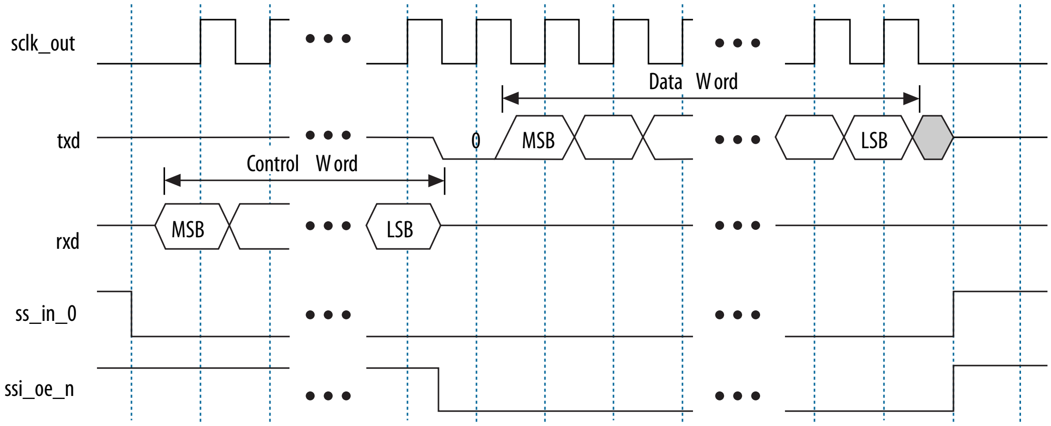


Figure 19-12: Single SPI Slave Microwire Serial Transfer (MDD=1)



DMA Controller Interface

The SPI controller supports DMA signaling to indicate when the receive FIFO buffer has data ready to be read or when the transmit FIFO buffer needs data. It requires two DMA channels, one for transmit data and one for receive data. The SPI controller can issue single or burst DMA transfers and accepts burst acknowledges from the DMA. System software can trigger the DMA burst mode by programming an appropriate value into the threshold registers. The typical setting of the threshold register value is half full.

To enable the DMA Controller interface on the SPI controller, you must write the DMA Control Register (DMACR). Writing a 1 into the TDMAE bit field of DMACR register enables the SPI transmit handshaking interface. Writing a 1 into the RDMAE bit field of the DMACR register enables the SPI receive handshaking interface. †

Slave Interface

The host processor accesses data, control, and status information about the SPI controller through the slave interface. The SPI supports a data bus width of 32 bits.

Control and Status Register Access

Control and status registers within the SPI controller are byte-addressable. The maximum width of the control or status register in the SPI controller is 16 bits. Therefore, all read and write operations to the SPI control and status registers require only one access. †

Data Register Access

The data register (DR) within the SPI controller is 16 bits wide in order to remain consistent with the maximum serial transfer size (data frame). A write operation to DR moves data from the slave write data bus into the transmit FIFO buffer. An read operation from DR moves data from the receive FIFO buffer onto the slave readback data bus. †

Note: The DR register in the SPI controller occupies sixty-four 32-bit locations of the memory map to facilitate burst transfers. There are no burst transactions on the system bus itself, but SPI supports bursts on the system interconnect. Writing to any of these address locations has the same effect as pushing the data from the slave write data bus into the transmit FIFO buffer. Reading from any of these locations has the same effect as popping data from the receive FIFO buffer onto the slave readback data bus. The FIFO buffers on the SPI controller are not addressable

Clocks and Resets

The SPI controller uses the clock and reset signals shown in the following table.

Table 19-5: SPI Controller Clocks and Resets

	Master	Slave
SPI clock	spi_m_clk	l4_main_clk
SPI bit-rate clock	sclk_out	sclk_in
Reset	spim_rst_n	spis_rst_n

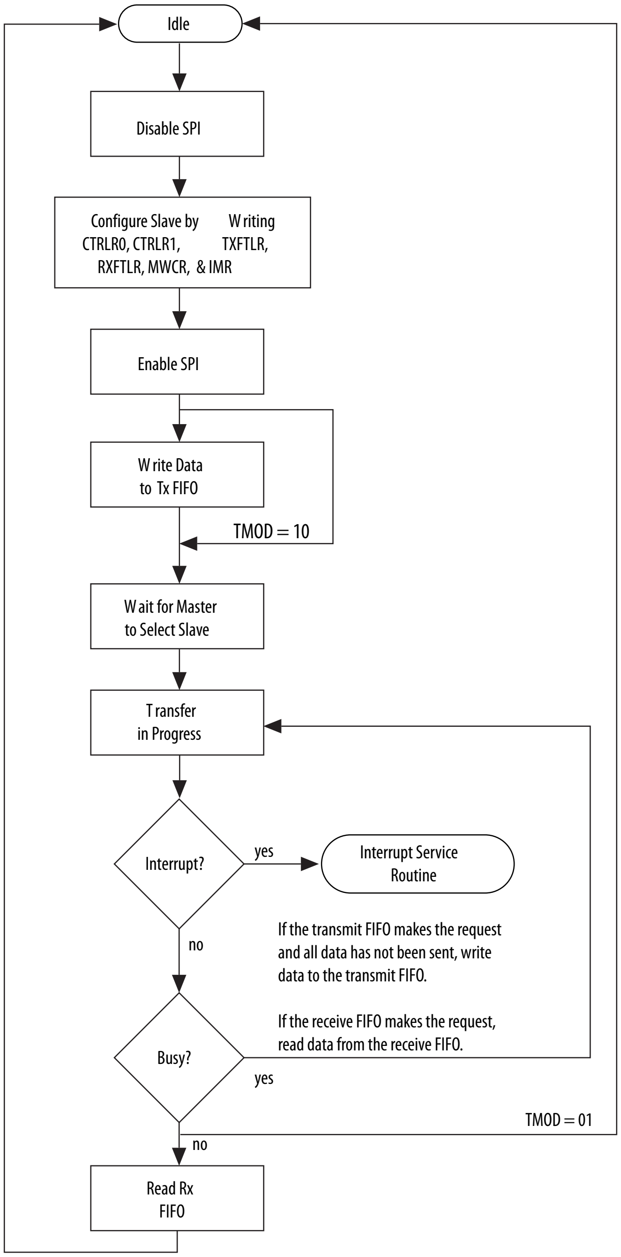
SPI Programming Model

This section describes the programming model for the SPI controller based on the following master and slave transfers:

- Master SPI and SSP serial transfers
- Master Microwire Serial Transfers
- Slave SPI and SSP Serial Transfers
- Slave Microwire Serial Transfers
- Software Control for Slave Selection

Master SPI and SSP Serial Transfers

Figure 19-13: Master SPI or SSP Serial Transfer Software Flow


Idle
Disable SPI
Configure Slave by Writing CTRLR0, CTRLR1, TXFTLR, RXFTLR, MWCR, & IMR
Enable SPI
Write Data to Tx FIFO
TMOD = 10
Wait for Master to Select Slave
Transfer in Progress
Interrupt?
yes
Interrupt Service Routine
no
If the transmit FIFO makes the request and all data has not been sent, write data to the transmit FIFO.
If the receive FIFO makes the request, read data from the receive FIFO.
Busy?
yes
TMOD = 01
no
Read Rx FIFO

To complete an SPI or SSP serial transfer from the SPI master, follow these steps:

1. If the SPI master is enabled, disable it by writing 0 to the SSI Enable register (`SSIENR`).
2. Set up the SPI master control registers for the transfer. You can set these registers in any order.

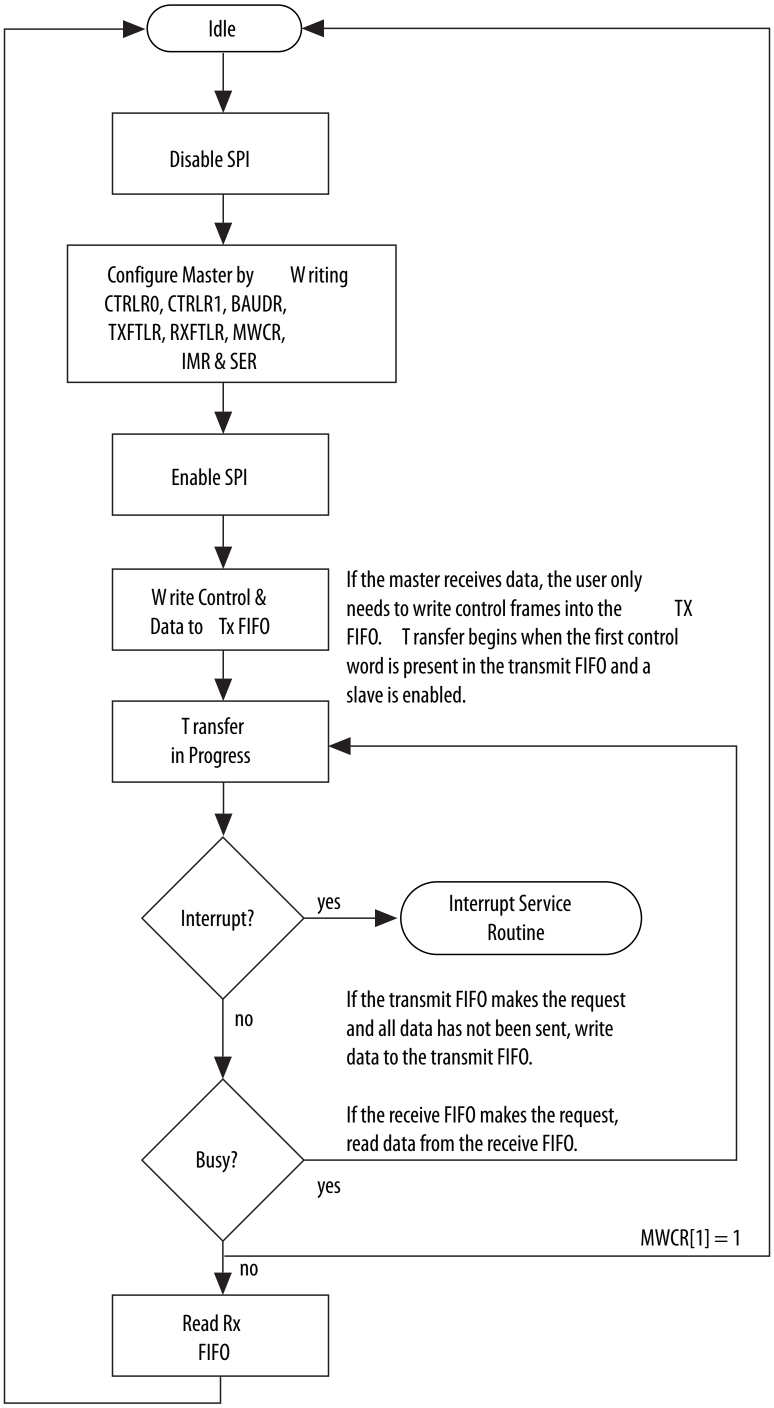
- Write Control Register 0 (`CTRLLR0`). For SPI transfers, you must set the serial clock polarity and serial clock phase parameters identical to the target slave device.
- If the transfer mode is receive only, write Control Register 1 (`CTRLR1`) with the number of frames in the transfer minus 1. For example, if you want to receive four data frames, write 3 to this register.
- Write the Baud Rate Select Register (`BAUDR`) to set the baud rate for the transfer.
- Write the Transmit and Receive FIFO Threshold Level registers (`TXFTLR` and `RXFTLR`) to set FIFO buffer threshold levels.
- Write the `IMR` register to set up interrupt masks.
- Write the Slave Enable Register (`SER`) register to enable the target slave for selection. If a slave is enabled at this time, the transfer begins as soon as one valid data entry is present in the transmit FIFO buffer. If no slaves are enabled prior to writing to the Data Register (`DR`), the transfer does not begin until a slave is enabled.

3. Enable the SPI master by writing 1 to the `SSIENR` register.
4. Write data for transmission to the target slave into the transmit FIFO buffer (write `DR`). If no slaves were enabled in the SER register at this point, enable it now to begin the transfer.
5. Poll the `BUSY` status to wait for the transfer to complete. If a transmit FIFO empty interrupt request is made, write the transmit FIFO buffer (write `DR`). If a receive FIFO full interrupt request is made, read the receive FIFO buffer (read `DR`).

6. The shift control logic stops the transfer when the transmit FIFO buffer is empty. If the transfer mode is receive only (TMOD = 2), the shift control logic stops the transfer when the specified number of frames have been received. When the transfer is done, the `BUSY` status is reset to 0.
7. If the transfer mode is not transmit only (TMOD is not equal to 1), read the receive FIFO buffer until it is empty
8. Disable the SPI master by writing 0 to `SSIENR`.

Master Microwire Serial Transfers

Figure 19-14: Microwire Serial


Idle
Disable SPI
Configure Master by Writing CTRLR0, CTRLR1, BAUDR, TXFTLR, RXFTLR, MWCR, IMR & SER
Enable SPI
Write Control & Data to Tx FIFO
If the master receives data, the user only needs to write control frames into the TX FIFO. Transfer begins when the first control word is present in the transmit FIFO and a slave is enabled.
Transfer in Progress
Interrupt?
yes
Interrupt Service Routine
no
If the transmit FIFO makes the request and all data has not been sent, write data to the transmit FIFO.
If the receive FIFO makes the request, read data from the receive FIFO.
Busy?
yes
MWCR[1] = 1
no
Read Rx FIFO

To complete a Microwire serial transfer from the SPI master, follow these steps:

1. If the SPI master is enabled, disable it by writing 0 to SSIENR.
2. Set up the SPI control registers for the transfer. You can set these registers in any order.
   - Write CTRLR0 to set transfer parameters. If the transfer is sequential and the SPI master receives data, write CTRLR1 with the number of frames in the transfer minus 1. For example, if you want to receive four data frames, write 3 to this register.
   - Write BAUDR to set the baud rate for the transfer.
   - Write TXFTLR and RXFTLR to set FIFO buffer threshold levels.
   - Write the IMR register to set up interrupt masks.

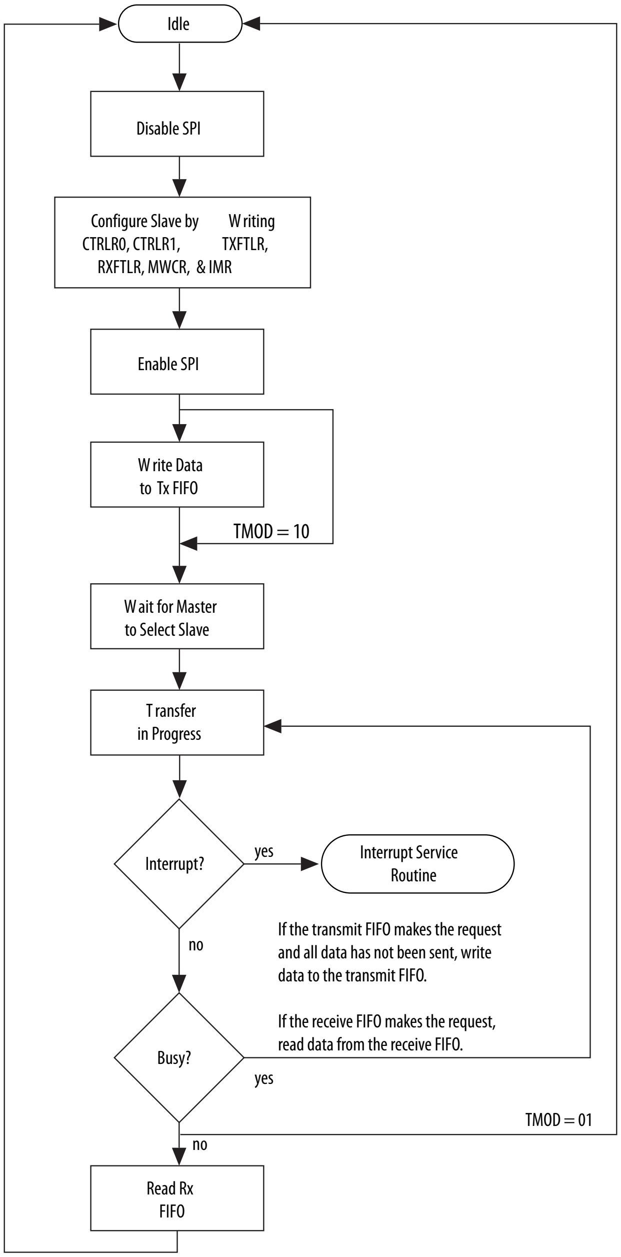
   You can write the SER register to enable the target slave for selection. If a slave is enabled here, the transfer begins as soon as one valid data entry is present in the transmit FIFO buffer. If no slaves are enabled prior to writing to the DR register, the transfer does not begin until a slave is enabled.
3. Enable the SPI master by writing 1 to the SSIENR register.
4. If the SPI master transmits data, write the control and data words into the transmit FIFO buffer (write DR). If the SPI master receives data, write the control word or words into the transmit FIFO buffer. If no slaves were enabled in the SER register at this point, enable now to begin the transfer.
5. Poll the BUSY status to wait for the transfer to complete. If a transmit FIFO empty interrupt request is made, write the transmit FIFO buffer (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO buffer (read DR).
6. The shift control logic stops the transfer when the transmit FIFO buffer is empty. If the transfer mode is sequential and the SPI master receives data, the shift control logic stops the transfer when the specified number of data frames is received. When the transfer is done, the BUSY status is reset to 0.
7. If the SPI master receives data, read the receive FIFO buffer until it is empty.
8. Disable the SPI master by writing 0 to SSIENR.

Related Information

SPI Controller Address Map and Register Definitions on page 19-30

Slave SPI and SSP Serial Transfers

Figure 19-15: Slave SPI or SSP Serial Transfer Software Flow


Idle
Disable SPI
Configure Slave by Writing CTRLR0, CTRLR1, TXFTLR, RXFTLR, MWCR, & IMR
Enable SPI
Write Data to Tx FIFO
TMOD = 10
Wait for Master to Select Slave
Transfer in Progress
Interrupt?
yes
Interrupt Service Routine
no
If the transmit FIFO makes the request and all data has not been sent, write data to the transmit FIFO.
If the receive FIFO makes the request, read data from the receive FIFO.
Busy?
yes
TMOD = 01
no
Read Rx FIFO

To complete a continuous serial transfer from a serial master to the SPI slave, follow these steps:

1. If the SPI slave is enabled, disable it by writing 0 to `SSIENR`.
2. Set up the SPI control registers for the transfer. You can set these registers in any order.

- Write CTRLR0 (for SPI transfers, set SCPH and SCPOL identical to the master device).
- Write TXFTLR and RXFTLR to set FIFO buffer threshold levels.
- Write the IMR register to set up interrupt masks.

3. Enable the SPI slave by writing 1 to the SSIENR register.
4. If the transfer mode is transmit and receive (TMOD=2'b00) or transmit only (TMOD=2'b01), write data for transmission to the master into the transmit FIFO buffer (write DR). If the transfer mode is receive only (TMOD=2'b10), you need not write data into the transmit FIFO buffer. The current value in the transmit shift register is retransmitted.
5. The SPI slave is now ready for the serial transfer. The transfer begins when a serial-master device selects the SPI slave.
6. When the transfer is underway, the BUSY status can be polled to return the transfer status. If a transmit FIFO empty interrupt request is made, write the transmit FIFO buffer (write DR). If a receive FIFO full interrupt request is made, read the receive FIFO buffer (read DR).
7. The transfer ends when the serial master removes the select input to the SPI slave. When the transfer is completed, the BUSY status is reset to 0.
8. If the transfer mode is not transmit only (TMOD != 01), read the receive FIFO buffer until empty.
9. Disable the SPI slave by writing 0 to SSIENR.

Slave Microwire Serial Transfers

For the SPI slave, the Microwire protocol operates in much the same way as the SPI protocol. The SPI slave does not decode the control frame.

Software Control for Slave Selection

When using software to select slave devices, the input select lines from serial slave devices is connected to a single slave select output on the SPI master.

Example: Slave Selection Software Flow for SPI Master

1. If the SPI master is enabled, disable it by writing 0 to SSIENR.
2. Write CTRLR0 to match the required transfer.
3. If the transfer is receive only, write the number of frames into CTRLR1.
4. Write BAUDR to set the transfer baud rate.
5. Write TXFTLR and RXFTLR to set FIFO buffer threshold levels.
6. Write IMR register to set interrupt masks.
7. Write SER register bit[0] to logic '1' to select slave 1 in this example.
8. Write SSIENR register bit[0] to logic '1' to enable SPI master.

Example: Slave Selection Software Flow for SPI Slave

1. If the SPI slave is enabled, disable it by writing 0 to SSIENR.
2. Write CTRLR0 to match the required transfer.
3. Write TXFTLR and RXFTLR to set FIFO buffer threshold levels.
4. Write IMR register to set interrupt masks.
5. Write SSIENR register bit[0] to logic '1' to enable SPI slave.
6. If the SPI slave transmits data, write data into TX FIFO buffer. Note all other SPI slaves are disabled (SSIENR = 0) and therefore will not respond to an active level on their ss_in_n port.

The FIFO buffer depth (FIFO_DEPTH) for both the RX and TX buffers in the SPI controller is 256 entries.

DMA Controller Operation

To enable the DMA controller interface on the SPI controller, you must write the DMA Control Register (DMACR). Writing a 1 to the TDMAE bit field of DMACR register enables the SPI controller transmit handshaking interface. Writing a 1 to the RDMAE bit field of the DMACR register enables the SPI controller receive handshaking.†

Related Information

DMA Controller on page 16-1
For details about the DMA controller, refer to the *DMA Controller* chapter.

Transmit FIFO Buffer Underflow

During SPI serial transfers, transmit FIFO buffer requests are made to the DMA Controller whenever the number of entries in the transmit FIFO buffer is less or equal to the value in DMA Transmit Data Level Register (DMATDLR); also known as the watermark level. The DMA Controller responds by writing a burst of data to the transmit FIFO buffer, of length specified as DMA burst length.†

Note: Data should be fetched from the DMA often enough for the transmit FIFO buffer to perform serial transfers continuously, that is, when the FIFO buffer begins to empty, another DMA request should be triggered. Otherwise, the FIFO buffer will run out of data (underflow). To prevent this condition, you must set the watermark level correctly.†

Related Information

DMA Controller on page 16-1
For details about the DMA burst length microcode setup, refer to the *DMA Controller* chapter.

Transmit Watermark Level

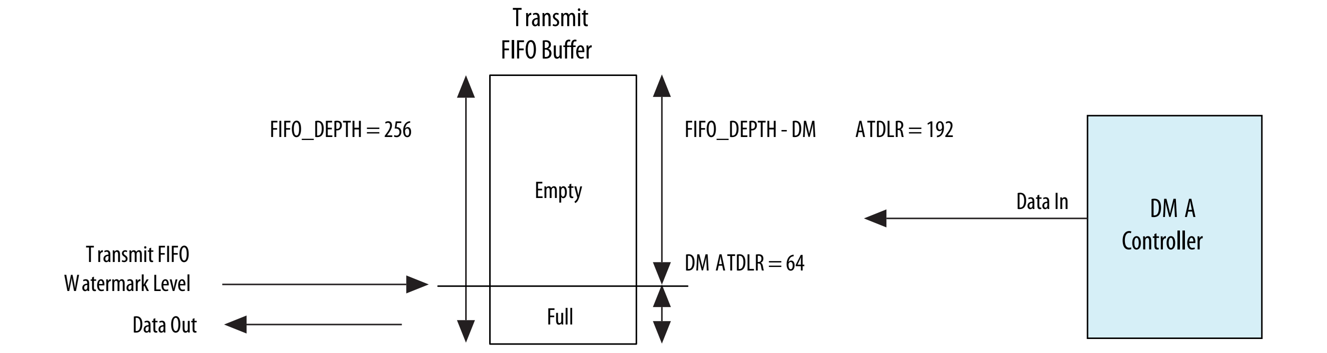
Consider the example where the assumption is made: †

DMA burst length = FIFO_DEPTH - DMATDLR

Here the number of data items to be transferred in a DMA burst is equal to the empty space in the transmit FIFO buffer. Consider the following two different watermark level settings. †

- Case 1: DMATDLR = 64: †
- Transmit FIFO watermark level = DMATDLR = 64: †
- DMA burst length = FIFO_DEPTH - DMATDLR = 192: †
- SPI transmit FIFO_DEPTH = 256: †
- Block transaction size = 960: †

Figure 19-16: Transmit FIFO Watermark Level = 64



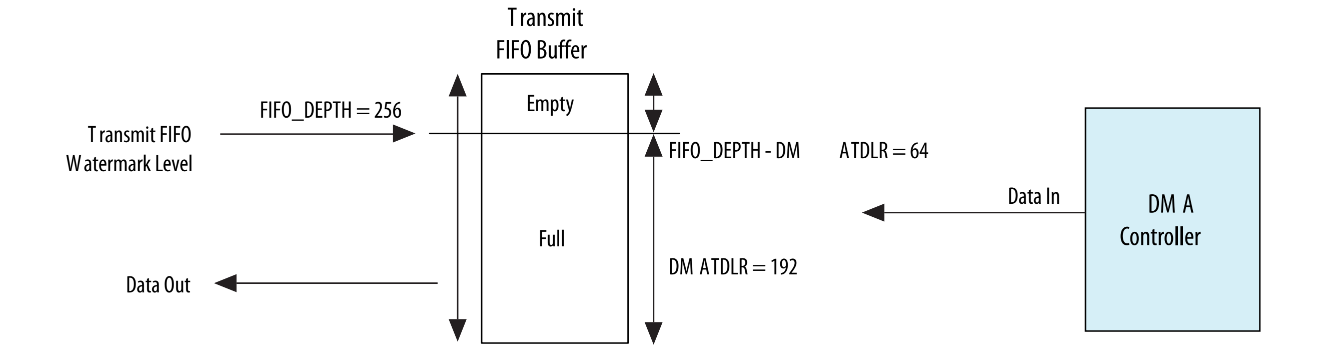
The number of burst transactions needed equals the block size divided by the number of data items per burst:

Block transaction size/DMA burst length = 960/192 = 5

The number of burst transactions in the DMA block transfer is 5. But the watermark level, DMATDLR, is quite low. Therefore, the probability of transmit underflow is high where the SPI serial transmit line needs to transmit data, but there is no data left in the transmit FIFO buffer. This occurs because the DMA has not had time to service the DMA request before the FIFO buffer becomes empty.

- Case 2: DMATDLR = 192†
- Transmit FIFO watermark level = D MATDLR = 192 †
- DMA burst length = FIFO_DEPTH - DMATDLR = 64 †
- SPI transmit FIFO_DEPTH = 256 †
- Block transaction size = 960 †

Figure 19-17: Transmit FIFO Watermark Level = 192



Number of burst transactions in block: †

Block transaction size/DMA burst length = 960/64 = 15 †

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, DMATDLR, is high. Therefore, the probability of SPI transmit underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the SPI transmit FIFO buffer becomes empty. †

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bursts per block and worse bus utilization than the former case. †

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the SPI transmits data to the rate at which the DMA can respond to destination burst requests. †

Transmit FIFO Buffer Overflow

Setting the DMA transaction burst length to a value greater than the watermark level that triggers the DMA request may cause overflow when there is not enough space in the transmit FIFO buffer to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow: †

DMA burst length <= `FIFO_DEPTH` - `DMATDLR`

In case 2: `DMATDLR` = 192, the amount of space in the transmit FIFO buffer at the time of the burst request is made is equal to the DMA burst length. Thus, the transmit FIFO buffer may be full, but not overflowed, at the completion of the burst transaction. †

Therefore, for optimal operation, DMA burst length should be set at the FIFO buffer level that triggers a transmit DMA request; that is: †

DMA burst length = `FIFO_DEPTH` - `DMATDLR`

Adhering to this equation reduces the number of DMA bursts needed for block transfer, and this in turn improves bus utilization. †

The transmit FIFO buffer will not be full at the end of a DMA burst transfer if the SPI controller has successfully transmitted one data item or more on the serial transmit line during the transfer. †

Receive FIFO Buffer Overflow

During SPI serial transfers, receive FIFO buffer requests are made to the DMA whenever the number of entries in the receive FIFO buffer is at or above the DMA Receive Data Level Register, that is `DMATDLR` + 1.This is known as the watermark level. The DMA responds by fetching a burst of data from the receive FIFO buffer. †

Data should be fetched by the DMA often enough for the receive FIFO buffer to accept serial transfers continuously, that is, when the FIFO buffer begins to fill, another DMA transfer is requested. Otherwise the FIFO buffer will fill with data (overflow). To prevent this condition, the user must set the watermark level correctly. †

Choosing Receive Watermark Level

Similar to choosing the transmit watermark level, the receive watermark level, `DMATDLR` + 1, should be set to minimize the probability of overflow. It is a trade off between the number of DMA burst transactions required per block versus the probability of an overflow occurring. †

Related Information

Texas Instruments Synchronous Serial Protocol (SSP) on page 19-16

Receive FIFO Buffer Underflow

Setting the source transaction burst length greater than the watermark level can cause underflow where there is not enough data to service the source burst request. Therefore, the following equation must be adhered to avoid underflow: †

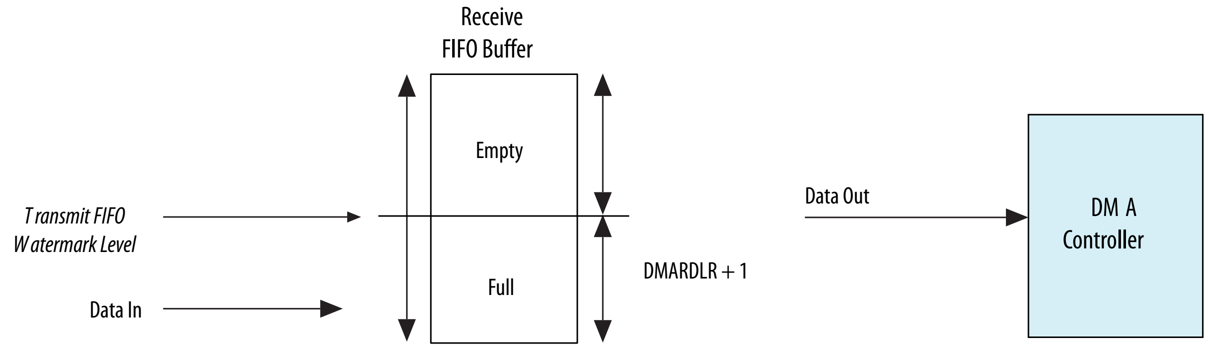
DMA burst length = `DMATDLR` + 1

If the number of data items in the receive FIFO buffer is equal to the source burst length at the time of the burst request is made, the receive FIFO buffer may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA burst length should be set at the watermark level, `DMATDLR` + 1. †

Adhering to this equation reduces the number of DMA bursts in a block transfer, which in turn can improve bus utilization. †

Note: The receive FIFO buffer will not be empty at the end of the source burst transaction if the SPI controller has successfully received one data item or more on the serial receive line during the burst. †

Figure 19-18: Receive FIFO Buffer



SPI Controller Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- SPI Slave Module 0
- SPI Slave Module 1
- SPI Master Module 0
- SPI Master Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

SPI Master Module Address Map

Registers in the SPI Master module

Module Instance	Base Address
spim0	0xFFF00000
spim1	0xFFF01000

SPI Master Module

Register	Offset	Width	Acces s	Reset Value	Description
ctrlr0 on page 19-32	0x0	32	RW	0x7	Control Register 0
ctrlr1 on page 19-34	0x4	32	RW	0x0	Control Register 1
spienr on page 19-35	0x8	32	RW	0x0	Enable Register
mwcr on page 19-36	0xC	32	RW	0x0	Microwire Control Register
ser on page 19-37	0x10	32	RW	0x0	Slave Enable Register
baudr on page 19-38	0x14	32	RW	0x0	Baud Rate Select Register
txftlr on page 19-39	0x18	32	RW	0x0	Transmit FIFO Threshold Level Register
rxftlr on page 19-39	0x1C	32	RW	0x0	Receive FIFO Threshold Level Register
txflr on page 19-40	0x20	32	RO	0x0	Transmit FIFO Level Register
rxflr on page 19-41	0x24	32	RO	0x0	Receive FIFO Level Register
sr on page 19-41	0x28	32	RO	0x6	Status Register
imr on page 19-42	0x2C	32	RW	0x3F	Interrupt Mask Register
isr on page 19-44	0x30	32	RO	0x0	Interrupt Status Register
risr on page 19-45	0x34	32	RO	0x0	Raw Interrupt Status Register
txoicr on page 19-47	0x38	32	RO	0x0	Transmit FIFO Overflow Interrupt Clear Register
rxoicr on page 19-47	0x3C	32	RO	0x0	Receive FIFO Overflow Interrupt Clear Register
rxuicr on page 19-48	0x40	32	RO	0x0	Receive FIFO Underflow Interrupt Clear Register
icr on page 19-48	0x48	32	RO	0x0	Interrupt Clear Register
dmacr on page 19-49	0x4C	32	RW	0x0	DMA Control Register
dmatdlr on page 19-50	0x50	32	RW	0x0	DMA Transmit Data Level Register
dmardlr on page 19-51	0x54	32	RW	0x0	DMA Receive Data Level Register
idr on page 19-51	0x58	32	RO	0x5510000	Identification Register
spi_version_id on page 19-52	0x5C	32	RW	0x3332302A	Component Version Register

Register	Offset	Width	Access	Reset Value	Description
dr on page 19-52	0x60	32	RW	0x0	Data Register
rx_sample_dly on page 19-53	0xFC	32	RW	0x0	RX Sample Delay Register

ctrlr0

This register controls the serial data transfer. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00000
spim1	0xFFF01000	0xFFF01000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cfs RW 0x0				srl RW 0x0	Reserved	tmod RW 0x0		scpol RW 0x0	scph RW 0x0	frf RW 0x0		dfs RW 0x7			

ctrlr0 Fields

Bit	Name	Description	Access	Reset
15:12	cfs	Selects the length of the control word for the Microwire frame format. The length (in bits) is the value of this field plus 1.	RW	0x0
11	srl	Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. **Value** **Description** 0x0 Normal Mode Operation 0x1 Test Mode Operation	RW	0x0

Bit	Name	Description	Access	Reset
9:8	`tmod`	Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory. It is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory, where it can be accessed by the host processor. **Value** **Description** 0x0 Transmit & and Receive 0x1 Transmit Only 0x2 Receive Only 0x3 EEPROM Read	`RW`	`0x0`
7	`scpol`	Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the SPI Master is not actively transferring data on the serial bus. **Value** **Description** 0x0 Inactive state of serial clock is low 0x1 Inactive state of serial clock is high	`RW`	`0x0`
6	`scph`	Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal. When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock. **Value** **Description** 0x0 Serial clock toggles in middle of first data bit 0x1 Serial clock toggles at start of first data bit	`RW`	`0x0`

Bit	Name	Description	Access	Reset
5:4	`frf`	Selects which serial protocol transfers the data. **Value** — **Description** 0x0 — Motorola SPI 0x1 — Texas Instruments SSP 0x2 — National Semi Microwire	RW	0x0
3:0	`dfs`	Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. **Value** — **Description** 0x3 — 4-bit serial data transfer 0x4 — 5-bit serial data transfer 0x5 — 6-bit serial data transfer 0x6 — 7-bit serial data transfer 0x7 — 8-bit serial data transfer 0x8 — 9-bit serial data transfer 0x9 — 10-bit serial data transfer 0xA — 11-bit serial data transfer 0xB — 12-bit serial data transfer 0xC — 13-bit serial data transfer 0xD — 14-bit serial data transfer 0xE — 15-bit serial data transfer 0xF — 16-bit serial data transfer	RW	0x7

ctrlr1

Control register 1 controls the end of serial transfers when in receive-only mode. It is impossible to write to this register when the SPI Master is enabled.The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00004
spim1	0xFFF01000	0xFFF01004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ndf RW 0x0															

ctrlr1 Fields

Bit	Name	Description	Access	Reset
15:0	ndf	When TMOD = 10 or TMOD =11, this register field sets the number of data frames to be continuously received by the SPI Master. The SPI Master continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer.	RW	0x0

spienr

Enables and Disables all SPI operations.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00008
spim1	0xFFF01000	0xFFF01008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															spi_en RW 0x0

spienr Fields

Bit	Name	Description	Access	Reset
0	spi_en	Enables and disables all SPI Master operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the SPI Master control registers when enabled. **Value** **Description** 0x0 Disables serial transfer operations 0x1 Enables serial transfer operations	RW	0x0

mwcr

This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0000C
spim1	0xFFF01000	0xFFF0100C

Offset: 0xC

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="13">Reserved</td><td>mhs
RW
0x0</td><td>mdd
RW
0x0</td><td>mwmod
RW 0x0</td></tr>
</table>

mwcr Fields

Bit	Name	Description	Access	Reset
2	mhs	Used to enable and disable the busy/ready handshaking interface for the Microwire protocol. When enabled, the SPI Master checks for a ready status from the target slave, after the transfer of the last data/control bit, before clearing the BUSY status in the SR register. **Value** **Description** 0x0 Handshaking interface is disabled 0x1 Handshaking interface is enabled	RW	0x0
1	mdd	Defines the direction of the data word when the Microwire serial protocol is used. **Value** **Description** 0x0 SPI Master receives data 0x1 SPI Master transmits data	RW	0x0
0	mwmod	Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received. **Value** **Description** 0x0 non-sequential transfer 0x1 sequential transfer	RW	0x0

ser

The register enables the individual slave select output lines from the SPI Master. Up to 4 slave-select output pins are available on the SPI Master. You cannot write to this register when SPI Master is busy and when SPI_EN = 1.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00010
spim1	0xFFF01000	0xFFF01010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ser RW 0x0			

ser Fields

Bit	Name	Description	Access	Reset
3:0	ser	Each bit in this register corresponds to a slave select line (spim_ss_x_n] from the SPI Master. When a bit in this register is set (1), the corresponding slave select line from the master is activated when a serial transfer begins. It should be noted that setting or clearing bits in this register have no effect on the corresponding slave select outputs until a transfer is started. Before beginning a transfer, you should enable the bit in this register that corresponds to the slave device with which the master wants to communicate. When not operating in broadcast mode, only one bit in this field should be set. **Value** **Description** 0x0 Slave x Not Selected 0x1 Slave x Selected	RW	0x0

baudr

This register derives the frequency of the serial clock that regulates the data transfer. The 16-bit field in this register defines the spi_m_clk divider value. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00014
spim1	0xFFF01000	0xFFF01014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sckdv RW 0x0															

baudr Fields

Bit	Name	Description	Access	Reset
15:0	`sckdv`	The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (spim_sclk_out) is disabled. The frequency of the spim_sclk_out is derived from the following equation: Fspim_sclk_out = Fspi_m_clk/ SCKDV where SCKDV is any even value between 2 and 65534. For example: for Fspi_m_clk = 3.6864MHz and SCKDV =2 Fspim_sclk_out = 3.6864/2 = 1.8432MHz	RW	0x0

txftlr

This register controls the threshold value for the transmit FIFO memory. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00018
spim1	0xFFF01000	0xFFF01018

Offset: `0x18`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								tft RW 0x0							

txftlr Fields

Bit	Name	Description	Access	Reset
7:0	`tft`	Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.	RW	0x0

rxftlr

This register controls the threshold value for the receive FIFO memory. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0001C
spim1	0xFFF01000	0xFFF0101C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rft RW 0x0							

rxftlr Fields

Bit	Name	Description	Access	Reset
7:0	rft	Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.	RW	0x0

txflr

This register contains the number of valid data entries in the transmit FIFO memory. Ranges from 0 to 256.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00020
spim1	0xFFF01000	0xFFF01020

Offset: 0x20

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							txtfl RO 0x0								

txflr Fields

Bit	Name	Description	Access	Reset
8:0	`txtfl`	Contains the number of valid data entries in the transmit FIFO.	RO	0x0

rxflr

This register contains the number of valid data entries in the receive FIFO memory. This register can be read at any time. Ranges from 0 to 256.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00024
spim1	0xFFF01000	0xFFF01024

Offset: `0x24`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rxtfl RO 0x0								

rxflr Fields

Bit	Name	Description	Access	Reset
8:0	`rxtfl`	Contains the number of valid data entries in the receive FIFO.	RO	0x0

sr

This register is used to indicate the current transfer status, FIFO status, and any transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00028
spim1	0xFFF01000	0xFFF01028

Offset: `0x28`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rff RO 0x0	rfne RO 0x0	tfe RO 0x1	tfnf RO 0x1	busy RO 0x0

sr Fields

Bit	Name	Description	Access	Reset
4	rff	Reports receive FIFO condition. **Value** **Description** 0x0 Receive FIFO is not full 0x1 Receive FIFO is full	RO	0x0
3	rfne	Reports receive FIFO condition. **Value** **Description** 0x0 Receive FIFO is empty 0x1 Receive FIFO is not empty	RO	0x0
2	tfe	Reports transmit FIFO condition. **Value** **Description** 0x1 Transmit FIFO is empty 0x0 Transmit FIFO is not empty	RO	0x1
1	tfnf	Reports transmit FIFO condition. **Value** **Description** 0x0 Transmit FIFO is full 0x1 Transmit FIFO is not full	RO	0x1
0	busy	Reports the staus of a serial transfer **Value** **Description** 0x0 SPI Master is inactive (idle or disabled) 0x1 SPI Master is actively transferring data	RO	0x0

imr

This register masks or enables all interrupts generated by the SPI Master.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0002C
spim1	0xFFF01000	0xFFF0102C

Offset: 0x2C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rxfim RW 0x1	rxoim RW 0x1	rxuim RW 0x1	txoim RW 0x1	txeim RW 0x1

imr Fields

Bit	Name	Description	Access	Reset
4	rxfim	Full Mask **Value** **Description** 0x0 spi_rxf_intr interrupt is masked (disabled) 0x1 spi_rxf_intr interrupt is enabled	RW	0x1
3	rxoim	Overflow Mask. **Value** **Description** 0x0 spi_rxo_intr interrupt is masked (disabled) 0x1 spi_rxo_intr interrupt is enabled	RW	0x1
2	rxuim	Underflow Mask **Value** **Description** 0x0 spi_rxu_intr interrupt is masked (disabled) 0x1 spi_rxu_intr interrupt is enabled	RW	0x1
1	txoim	Overflow Mask **Value** **Description** 0x0 spi_txo_intr interrupt is masked (disabled) 0x1 spi_txo_intr interrupt is enabled	RW	0x1

Bit	Name	Description	Access	Reset
0	txeim	Empty mask. **Value** **Description** 0x0 spi_txe_intr interrupt is masked (disabled) 0x1 spi_txe_intr interrupt is enabled	RW	0x1

isr

This register reports the status of the SPI Master interrupts after they have been masked.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00030
spim1	0xFFF01000	0xFFF01030

Offset: 0x30

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										mstis RO 0x0	rxfis RO 0x0	rxois RO 0x0	rxuis RO 0x0	txois RO 0x0	txeis RO 0x0

isr Fields

Bit	Name	Description	Access	Reset
5	mstis	Multi-master contention status. **Value** **Description** 0x0 0 = ssi_mst_intr interrupt not active after masking 0x1 1 = ssi_mst_intr interrupt is active after masking	RO	0x0
4	rxfis	Full Status. **Value** **Description** 0x0 spi_rxf_intr interrupt is not active after masking 0x1 spi_rxf_intr interrupt is full after masking	RO	0x0

Bit	Name	Description	Access	Reset
3	rxois	Overflow Status. **Value** — **Description** 0x0 — spi_rxo_intr interrupt is not active after masking 0x1 — spi_rxo_intr interrupt is active after masking	RO	0x0
2	rxuis	Underflow Status. **Value** — **Description** 0x0 — spi_rxu_intr interrupt is not active after masking 0x1 — spi_rxu_intr interrupt is active after masking	RO	0x0
1	txois	Overflow Status. **Value** — **Description** 0x0 — spi_txo_intr interrupt is not active after masking 0x1 — spi_txo_intr interrupt is active after masking	RO	0x0
0	txeis	Empty status. **Value** — **Description** 0x0 — spi_txe_intr interrupt is not active after masking 0x1 — spi_txe_intr interrupt is active after masking	RO	0x0

risr

This register reports the status of the SPI Master interrupts prior to masking.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00034
spim1	0xFFF01000	0xFFF01034

Offset: 0x34

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rxfir RO 0x0	rxoir RO 0x0	rxuir RO 0x0	txoir RO 0x0	txeir RO 0x0

risr Fields

Bit	Name	Description	Access	Reset
4	rxfir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxf_intr interrupt is not active prior to masking 0x1 spi_rxf_intr interrupt is active prior to masking	RO	0x0
3	rxoir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxo_intr interrupt is not active prior to masking 0x1 spi_rxo_intr interrupt is active prior masking	RO	0x0
2	rxuir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxu_intr interrupt is not active prior to masking 0x1 spi_rxu_intr interrupt is active prior to masking	RO	0x0
1	txoir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_txo_intr interrupt is not active prior to masking 0x1 spi_txo_intr interrupt is active prior masking	RO	0x0

Bit	Name	Description	Access	Reset
0	txeir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_txe_intr interrupt is not active prior to masking 0x1 spi_txe_intr interrupt is active prior masking	RO	0x0

txoicr

Transmit FIFO Overflow Interrupt Clear Register

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00038
spim1	0xFFF01000	0xFFF01038

Offset: 0x38

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															txoicr RO 0x0

txoicr Fields

Bit	Name	Description	Access	Reset
0	txoicr	This register reflects the status of the interrupt. A read from this register clears the spi_txo_intr interrupt; writing has no effect.	RO	0x0

rxoicr

Receive FIFO Overflow Interrupt Clear Register

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0003C
spim1	0xFFF01000	0xFFF0103C

Offset: 0x3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															rxoicr RO 0x0

rxoicr Fields

Bit	Name	Description	Access	Reset
0	rxoicr	This register reflects the status of the interrupt. A read from this register clears the spi_rxo_intr interrupt; writing has no effect.	RO	0x0

rxuicr

Receive FIFO Underflow Interrupt Clear Register

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00040
spim1	0xFFF01000	0xFFF01040

Offset: 0x40

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															rxuicr RO 0x0

rxuicr Fields

Bit	Name	Description	Access	Reset
0	rxuicr	This register reflects the status of the interrupt. A read from this register clears the spi_rxu_intr interrupt; writing has no effect.	RO	0x0

icr

Clear Interrupt

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00048
spim1	0xFFF01000	0xFFF01048

Offset: 0x48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															icr RO 0x0

icr Fields

Bit	Name	Description	Access	Reset
0	icr	This register is set if any of the interrupts are active. A read clears the spi_txo_intr, spi_rxu_intr, spi_rxo_intr, and the spi_mst_intr interrupts. Writing to this register has no effect.	RO	0x0

dmacr

This register is used to enable the DMA Controller interface operation.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0004C
spim1	0xFFF01000	0xFFF0104C

Offset: 0x4C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														tdmae RW 0x0	rdmae RW 0x0

dmacr Fields

Bit	Name	Description	Access	Reset
1	`tdmae`	This bit enables/disables the transmit FIFO DMA channel. **Value** **Description** 0x0 Transmit DMA disabled 0x1 Transmit DMA enabled	RW	0x0
0	`rdmae`	This bit enables/disables the receive FIFO DMA channel. **Value** **Description** 0x0 Receive DMA disabled 0x1 Receive DMA enabled	RW	0x0

dmatdlr

Controls the FIFO Level for a DMA transmit request

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00050
spim1	0xFFF01000	0xFFF01050

Offset: `0x50`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dmatdl RW 0x0							

dmatdlr Fields

Bit	Name	Description	Access	Reset
7:0	`dmatdl`	This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.	RW	0x0

dmardlr

Controls the FIFO Level for a DMA receeive request

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00054
spim1	0xFFF01000	0xFFF01054

Offset: 0x54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dmardl RW 0x0							

dmardlr Fields

Bit	Name	Description	Access	Reset
7:0	dmardl	This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.	RW	0x0

idr

This register contains the peripherals identification code, which is 0x05510000.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00058
spim1	0xFFF01000	0xFFF01058

Offset: 0x58

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
idr RO 0x5510000															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
idr RO 0x5510000															

idr Fields

Bit	Name	Description	Access	Reset
31:0	idr	This register contains the peripherals identification code	RO	0x5510000

spi_version_id

Version ID Register value

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF0005C
spim1	0xFFF01000	0xFFF0105C

Offset: 0x5C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
spi_version_id RW 0x3332302A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
spi_version_id RW 0x3332302A															

spi_version_id Fields

Bit	Name	Description	Access	Reset
31:0	spi_version_id	Contains the hex representation of the Synopsys component version. Consists of ASCII value for each number in the version.	RW	0x3332302A

dr

This register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SPI_EN = 1. FIFOs are reset when SPI_EN = 0. The data register occupies 36 32-bit locations in the address map (0x60 to 0xec). These are all aliases for the same data register. This is done to support burst accesses.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF00060
spim1	0xFFF01000	0xFFF01060

Offset: 0x60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
dr RW 0x0															

dr Fields

Bit	Name	Description	Access	Reset
15:0	dr	When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer	RW	0x0

rx_sample_dly

This register controls the number of spi_m_clk cycles that are delayed (from the default sample time) before the actual sample of the rxd input occurs. It is impossible to write to this register when the SPI Master is enabled. The SPI Master is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spim0	0xFFF00000	0xFFF000FC
spim1	0xFFF01000	0xFFF010FC

Offset: 0xFC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									rsd RW 0x0						

rx_sample_dly Fields

Bit	Name	Description	Access	Reset
6:0	rsd	This register is used to delay the sample of the rxd input port. Each value represents a single spi_m_clk delay on the sample of rxd. Note; If this register is programmed with a value that exceeds 64, a 0 delay will be applied to the receive sample. The maximum delay is 64 spi_m_clk cycles.	RW	0x0

SPI Slave Module Address Map

Registers in the SPI Slave module

Module Instance	Base Address
spis0	0xFFE02000
spis1	0xFFE03000

SPI Slave Module

Register	Offset	Width	Acces s	Reset Value	Description
ctrlr0 on page 19-54	0x0	32	RW	0x7	Control Register 0
spienr on page 19-57	0x8	32	RW	0x0	Enable Register
mwcr on page 19-58	0xC	32	RW	0x0	Microwire Control Register
txftlr on page 19-59	0x18	32	RW	0x0	Transmit FIFO Threshold Level Register
rxftlr on page 19-60	0x1C	32	RW	0x0	Receive FIFO Threshold Level Register
txflr on page 19-61	0x20	32	RO	0x0	Transmit FIFO Level Register
rxflr on page 19-61	0x24	32	RO	0x0	Receive FIFO Level Register
sr on page 19-62	0x28	32	RO	0x6	Status Register
imr on page 19-63	0x2C	32	RW	0x1F	Interrupt Mask Register
isr on page 19-65	0x30	32	RO	0x0	Interrupt Status Register
risr on page 19-66	0x34	32	RO	0x0	Raw Interrupt Status Register
txoicr on page 19-67	0x38	32	RO	0x0	Transmit FIFO Overflow Interrupt Clear Register
rxoicr on page 19-68	0x3C	32	RO	0x0	Receive FIFO Overflow Interrupt Clear Register
rxuicr on page 19-69	0x40	32	RO	0x0	Receive FIFO Underflow Interrupt Clear Register
icr on page 19-69	0x48	32	RO	0x0	Interrupt Clear Register
dmacr on page 19-70	0x4C	32	RW	0x0	DMA Control Register
dmatdlr on page 19-71	0x50	32	RW	0x0	DMA Transmit Data Level Regoster
dmardlr on page 19-71	0x54	32	RW	0x0	DMA Receive Data Level Register
idr on page 19-72	0x58	32	RO	0x5510005	Identification Register
spi_version_id on page 19-72	0x5C	32	RW	0x3332302A	Component Version Register
dr on page 19-73	0x60	32	RW	0x0	Data Register

ctrlr0

This register controls the serial data transfer. It is impossible to write to this register when the SPI Slave is enabled. The SPI Slave is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02000
spis1	0xFFE03000	0xFFE03000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cfs RW 0x0				srl RW 0x0	slv_oe RW 0x0	tmod RW 0x0		scpol RW 0x0	scph RW 0x0	frf RW 0x0		dfs RW 0x7			

ctrlr0 Fields

Bit	Name	Description	Access	Reset
15:12	cfs	Selects the length of the control word for the Microwire frame format. The length (in bits) is the value of this field plus 1.	RW	0x0
11	srl	Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. **Value** **Description** 0x0 Normal Mode Operation 0x1 Test Mode Operation	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>10</td><td>slv_oe</td><td>This bit enables or disables the setting of the spis0_ssi_oe_n output from the SPI Slave. When SLV_OE = 1, the spis0_ssi_oe_n output can never be active. When the spis0_ssi_oe_n output controls the tri-state buffer on the txd output from the slave, a high impedance state is always present on the slave spis0_txd output when SLV_OE = 1. This is useful when the master transmits in broadcast mode (master transmits data to all slave devices). Only one slave may respond with data on the master spis0_rxd line. This bit is enabled after reset and must be disabled by software (when broadcast mode is used), if you do not want this device to respond with data.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Slave txd is enabled</td></tr>
<tr><td>0x1</td><td>Slave txd is disabled</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>9:8</td><td>tmod</td><td>Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. In transmit-only mode, data received from the external device is not valid and is not stored in the receive FIFO memory; it is overwritten on the next transfer. In receive-only mode, transmitted data are not valid. After the first write to the transmit FIFO, the same word is retransmitted for the duration of the transfer. In transmit-and-receive mode, both transmit and receive data are valid. The transfer continues until the transmit FIFO is empty. Data received from the external device are stored into the receive FIFO memory
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Transmit & and Receive</td></tr>
<tr><td>0x1</td><td>Transmit Only</td></tr>
<tr><td>0x2</td><td>Receive Only</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>7</td><td>scpol</td><td>Valid when the frame format (FRF) is set to Motorola SPI. Used to select the polarity of the inactive serial clock, which is held inactive when the spi master is not actively transferring data on the serial bus.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Serial clock toggles in middle of first data bit</td></tr>
<tr><td>0x1</td><td>Serial clock toggles at start of first data bit</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>6</td><td>scph</td><td>Valid when the frame format (FRF) is set to Motorola SPI. The serial clock phase selects the relationship of the serial clock with the slave select signal. When SCPH = 0, data are captured on the first edge of the serial clock. When SCPH = 1, the serial clock starts toggling one cycle after the slave select line is activated, and data are captured on the second edge of the serial clock.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Inactive state of serial clock is low</td></tr>
<tr><td>0x1</td><td>Inactive state of serial clock is high</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>5:4</td><td>frf</td><td>Selects which serial protocol transfers the data.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Motorola SPI</td></tr>
<tr><td>0x1</td><td>Texas instruments SSP</td></tr>
<tr><td>0x2</td><td>National Semi Microwire</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>3:0</td><td>dfs</td><td>Selects the data frame length. When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receiver FIFO zero-padded. You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x3</td><td>4-bit serial data transfer</td></tr>
<tr><td>0x4</td><td>5-bit serial data transfer</td></tr>
<tr><td>0x5</td><td>6-bit serial data transfer</td></tr>
<tr><td>0x6</td><td>7-bit serial data transfer</td></tr>
<tr><td>0x7</td><td>8-bit serial data transfer</td></tr>
<tr><td>0x8</td><td>9-bit serial data transfer</td></tr>
<tr><td>0x9</td><td>10-bit serial data transfer</td></tr>
</table></td><td>RW</td><td>0x7</td></tr>
</table>

spienr

Enables and disables all SPI operations.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02008
spis1	0xFFE03000	0xFFE03008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															spi_en RW 0x0

spienr Fields

Bit	Name	Description	Access	Reset
0	spi_en	When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the SPI Slave control registers when enabled. **Value** **Description** 0x0 Disables serial transfer operations 0x1 Enables serial transfer operations	RW	0x0

mwcr

This register controls the direction of the data word for the half-duplex Microwire serial protocol. It is impossible to write to this register when the SPI Slave is enabled. The SPI Slave is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE0200C
spis1	0xFFE03000	0xFFE0300C

Offset: 0xC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														mdd RW 0x0	mwmod RW 0x0

mwcr Fields

Bit	Name	Description	Access	Reset
1	mdd	Defines the direction of the data word when the Microwire serial protocol is used. **Value** **Description** 0x0 SPI Slave receives data 0x1 SPI Slave transmits data	RW	0x0
0	mwmod	Defines whether the Microwire transfer is sequential or non-sequential. When sequential mode is used, only one control word is needed to transmit or receive a block of data words. When non-sequential mode is used, there must be a control word for each data word that is transmitted or received. **Value** **Description** 0x0 non-sequential transfer 0x1 sequential transfer	RW	0x0

txftlr

This register controls the threshold value for the transmit FIFO memory. It is impossible to write to this register when the SPI Slave is enabled. The SPI Slave is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02018
spis1	0xFFE03000	0xFFE03018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								tft RW 0x0							

txftlr Fields

Bit	Name	Description	Access	Reset
7:0	tft	Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt. When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.	RW	0x0

rxftlr

This register controls the threshold value for the receive FIFO memory. It is impossible to write to this register when the SPI Slave is enabled. The SPI Slave is enabled and disabled by writing to the SPIENR register.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE0201C
spis1	0xFFE03000	0xFFE0301C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rft RW 0x0							

rxftlr Fields

Bit	Name	Description	Access	Reset
7:0	rft	Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt. When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.	RW	0x0

txflr

This register contains the number of valid data entries in the transmit FIFO memory. Ranges from 0 to 256.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02020
spis1	0xFFE03000	0xFFE03020

Offset: 0x20

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							txtfl RO 0x0								

txflr Fields

Bit	Name	Description	Access	Reset
8:0	txtfl	Contains the number of valid data entries in the transmit FIFO.	RO	0x0

rxflr

This register contains the number of valid data entriesin the receive FIFO memory. This register can be read at any time. Ranges from 0 to 256.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02024
spis1	0xFFE03000	0xFFE03024

Offset: 0x24

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							rxtfl RO 0x0								

rxflr Fields

Bit	Name	Description	Access	Reset
8:0	rxtfl	Contains the number of valid data entries in the receive FIFO.	RO	0x0

sr

Reports FIFO transfer status, and any transmission/reception errors that may have occurred. The status register may be read at any time. None of the bits in this register request an interrupt.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02028
spis1	0xFFE03000	0xFFE03028

Offset: 0x28

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										txe RO 0x0	rff RO 0x0	rfne RO 0x0	tfe RO 0x1	tfnf RO 0x1	busy RO 0x0

sr Fields

Bit	Name	Description	Access	Reset
5	txe	Data from the previous transmission is resent on the txd line. This bit is cleared when read. **Value** — **Description** 0x0 — No Error 0x1 — Transmission Error	RO	0x0
4	rff	Reports the status of receive FIFO Full **Value** — **Description** 0x0 — Receive FIFO is not full 0x1 — Receive FIFO is full	RO	0x0

Bit	Name	Description	Access	Reset
3	`rfne`	Reports the status of receive FIFO empty. **Value** **Description** 0x0 Receive FIFO is empty 0x1 Receive FIFO is not empty	`RO`	`0x0`
2	`tfe`	Reports the status of transmit FIFO empty. This bit field does not request an interrupt. **Value** **Description** 0x1 Transmit FIFO is empty 0x0 Transmit FIFO is not empty	`RO`	`0x1`
1	`tfnf`	Reports the status of the transmit FIFO. **Value** **Description** 0x0 Transmit FIFO is full 0x1 Transmit FIFO is not full	`RO`	`0x1`
0	`busy`	Reports the status of a serial transfer **Value** **Description** 0x0 SPI Slave is inactive (idle or disabled) 0x1 SPI Slave is actively transferring data	`RO`	`0x0`

imr

This register masks or enables all interrupts generated by the SPI Slave.

Module Instance	Base Address	Register Address
`spis0`	`0xFFE02000`	`0xFFE0202C`
`spis1`	`0xFFE03000`	`0xFFE0302C`

Offset: `0x2C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rxfim RW 0x1	rxoim RW 0x1	rxuim RW 0x1	txoim RW 0x1	txeim RW 0x1

imr Fields

Bit	Name	Description	Access	Reset
4	rxfim	FIFO Full Mask. **Value** **Description** 0x0 spi_rxf_intr interrupt is masked (disabled) 0x1 spi_rxf_intr interrupt is enabled	RW	0x1
3	rxoim	Overflow Mask. **Value** **Description** 0x0 spi_rxo_intr interrupt is masked (disabled) 0x1 spi_rxo_intr interrupt is enabled	RW	0x1
2	rxuim	Underfow Mask **Value** **Description** 0x0 spi_rxu_intr interrupt is masked (disabled) 0x1 spi_rxu_intr interrupt is enabled	RW	0x1
1	txoim	Overflow mask. **Value** **Description** 0x0 spi_txo_intr interrupt is masked (disabled) 0x1 spi_txo_intr interrupt is enabled	RW	0x1
0	txeim	Empty mask. **Value** **Description** 0x0 spi_txe_intr interrupt is masked (disabled) 0x1 spi_txe_intr interrupt is enabled	RW	0x1

isr

This register reports the status of the SPI Slave interrupts after they have been masked.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02030
spis1	0xFFE03000	0xFFE03030

Offset: 0x30

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rxfis RO 0x0	rxois RO 0x0	rxuis RO 0x0	txois RO 0x0	txeis RO 0x0

isr Fields

Bit	Name	Description	Access	Reset
4	rxfis	Full Status **Value** **Description** 0x0 spi_rxf_intr interrupt is not active after masking 0x1 spi_rxf_intr interrupt is full after masking	RO	0x0
3	rxois	Overflow Status. **Value** **Description** 0x0 spi_rxo_intr interrupt is not active after masking 0x1 spi_rxo_intr interrupt is active after masking	RO	0x0
2	rxuis	Underflow Status. **Value** **Description** 0x0 spi_rxu_intr interrupt is not active after masking 0x1 spi_rxu_intr interrupt is active after masking	RO	0x0

Bit	Name	Description	Access	Reset
1	`txois`	Overflow Status. **Value** **Description** 0x0 spi_txo_intr interrupt is not active after masking 0x1 spi_txo_intr interrupt is active after masking	RO	0x0
0	`txeis`	Empty Status. **Value** **Description** 0x0 spi_txe_intr interrupt is not active after masking 0x1 spi_txe_intr interrupt is active after masking	RO	0x0

risr

This register reports the status of the SPI Slave interrupts prior to masking.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02034
spis1	0xFFE03000	0xFFE03034

Offset: 0x34

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rxfir RO 0x0	rxoir RO 0x0	rxuir RO 0x0	txoir RO 0x0	txeir RO 0x0

risr Fields

Bit	Name	Description	Access	Reset
4	rxfir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxf_intr interrupt is not active prior to masking 0x1 spi_rxf_intr interrupt is active prior to masking	RO	0x0
3	rxoir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxo_intr interrupt is not active prior to masking 0x1 spi_rxo_intr interrupt is active prior masking	RO	0x0
2	rxuir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_rxu_intr interrupt is not active prior to masking 0x1 spi_rxu_intr interrupt is active prior to masking	RO	0x0
1	txoir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_txo_intr interrupt is not active prior to masking 0x1 spi_txo_intr interrupt is active prior masking	RO	0x0
0	txeir	The interrupt is active or inactive prior to masking. **Value** **Description** 0x0 spi_txe_intr interrupt is not active prior to masking 0x1 spi_txe_intr interrupt is active prior masking	RO	0x0

txoicr

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02038
spis1	0xFFE03000	0xFFE03038

Offset: 0x38

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															txoicr RO 0x0

txoicr Fields

Bit	Name	Description	Access	Reset
0	txoicr	This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.	RO	0x0

rxoicr

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE0203C
spis1	0xFFE03000	0xFFE0303C

Offset: 0x3C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															rxoicr RO 0x0

rxoicr Fields

Bit	Name	Description	Access	Reset
0	rxoicr	This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.	RO	0x0

rxuicr

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02040
spis1	0xFFE03000	0xFFE03040

Offset: 0x40

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															rxuicr RO 0x0

rxuicr Fields

Bit	Name	Description	Access	Reset
0	rxuicr	This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect.	RO	0x0

icr

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02048
spis1	0xFFE03000	0xFFE03048

Offset: 0x48

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															icr RO 0x0

icr Fields

Bit	Name	Description	Access	Reset
0	`icr`	This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.	RO	0x0

dmacr

The register is used to enable the DMA Controller interface operation.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE0204C
spis1	0xFFE03000	0xFFE0304C

Offset: `0x4C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														tdmae RW 0x0	rdmae RW 0x0

dmacr Fields

Bit	Name	Description	Access	Reset
1	`tdmae`	This bit enables/disables the transmit FIFO DMA channel. **Value** / **Description** 0x0 Transmit DMA disabled 0x1 Transmit DMA enabled	RW	0x0

Bit	Name	Description	Access	Reset
0	rdmae	This bit enables/disables the receive FIFO DMA channel **Value** **Description** 0x0 Receive DMA disabled 0x1 Receive DMA enabled	RW	0x0

dmatdlr

Controls DMA Transmit FIFO Threshold

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02050
spis1	0xFFE03000	0xFFE03050

Offset: 0x50

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dmatdl RW 0x0							

dmatdlr Fields

Bit	Name	Description	Access	Reset
7:0	dmatdl	This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.	RW	0x0

dmardlr

Controls DMA Receive FIFO Threshold

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02054
spis1	0xFFE03000	0xFFE03054

Offset: 0x54

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dmardl RW 0x0							

dmardlr Fields

Bit	Name	Description	Access	Reset
7:0	dmardl	This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.	RW	0x0

idr

This register stores a peripheral identification code

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02058
spis1	0xFFE03000	0xFFE03058

Offset: 0x58

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
idr RO 0x5510005															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
idr RO 0x5510005															

idr Fields

Bit	Name	Description	Access	Reset
31:0	idr	This filed contains the peripherals identification code, 0x05510005.	RO	0x55100 05

spi_version_id

This read-only register stores the specific SPI Slave component version.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE0205C

Module Instance	Base Address	Register Address
spis1	0xFFE03000	0xFFE0305C

Offset: 0x5C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">spi_version_id
RW 0x3332302A</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">spi_version_id
RW 0x3332302A</td></tr>
</table>

spi_version_id Fields

Bit	Name	Description	Access	Reset
31:0	spi_version_id	Contains the hex representation of the Synopsys component version. Consists of ASCII value for each number in the version.	RW	0x33323 02A

dr

The data register is a 16-bit read/write buffer for the transmit/receive FIFOs. When the register is read, data in the receive FIFO buffer is accessed. When it is written to, data are moved into the transmit FIFO buffer; a write can occur only when SPI_EN = 1. FIFOs are reset when SPI_EN = 0.

Module Instance	Base Address	Register Address
spis0	0xFFE02000	0xFFE02060
spis1	0xFFE03000	0xFFE03060

Offset: 0x60

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">dr
RW 0x0</td></tr>
</table>

dr Fields

Bit	Name	Description	Access	Reset
15:0	dr	When writing to this register, you must right-justify the data. Read data are automatically right-justified. Read = Receive FIFO buffer Write = Transmit FIFO buffer	RW	0x0

Document Revision History

Table 19-6: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• "Glue Logic for Master Port ss_in_n" section added • Interface Pins topic added • FPGA Routing topic added • Added address aap and register descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting updates
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model, address map and register definitions, clocks, and reset sections.
January 2012	1.0	Initial release.

I^{2}C Controller 20

2014.06.30

cv_54020 Subscribe Send Feedback

The I^{2}C controller provides support for a communication link between integrated circuits on a board. It is a simple two-wire bus which consists of a serial data line (SDA) and a serial clock (SCL) for use in applications such as temperature sensors and voltage level translators to EEPROMs, A/D and D/A converters, CODECs, and many types of microprocessors. †

The hard processor system (HPS) provides four I^{2}C controllers to enable system software to communicate serially with I^{2}C buses. Each I^{2}C controller can operate in master or slave mode, and support standard mode of up to 100 Kbps or fast mode of up to 400 Kbps. These I^{2}C controllers are instances of the Synopsys® DesignWare® APB I^{2}C (DW_apb_i2c) controller.

Each I^{2}C controller must be programmed to operate in either master or slave mode only. Operating as a master and slave simultaneously is not supported. † [43]

Features of the I^{2}C Controller

The I^{2}C controller has the following features:

- Maximum clock speed of up to 400 Kbps
- One of the following I^{2}C operations:
  - A master in an I^{2}C system and programmed only as a master †
  - A slave in an I^{2}C system and programmed only as a slave †
- 7- or 10-bit addressing †
- Mixed read and write combined-format transactions in both 7-bit and 10-bit addressing mode †
- Bulk transmit mode †
- Transmit and receive buffers †
- Handles bit and byte waiting at all bus speeds †
- DMA handshaking interface †

[43] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

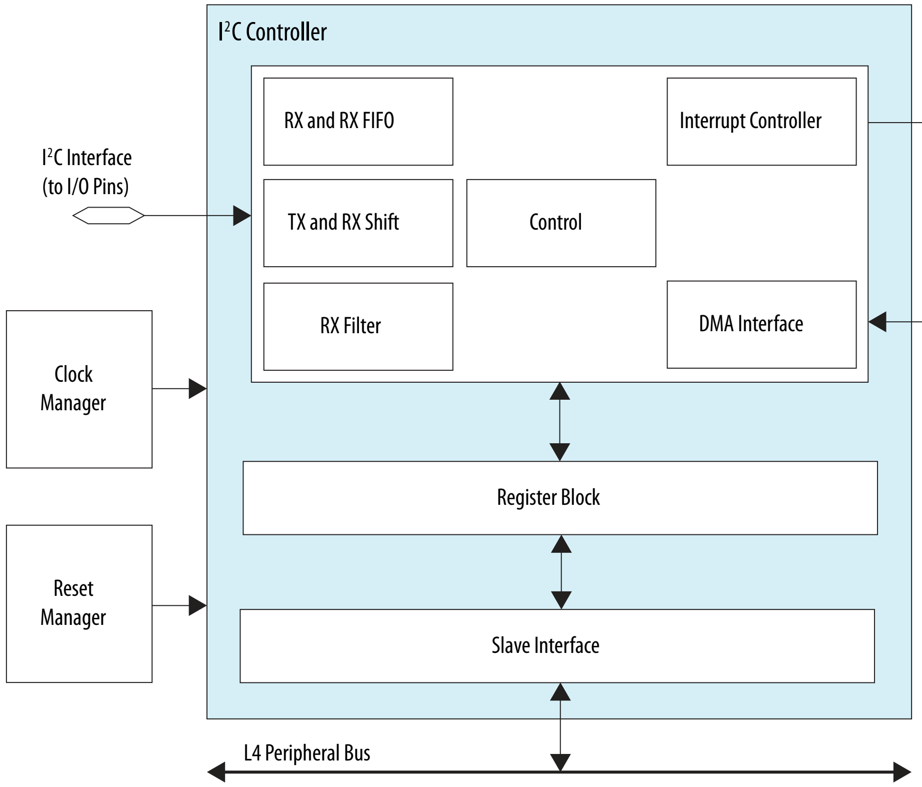
†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

I^{2}C Controller Block Diagram and System Integration

The I^{2}C controller consists of an internal slave interface, an I^{2}C interface, and FIFO logic to buffer data between the two interfaces. †

The host processor accesses data, control, and status information about the I^{2}C controller through a 32-bit slave interface.

Figure 20-1: I^{2}C Controller Block Diagram



The I^{2}C controller consists of the following modules and interfaces:

- Slave interface for control and status register (CSR) accesses and DMA transfers, allowing a master to access the CSRs and the DMA to read or write data directly.
- Two FIFO buffers for transmit and receive data, which hold the Rx FIFO and Tx FIFO buffer register banks and controllers, along with their status levels. †
- Shift logic for parallel-to-serial and serial-to-parallel conversion
    - Rx shift – Receives data into the design and extracts it in byte format. †
    - Tx shift – Presents data supplied by CPU for transfer on the I^{2}C bus. †
- Control logic responsible for implementing the I^{2}C protocol.

- DMA interface that generates handshaking signals to the DMA controller in order to automate the data transfer without CPU intervention. †
- Interrupt controller that generates raw interrupts and interrupt flags, allowing them to be set and cleared. †
- Receive filter for detecting events, such as start and stop conditions, in the bus; for example, start, stop and arbitration lost. †

Functional Description of the I^{2}C Controller

Feature Usage

The I^{2}C controller can operate in standard mode (with data rates of up to 100 Kbps) or fast mode (with data rates less than or equal to 400 Kbps). Additionally, fast mode devices are downward compatible. For instance, fast mode devices can communicate with standard mode devices in 0 to 100 Kbps I^{2}C bus system. However, standard mode devices are not upward compatible and should not be incorporated in a fast-mode I^{2}C bus system as they cannot follow the higher transfer rate and therefore unpredictable states would occur. †

You can attach any I^{2}C controller to an I^{2}C-bus and every device can talk with any master, passing information back and forth. There needs to be at least one master (such as a microcontroller or DSP) on the bus and there can be multiple masters, which require them to arbitrate for ownership. Multiple masters and arbitration are explained later in this chapter. †

Behavior

You can control the I^{2}C controller via software to be in either mode:

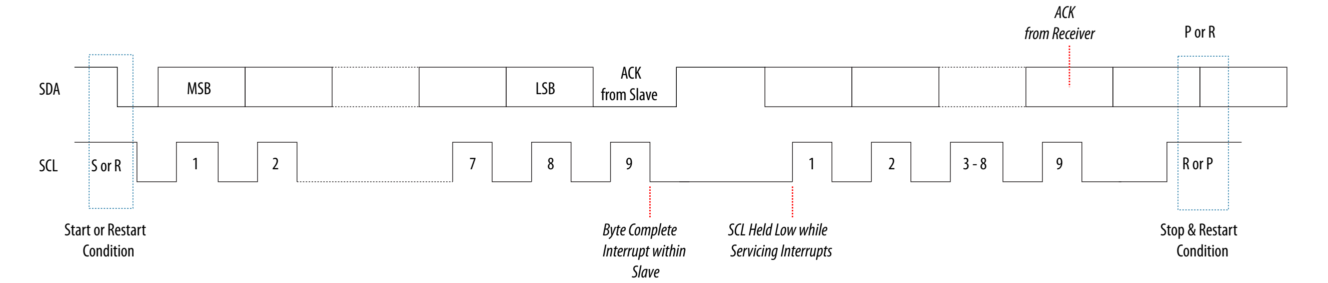
- An I^{2}C master only, communicating with other I^{2}C slaves.
- An I^{2}C slave only, communicating with one or more I^{2}C masters.

The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. The acknowledgement of data is sent by the device that is receiving data, which can be either a master or a slave. As mentioned previously, the I^{2}C protocol also allows multiple masters to reside on the I^{2}C bus and uses an arbitration procedure to determine bus ownership. †

Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address. †

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver receives one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with an ACK pulse. This transaction continues until the master terminates the transmission by not acknowledging (NACK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. †

Figure 20-2: Data transfer on the I^{2}C Bus †



The I^{2}C controller is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. The maximum number of devices on the bus is limited by only the maximum capacitance specification of 400 pF. Data is transmitted in byte packages. †

START and STOP Generation

When operating as a master, putting data into the transmit FIFO causes the I^{2}C controller to generate a START condition on the I^{2}C bus. In order for the master to complete the transfer and issue a STOP condition it must find a transmit FIFO entry tagged with a stop bit. Allowing the transmit FIFO to empty without a stop bit set, the master will stall the transfer by holding the SCL line low. †

When operating as a slave, the I^{2}C controller does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the I^{2}C controller, it holds the SCL line low until read data has been supplied to it. This stalls the I^{2}C bus until read data is provided to the slave I^{2}C controller, or the I^{2}C controller slave is disabled by writing a 0 to bit 0 of `IC_ENABLE` register. †

Combined Formats

The I^{2}C controller supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes. †

The I^{2}C controller does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions. †

To initiate combined format transfers, the `IC_RESTART_EN` bit in the `IC_CON` register should be set to 1. With this value set and operating as a master, when the I^{2}C controller completes an I^{2}C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the `IC_RESTART_EN` is 0, a STOP will be issued followed by a START condition. Another way to generate the RESTART condition is to set the Restart bit [10] of the `DATA_CMD` register. Regardless if the direction of the transfer changes or not the RESTART condition will be generated.†

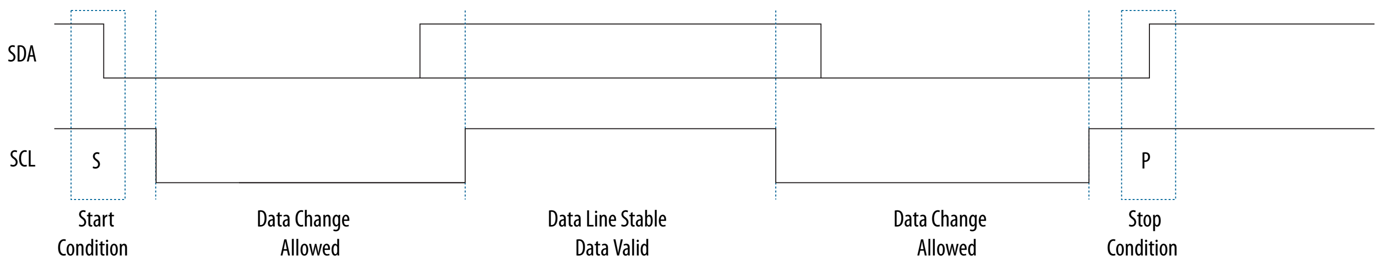
Protocol Details

START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through pull-up resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. †

The following figure shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1. †

Figure 20-3: Timing Diagram for START and STOP Conditions



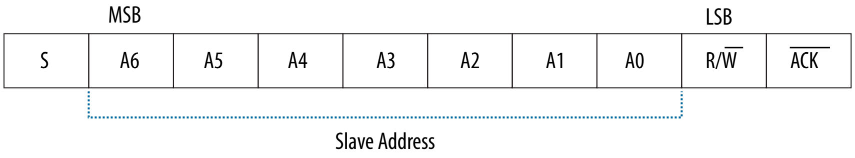
The signal transitions for the START or STOP condition, as shown in the figure, reflect those observed at the output signals of the master driving the I^{2}C bus. Care should be taken when observing the SDA or SCL signals at the input signals of the slave(s), because unequal line delays may result in an incorrect SDA or SCL timing relationship. †

Addressing Slave Protocol

7-Bit Address Format

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in the following figure. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave. †

Figure 20-4: 7- Bit Address Format

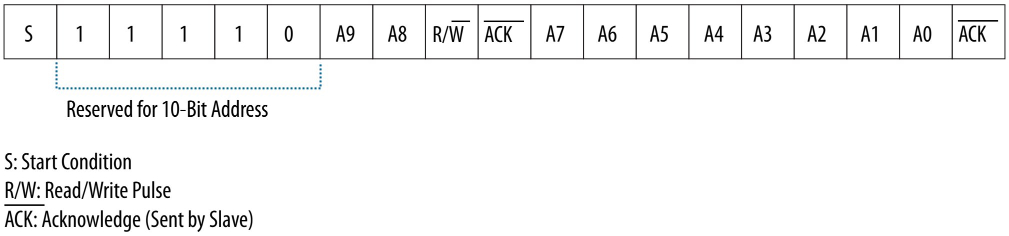


S: Start Condition
$R/\overline{W}$: Read/Write Pulse
$\overline{ACK}$: Acknowledge (Sent by Slave)

10-Bit Address Format

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. †

Figure 20-5: 10-Bit Address Format



The following table defines the special purpose and reserved first byte addresses. †

Table 20-1: I^2C Definition of Bits in First Byte

Slave Address	R/W Bit	Description
0000 000	0	General call address. The I^2C controller places the data in the receive buffer and issues a general call interrupt.
0000 000	1	START byte. For more details, refer to “START BYTE Transfer Protocol”
0000 001	X	CBUS address. The I^2C controller ignores these accesses.
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	Unused
1111 1XX	X	Reserved
1111 0XX	X	10-bit slave addressing.

Note to Table: ‘X’ indicates do not care.

Related Information

START BYTE Transfer Protocol on page 20-8

Transmitting and Receiving Protocol

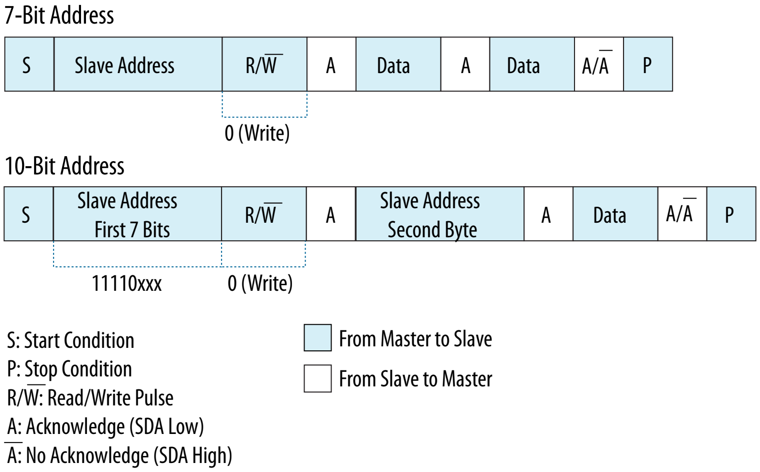
The master can initiate data transmission and reception to or from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to or from the bus, acting as either a slave-transmitter or slave-receiver, respectively. †

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer. †

If the master-transmitter is transmitting data as shown in the following figure, then the slave-receiver responds to the master-transmitter with an ACK pulse after every byte of data is received. †

Figure 20-6: Master-Transmitter Protocol †



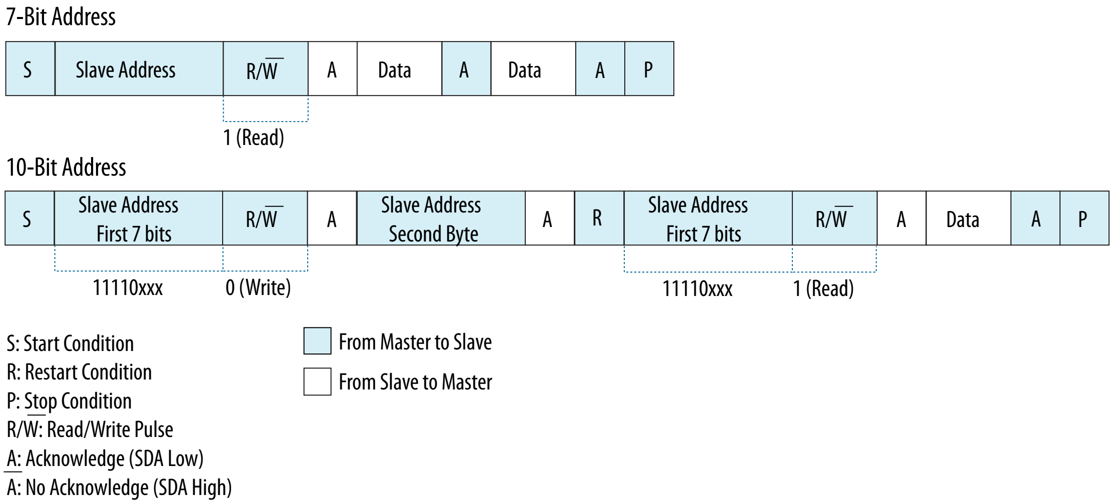
Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in the following figure, then the master responds to the slave-transmitter with an ACK pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) bit so that the master can issue a STOP condition. †

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode, the I^{2}C controller can then communicate with the same slave using a transfer of a different direction. For a description of the combined format transactions that the I^{2}C controller supports, refer to “Combined Formats” section of this chapter. †

Note: The I^{2}C controller must be inactive on the serial port (`I2C_DYNAMIC_TAR_UPDATE` = 1) before the target slave address register, `IC_TAR` can be reprogrammed. †

Figure 20-7: Master-Receiver Protocol †



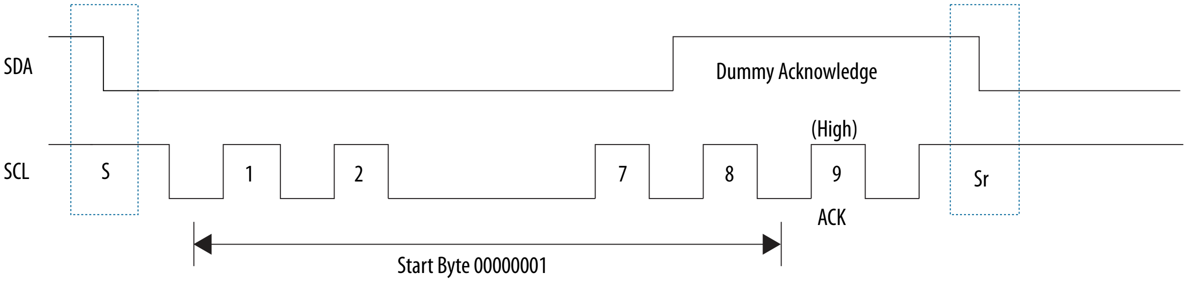
Related Information

Combined Formats on page 20-4

START BYTE Transfer Protocol

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I^{2}C hardware module. When the I^{2}C controller is set as a slave, it always samples the I^{2}C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I^{2}C controller is set as a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it. This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in the following figure. This allows the processor that is polling the bus to under-sample the address phase until the microcontroller detects a 0. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master. †

Figure 20-8: START BYTE Transfer †



The START BYTE has the following procedure: †

1. Master generates a START condition. †
2. Master transmits the START byte (0000 0001). †
3. Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus) †
4. No slave sets the ACK signal to 0. †
5. Master generates a RESTART (R) condition. †

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated. †

Multiple Master Arbitration

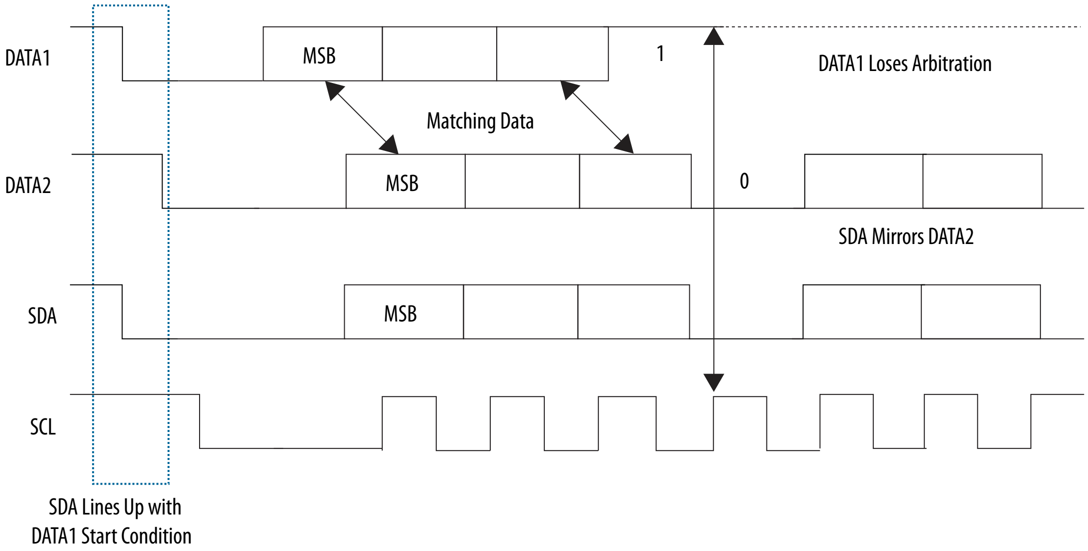
The I^{2}C controller bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I^{2}C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by simultaneously generating a START condition. Once a master (for example, a microcontroller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state. †

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase. †

Upon detecting that it has lost arbitration to another master, the I^{2}C controller stops generating SCL. †

The following figure illustrates the timing of two masters arbitrating on the bus.

Figure 20-9: Multiple Master Arbitration †



The bus control is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus. †

Arbitration is not allowed between the following conditions: †

- A RESTART condition and a data bit †
- A STOP condition and a data bit †
- A RESTART condition and a STOP condition †

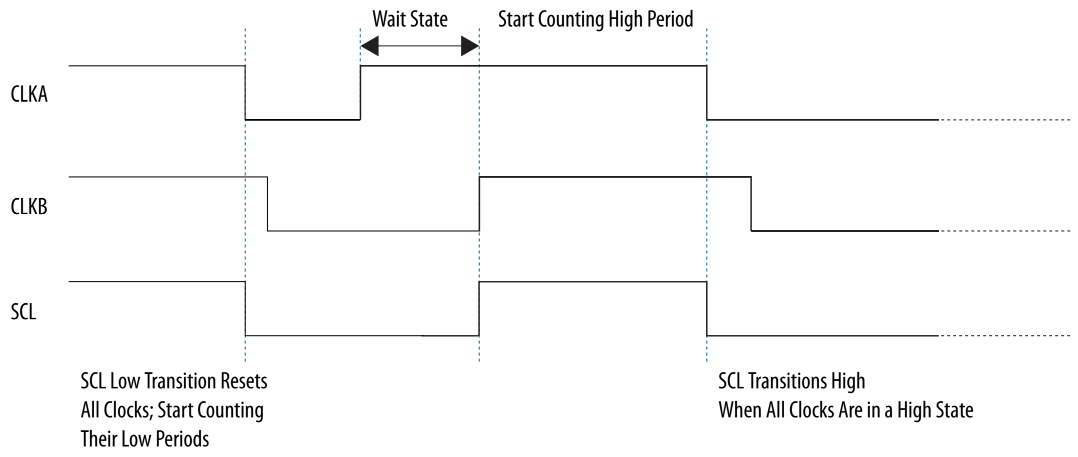
Slaves are not involved in the arbitration process. †

Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1. †

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in the following figure. Optionally, slaves may hold the SCL line low to slow down the timing on the I^{2}C bus. †

Figure 20-10: Multiple Master Clock Synchronization †


Wait State
Start Counting High Period
CLKA
CLKB
SCL
SCL Low Transition Resets All Clocks; Start Counting Their Low Periods
SCL Transitions High When All Clocks Are in a High State

Clock Frequency Configuration

When you configure the I^{2}C controller as a master, the SCL count registers must be set before any I^{2}C bus transaction can take place in order to ensure proper I/O timing. † There are four SCL count registers:

- Standard speed I^{2}C clock SCL high count, `IC_SS_SCL_HCNT` †
- Standard speed I^{2}C clock SCL low count, `IC_SS_SCL_LCNT` †
- Fast speed I^{2}C clock SCL high count, `IC_FS_SCL_HCNT` †
- Fast speed I^{2}C clock SCL low count, `IC_FS_SCL_LCNT` †

It is not necessary to program any of the SCL count registers if the I^{2}C controller is enabled to operate only as an I^{2}C slave, since these registers are used only to determine the SCL timing requirements for operation as an I^{2}C master. †

Minimum High and Low Counts

When the I^2C controller operates as an I^2C master in both transmit and receive transfers, the minimum value that can be programmed in the SCL low count registers is 8 while the minimum value allowed for the SCL high count registers is 6. †

The minimum value of 8 for the low count registers is due to the time required for the I^2C controller to drive SDA after a negative edge of SCL. The minimum value of 6 for the high count register is due to the time required for the I^2C controller to sample SDA during the high period of SCL. †

The I^2C controller adds one cycle to the low count register values in order to generate the low period of the SCL clock.

The I^2C controller adds seven cycles to the high count register values in order to generate the high period of the SCL clock. This is due to the following factors: †

- The digital filtering applied to the SCL line incurs a delay of four `l4_sp_clk` cycles. This filtering includes metastability removal and a 2-out-of-3 majority vote processing on SDA and SCL edges. †
- Whenever SCL is driven 1 to 0 by the I^2C controller—that is, completing the SCL high time—an internal logic latency of three `l4_sp_clk` cycles incurs. †

Consequently, the minimum SCL low time of which the I^2C controller is capable is nine (9) `l4_sp_clk` periods (8+1), while the minimum SCL high time is thirteen (13) `l4_sp_clk` periods (6+1+3+3). †

Calculating High and Low Counts

The calculations below show an example of how to calculate SCL high and low counts for each speed mode in the I^2C controller.

The equation to calculate the proper number of `l4_sp_clk` clock pulses required for setting the proper SCL clocks high and low times is as follows: †

Table 20-2: Equation

IC_HCNT = ceil(MIN_SCL_HIGHtime*OSCFREQ)
IC_LCNT = ceil(MIN_SCL_LOWtime*OSCFREQ)
MIN_SCL_HIGHtime = minimum high period
MIN_SCL_HIGHtime =
4000 ns for 100 kbps
600 ns for 400 kbps
60 ns for 3.4 Mbs, bus loading = 100pF
160 ns for 3.4 Mbs, bus loading = 400pF
MIN_SCL_LOWtime = minimum low period
MIN_SCL_LOWtime =
4700 ns for 100 kbps
1300 ns for 400 kbps
120 ns for 3.4Mbs, bus loading = 100pF
320 ns for 3.4Mbs, bus loading = 400pF
OSCFREQ = l4_sp_clk clock frequency (Hz)
For example:
OSCFREQ = 100 MHz
I2Cmode = fast, 400 kbps
MIN_SCL_HIGHtime = 600 ns
MIN_SCL_LOWtime = 1300 ns
IC_HCNT = ceil(600 ns * 100 MHz) IC_HCNTSCL PERIOD = 60
IC_LCNT = ceil(1300 ns * 100 MHz) IC_LCNTSCL PERIOD = 130
Actual MIN_SCL_HIGHtime = 60*(1/100 MHz) = 600 ns
Actual MIN_SCL_LOWtime = 130*(1/100 MHz) = 1300 ns †

SDA Hold Time

The I^{2}C protocol specification requires 300 ns of hold time on the SDA signal in standard and fast speed modes. Board delays on the SCL and SDA signals can mean that the hold time requirement is met at the I^{2}C master, but not at the I^{2}C slave (or vice-versa). As each application encounters differing board delays, the I^{2}C controller contains a software programmable register, IC_SDA_HOLD, to enable dynamic adjustment of the SDA hold time. †

DMA Controller Interface

The I^{2}C controller supports DMA signaling to indicate when data is ready to be read or when the transmit FIFO needs data. This support requires 2 DMA channels, one for transmit data and one for receive data. The I^{2}C controller supports both single and burst DMA transfers. System software can choose the DMA burst mode by programming an appropriate value into the threshold registers. The recommended setting of the FIFO threshold register value is half full.

To enable the DMA controller interface on the I^{2}C controller, you must write to the DMA control register (DMACR) bits. Writing a 1 into the TDMAE bit field of DMACR register enables the I^{2}C controller transmit handshaking interface. Writing a 1 into the RDMAE bit field of the DMACR register enables the I^{2}C controller receive handshaking interface. †

Related Information

DMA Controller on page 16-1
For details about the DMA burst length microcode setup, refer to the *DMA controller* chapter.

Clocks

Each I^{2}C controller is connected to the `l4_sp_clk` clock, which clocks transfers in standard and fast mode. The clock input is driven by the clock manager.

Related Information

Clock Manager on page 2-1
For more information, refer to *Clock Manager* chapter.

Resets

Each I^{2}C controller has a separate reset signal. The reset manager drives the signals on a cold or warm reset.

Related Information

Reset Manager on page 3-1
For more information, refer to *Reset Manager* chapter.

Interface Pins

All instances of the I^{2}C controller connect to external pins through pin multiplexers. Pin multiplexing allows all instances to function simultaneously and independently. The pins must be connected to a pull-up resistors and the I^{2}C bus capacitance cannot exceed 400 pF.

Table 20-3: I^{2}C Controller Interface Pins

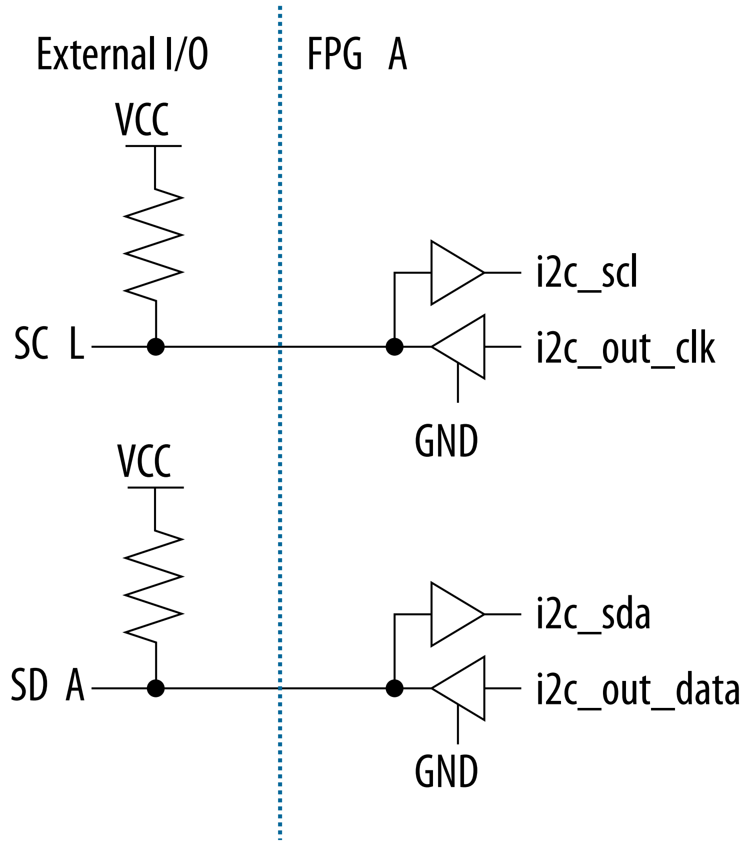
Pin Name	Signal Width	Direction	Description
SCL	1 bit	Bidirectional	Serial clock
SDA	1 bit	Bidirectional	Serial data

Table 20-4: HPS I^{2}C Signals for FPGA Routing

Signal Name	Signal Width	Direction	Description
i2c<#>_scl	1 bit	Input	Incoming I^{2}C clock source. This is the input SCL signal

Signal Name	Signal Width	Direction	Description
i2c<#>_out_clk	1 bit	Output	Outgoing I^2C clock enable. Output SCL signal. This signal is logically inverted and is synchronous to the HPS peripheral clock
i2c<#>_sda	1 bit	Input	Incoming I^2C data. This is the input SDA signal.
i2c<#>_out_data	1 bit	Output	Outgoing I^2C data enable. Output SDA signal. This signal is logically inverted and is synchronous to the HPS peripheral clock.

Figure 20-11: I^2C interface in FPGA fabric



The figure above shows the typical connection on the I^2C interface in FPGA fabric with `alt_iobuff`.

For both I2C clock and data, external IO pins are open drain connection. When output enables `i2c_out_data` and `i2c_out_clk` are asserted, external signal will be driven to ground.

Related Information

- **Instantiating the HPS Component** on page 27-1
- **I/O Buffer (ALTIOBUF)**
  For more information on configuring open drain I/O buffer to connect I²C signals to external I/O pins, please refer to Altera I/O Buffer (ALTIOBUF) Megafunction User Guide.

I²C Controller Programming Model

This section describes the programming model for the I²C controllers based on the two master and slave operation modes. †

Note: Each I²C controller should be set to operate only as an I²C master or as an I²C slave, never set both simultaneously. Ensure that bit 6 (`IC_SLAVE_DISABLE`) and 0 (`IC_MASTER_MODE`) of the `IC_CON` register are never set to 0 and 1, respectively. †

Slave Mode Operation

Initial Configuration

To use the I²C controller as a slave, perform the following steps: †

1. Disable the I²C controller by writing a 0 to bit 0 of the `IC_ENABLE` register. †
2. Write to the `IC_SAR` register (bits 9:0) to set the slave address. This is the address to which the I²C controller responds. †

   Note: The reset value for the I²C controller slave address is 0x55. If you are using 0x55 as the slave address, you can safely skip this step.
3. Write to the `IC_CON` register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I²C controller in slave-only mode by writing a 0 into bit 6 (`IC_SLAVE_DISABLE`) and a 0 to bit 0 (`MASTER_MODE`). †

   Note: Slaves and masters do not have to be programmed with the same type of addressing 7- or 10-bit address. For instance, a slave can be programmed with 7-bit addressing and a master with 10-bit addressing, and vice versa. †
4. Enable the I²C controller by writing a 1 in bit 0 of the `IC_ENABLE` register. †

Slave-Transmitter Operation for a Single Byte

When another I²C master device on the bus addresses the I²C controller and requests data, the I²C controller acts as a slave-transmitter and the following steps occur: †

1. The other I²C master device initiates an I²C transfer with an address that matches the slave address in the `IC_SAR` register of the I²C controller †
2. The I²C controller acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter. †
3. The I²C controller asserts the `RD_REQ` interrupt (bit 5 of the `IC_RAW_INTR_STAT` register) and waits for software to respond. †

If the RD_REQ interrupt has been masked, due to bit 5 of the IC_INTR_MASK register (M_RD_REQ bit field) being set to 0, then it is recommended that you instruct the CPU to perform periodic reads of the IC_RAW_INTR_STAT register. †

- Reads that indicate bit 5 of the IC_RAW_INTR_STAT register (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted. †
- Software must then act to satisfy the I²C transfer. †
- The timing interval used should be in the order of 10 times the fastest SCL clock period the I²C controller can handle. For example, for 400 Kbps, the timing interval is 25 us. †

  Note: The value of 10 is recommended here because this is approximately the amount of time required for a single byte of data transferred on the I²C bus.†

4. If there is any data remaining in the TX FIFO before receiving the read request, the I²C controller asserts a TX_ABRT interrupt (bit 6 of the IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO. †

   Note: Because the I²C controller's TX FIFO is forced into a flushed/reset state whenever a TX_ABRT event occurs, it is necessary for software to release the I²C controller from this state by reading the IC_CLR_TX_ABRT register before attempting to write into the TX FIFO. For more information, refer to the C_RAW_INTR_STAT register description in the register map.†

   If the TX_ABRT interrupt has been masked, due to of IC_INTR_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that the CPU performs periodic reads of the IC_RAW_INTR_STAT register. †

   - Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted. †
   - There is no further action required from software. †
   - The timing interval used should be similar to that described in the previous step for the IC_RAW_INTR_STAT[5] register. †
5. Software writes to the DAT bits of the IC_DATA_CMD register with the data to be written and writes a 0 in bit 8.†
6. Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the IC_RAW_INTR_STAT register before proceeding. †

   If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1. †
7. The I²C controller transmits the byte. †
8. The master may hold the I²C bus by issuing a RESTART condition or release the bus by issuing a STOP condition. †

Slave-Receiver Operation for a Single Byte

When another I²C master device on the bus addresses the I²C controller and is sending data, the I²C controller acts as a slave-receiver and the following steps occur:†

1. The other I²C master device initiates an I²C transfer with an address that matches the I²C controller's slave address in the IC_SAR register. †
2. The I²C controller acknowledges the sent address and recognizes the direction of the transfer to indicate that the I²C controller is acting as a slave-receiver. †
3. I²C controller receives the transmitted byte and places it in the receive buffer. †

   Note: If the RX FIFO is completely filled with data when a byte is pushed, then an overflow occurs and the I²C controller continues with subsequent I²C transfers. Because a NACK is not generated, software must recognize the overflow when indicated by the I²C controller (by the

R_RX_OVER bit in the IC_INTR_STAT register) and take appropriate actions to recover from lost data. Hence, there is a real time constraint on software to service the RX FIFO before the latter overflow as there is no way to reapply pressure to the remote transmitting master. †

4. I^{2}C controller asserts the RX_FULL interrupt (IC_RAW_INTR_STAT[2] register). †

   If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that the CPU does periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted. †
5. Software may read the byte from the IC_DATA_CMD register (bits 7:0). †
6. The other master device may hold the I^{2}C bus by issuing a RESTART condition or release the bus by issuing a STOP condition. †

Slave-Transfer Operation for Bulk Transfers

In the standard I^{2}C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO. The I^{2}C controller is designed to handle more data in the TX FIFO so that subsequent read requests can receive that data without raising an interrupt to request more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO. †

This mode only occurs when I^{2}C controller is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the I^{2}C controller raises the read request interrupt (RD_REQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master. †

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IC_INTR_STAT register being set to 0, then it is recommended that the CPU does periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section.†

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the TX FIFO. During the transmission of these bytes to the master, if the master acknowledges the last byte then the slave must raise the RD_REQ again because the master is requesting for more data. †

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses the I^{2}C controller and requests data, the TX FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the I^{2}C controller slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the TX FIFO. There is no need to issue RD_REQ again. †

If the remote master is to receive n bytes from the I^{2}C controller but the programmer wrote a number of bytes larger than n to the TX FIFO, then when the slave finishes sending the requested n bytes, it clears the TX FIFO and ignores any excess bytes. †

The I^{2}C controller generates a transmit abort (TX_ABRT) event to indicate the clearing of the TX FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the TX FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the TX FIFO are cleared at that time. †

Master Mode Operation

Initial Configuration

For master mode operation, the target address and address format can be changed dynamically without having to disable the I^{2}C controller. This feature is only applicable when the I^{2}C controller is acting as a master because the slave requires the component to be disabled before any changes can be made to the address. To use the I^{2}C controller as a master, perform the following steps: †

For multiple I^{2}C transfers, perform additional writes to the Tx FIFO such that the Tx FIFO does not become empty during the I^{2}C transaction. IF the Tx FIFO is completely emptied at any stage, then the master stalls the transfer by holding the SCL line low because there was no stop bit indicating the master to issue a STOP. The master will complete the transfer when it finds a Tx FIFO entry tagged with a Stop bit.

1. Disable the I^{2}C controller by writing 0 to bit 0 of the `IC_ENABLE` register. †
2. Write to the `IC_CON` register to set the maximum speed mode supported for slave operation (bits 2:1) and to specify whether the I^{2}C controller starts its transfers in 7/10 bit addressing mode when the device is a slave (bit 3). †
3. Write to the `IC_TAR` register the address of the I^{2}C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I^{2}C. The desired speed of the I^{2}C controller master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the `IC_10BITADDR_MASTER` bit field (bit 12). †
4. Enable the I^{2}C controller by writing a 1 in bit 0 of the `IC_ENABLE` register. †
5. Now write the transfer direction and data to be sent to the `IC_DATA_CMD` register. If the `IC_DATA_CMD` register is written before the I^{2}C controller is enabled, the data and commands are lost as the buffers are kept cleared when the I^{2}C controller is not enabled. †

Dynamic IC_TAR or IC_10BITADDR_MASTER Update

The I^{2}C controller supports dynamic updating of the `IC_TAR` (bits 9:0) and `IC_10BITADDR_MASTER` (bit 12) bit fields of the `IC_TAR` register. You can dynamically write to the `IC_TAR` register provided the following conditions are met: †

- The I^{2}C controller is not enabled (`IC_ENABLE`=0); †
- The I^{2}C controller is enabled (`IC_ENABLE`=1); AND I^{2}C controller is NOT engaged in any Master (TX, RX) operation (`IC_STATUS[5]`=0); AND I^{2}C controller is enabled to operate in Master mode (`IC_CON[0]`=1); AND there are no entries in the TX FIFO (`IC_STATUS[2]`=1) †

Master Transmit and Master Receive

The I^{2}C controller supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I^{2}C Rx/Tx Data Buffer and Command Register (`IC_DATA_CMD`). The CMD bit [8] should be written to 0 for I^{2}C write operations. Subsequently, a read command may be issued by writing "don't cares" to the lower byte of the `IC_DATA_CMD` register, and a 1 should be written to the CMD bit.†

Disabling the I^{2}C Controller

The register IC_ENABLE_STATUS is added to allow software to unambiguously determine when the hardware has completely shutdown in response to the IC_ENABLE register being set from 1 to 0. †

1. Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I^{2}C transfer speed used in the system and supported by the I^{2}C controller. For example, if the highest I^{2}C transfer mode is 400 Kbps, then ti2c_poll is 25 us. †
2. Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported. †
3. Execute a blocking thread/process/function that prevents any further I^{2}C master transactions to be started by software, but allows any pending transfers to be completed.
   - This step can be ignored if the I^{2}C controller is programmed to operate as an I^{2}C slave only. †
4. The variable POLL_COUNT is initialized to zero. †
5. Set IC_ENABLE to 0. †
6. Read the IC_ENABLE_STATUS register and test the IC_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT >= MAX_T_POLL_COUNT, exit with the relevant error code. †
7. If IC_ENABLE_STATUS[0] is 1, then sleep for ti2c_poll and proceed to the previous step. Otherwise, exit with a relevant success code. †

DMA Controller Operation

To enable the DMA controller interface on the I^{2}C controller, you must write the DMA Control Register (IC_DMA_CR). Writing a 1 to the TDMAE bit field of IC_DMA_CR register enables the I^{2}C controller transmit handshaking interface. Writing a 1 to the RDMAE bit field of the IC_DMA_CR register enables the I^{2}C controller receive handshaking interface.†

The FIFO buffer depth (FIFO_DEPTH) for both the RX and TX buffers in the I^{2}C controller is 64 entries.

Related Information

DMA Controller on page 16-1
For details about the DMA burst length microcode setup, refer to the *DMA controller* chapter.

Transmit FIFO Underflow

During I^{2}C serial transfers, transmit FIFO requests are made to the DMA controller whenever the number of entries in the transmit FIFO is less than or equal to the value in DMA Transmit Data Level Register (IC_DMA_TDLR), also known as the watermark level. The DMA controller responds by writing a burst of data to the transmit FIFO buffer, of length specified as DMA burst length. †

Note: Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously, that is, when the FIFO begins to empty, another DMA request should be triggered. Otherwise, the FIFO will run out of the data (underflow) causing the maser to stall the transfer by holding the SCL line low. To prevent this condition, you must set the watermark level correctly.†

Related Information

DMA Controller on page 16-1
For details about the DMA burst length microcode setup, refer to the *DMA controller* chapter.

Transmit Watermark Level

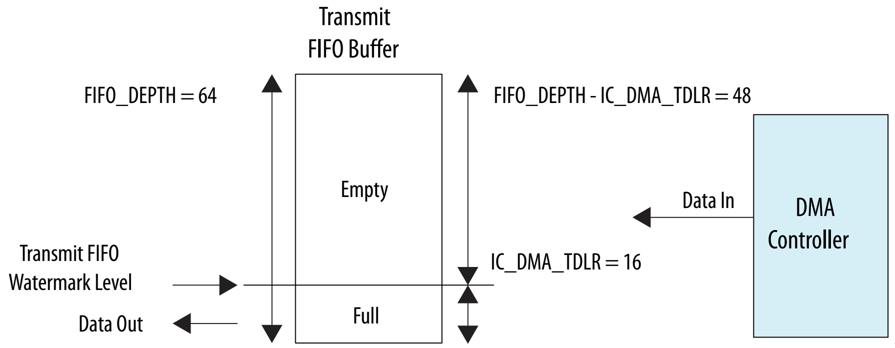
Consider the example where the assumption is made: †

DMA burst length = `FIFO_DEPTH` - `IC_DMA_TDLR` †

Here the number of data items to be transferred in a DMA burst is equal to the empty space in the transmit FIFO. Consider the following two different watermark level settings: †

- Case 1: `IC_DMA_TDLR` = 16: †
  - Transmit FIFO watermark level = `IC_DMA_TDLR` = 16: †
  - DMA burst length = `FIFO_DEPTH` - `IC_DMA_TDLR` = 48: †
  - I^2C transmit `FIFO_DEPTH` = 64: †
  - Block transaction size = 240: †

Figure 20-12: Transmit FIFO Watermark Level = 16



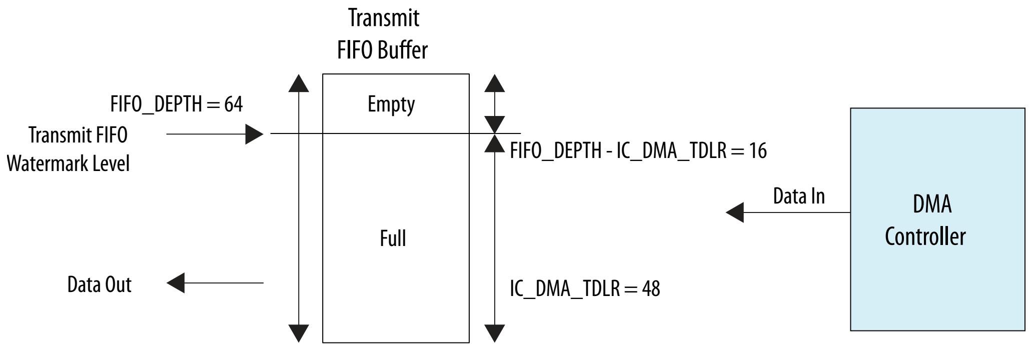
The number of burst transactions needed equals the block size divided by the number of data items per burst:

Block transaction size/DMA burst length = 240/48 = 5

The number of burst transactions in the DMA block transfer is 5. But the watermark level, `IC_DMA_TDLR`, is quite low. Therefore, the probability of transmit underflow is high where the I^2C serial transmit line needs to transmit data, but there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the FIFO becomes empty.

- Case 2: `IC_DMA_TDLR` = 48 †
  - Transmit FIFO watermark level = `IC_DMA_TDLR` = 48 †
  - DMA burst length = `FIFO_DEPTH` - `IC_DMA_TDLR` = 16 †
  - I^2C transmit `FIFO_DEPTH` = 64 †
  - Block transaction size = 240 †

Figure 20-13: Transmit FIFO Watermark Level = 48



Number of burst transactions in block: †

Block transaction size/DMA burst length = 240/16 = 15 †

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, `IC_ DMA_TDLR`, is high. Therefore, the probability of I^2C transmit underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the I^2C transmit FIFO becomes empty. †

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bursts per block and worse bus utilization than the former case. †

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the I^2C transmits data to the rate at which the DMA can respond to destination burst requests. †

Transmit FIFO Overflow

Setting the DMA burst length to a value greater than the watermark level that triggers the DMA request might cause overflow when there is not enough space in the transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow: †

DMA burst length <= `FIFO_DEPTH` - `IC_DMA_TDLR`

In case 2: `IC_DMA_TDLR` = 48, the amount of space in the transmit FIFO at the time of the burst request is made is equal to the DMA burst length. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction. †

Therefore, for optimal operation, DMA burst length should be set at the FIFO level that triggers a transmit DMA request; that is: †

DMA burst length = `FIFO_DEPTH` - `IC_DMA_TDLR`

Adhering to this equation reduces the number of DMA bursts needed for block transfer, and this in turn improves bus utilization. †

The transmit FIFO will not be full at the end of a DMA burst transfer if the I^2C controller has successfully transmitted one data item or more on the I^2C serial transmit line during the transfer. †

Receive FIFO Overflow

During I^{2}C serial transfers, receive FIFO requests are made to the DMA whenever the number of entries in the receive FIFO is at or above the DMA Receive Data Level Register, that is IC_DMA_RDLR + 1. This is known as the watermark level. The DMA responds by fetching a burst of data from the receive FIFO. †

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously, that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise the FIFO will fill with data (overflow). To prevent this condition, the user must set the watermark level correctly. †

Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, IC_DMA_RDLR + 1, should be set to minimize the probability of overflow, as shown in the Receive FIFO Buffer diagram. It is a trade off between the number of DMA burst transactions required per block versus the probability of an overflow occurring. †

Receive FIFO Underflow

Setting the source transaction burst length greater than the watermark level can cause underflow where there is not enough data to service the source burst request. Therefore, the following equation must be adhered to avoid underflow: †

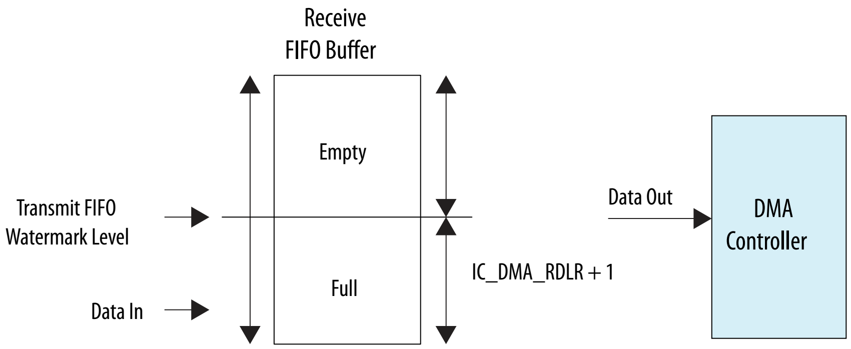
DMA burst length = IC_DMA_RDLR + 1

If the number of data items in the receive FIFO is equal to the source burst length at the time of the burst request is made, the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA burst length should be set at the watermark level, IC_DMA_RDLR + 1. †

Adhering to this equation reduces the number of DMA bursts in a block transfer, which in turn can avoid underflow and improve bus utilization. †

Note: The receive FIFO will not be empty at the end of the source burst transaction if the I^{2}C controller has successfully received one data item or more on the I^{2}C serial receive line during the burst. †

Figure 20-14: Receive FIFO Buffer



I^{2}C Controller Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- I^{2}C Module 0
- I^{2}C Module 1
- I^{2}C Module 2
- I^{2}C Module 3

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

I2C Module Address Map

Registers in the I2C module

Module Instance	Base Address
i2c0	0xFFC04000
i2c1	0xFFC05000
i2c2	0xFFC06000
i2c3	0xFFC07000

I2C Module

Register	Offset	Width	Access	Reset Value	Description
ic_con on page 20-25	0x0	32	RW	0x7D	Control Register
ic_tar on page 20-27	0x4	32	RW	0x1055	Target Address Register
ic_sar on page 20-29	0x8	32	RW	0x55	Slave Address Register
ic_data_cmd on page 20-30	0x10	32	RW	0x0	Tx Rx Data and Command Register
ic_ss_scl_hcnt on page 20-32	0x14	32	RW	0x190	Std Spd Clock SCL HCNT Register
ic_ss_scl_lcnt on page 20-33	0x18	32	RW	0x1D6	Std Spd Clock SCL LCNT Register
ic_fs_scl_hcnt on page 20-34	0x1C	32	RW	0x3C	Fast Spd Clock SCL HCNT Register
ic_fs_scl_lcnt on page 20-35	0x20	32	RW	0x82	Fast Spd Clock SCL LCNT Register
ic_intr_stat on page 20-36	0x2C	32	RO	0x0	Interrupt Status Register

Register	Offset	Width	Acces s	Reset Value	Description
ic_intr_mask on page 20-39	0x30	32	RW	0x8FF	Interrupt Mask Register
ic_raw_intr_stat on page 20-41	0x34	32	RO	0x0	Raw Interrupt Status Register
ic_rx_tl on page 20-44	0x38	32	RW	0x0	Receive FIFO Threshold Register
ic_tx_tl on page 20-45	0x3C	32	RW	0x0	Transmit FIFO Threshold Level Register
ic_clr_intr on page 20-45	0x40	32	RO	0x0	Combined and Individual Interrupt Register
ic_clr_rx_under on page 20-46	0x44	32	RO	0x0	Rx Under Interrupt Register
ic_clr_rx_over on page 20-47	0x48	32	RO	0x0	RX Over Interrupt Register
ic_clr_tx_over on page 20-47	0x4C	32	RO	0x0	TX Over Interrupt Register
ic_clr_rd_req on page 20-48	0x50	32	RO	0x0	Interrupt Read Request Register
ic_clr_tx_abrt on page 20-49	0x54	32	RO	0x0	Tx Abort Interrupt Register
ic_clr_rx_done on page 20-49	0x58	32	RO	0x0	Rx Done Interrupt Register
ic_clr_activity on page 20-50	0x5C	32	RO	0x0	Activity Interrupt Register
ic_clr_stop_det on page 20-51	0x60	32	RO	0x0	Stop Detect Interrupt Register
ic_clr_start_det on page 20-52	0x64	32	RO	0x0	Start Detect Interrupt Register
ic_clr_gen_call on page 20-52	0x68	32	RO	0x0	GEN CALL Interrupt Register
ic_enable on page 20-53	0x6C	32	RW	0x0	Enable Register
ic_status on page 20-55	0x70	32	RO	0x6	Status Register
ic_txflr on page 20-56	0x74	32	RO	0x0	Transmit FIFO Level Register
ic_rxflr on page 20-57	0x78	32	RO	0x0	Receive FIFO Level Register
ic_sda_hold on page 20-58	0x7C	32	RW	0x1	SDA Hold Register
ic_tx_abrt_source on page 20-58	0x80	32	RW	0x0	Transmit Abort Source Register

Register	Offset	Width	Acces s	Reset Value	Description
ic_slv_data_nack_only on page 20-61	0x84	32	RW	0x0	Generate Slave Data NACK
ic_dma_cr on page 20-62	0x88	32	RW	0x0	DMA Control
ic_dma_tdlr on page 20-63	0x8C	32	RW	0x0	DMA Transmit Data Level
ic_dma_rdlr on page 20-63	0x90	32	RW	0x0	Receive Data Level
ic_sda_setup on page 20-64	0x94	32	RW	0x64	SDA Setup Register
ic_ack_general_call on page 20-65	0x98	32	RW	0x1	ACK General Call
ic_enable_status on page 20-66	0x9C	32	RO	0x0	Enable Status Register
ic_fs_spklen on page 20-68	0xA0	32	RW	0x2	SS and FS Spike Suppression Limit Register
ic_comp_param_1 on page 20-68	0xF4	32	RO	0x3F3FEA	Component Parameter Register 1
ic_comp_version on page 20-70	0xF8	32	RO	0x3132302A	Component Version Register
ic_comp_type on page 20-71	0xFC	32	RO	0x44570140	Component Type Register

ic_con

This register can be written only when the I2C is disabled, which corresponds to the Bit [0] of the Enable Register being set to 0. Writes at other times have no effect.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04000
i2c1	0xFFC05000	0xFFC05000
i2c2	0xFFC06000	0xFFC06000
i2c3	0xFFC07000	0xFFC07000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ic_slave_disable RW 0x1	ic_restart_en RW 0x1	ic_10bitaddr_master RW 0x1	ic_10bitaddr_slave RW 0x1	speed RW 0x2		master_mode RW 0x1

ic_con Fields

Bit	Name	Description	Access	Reset
6	ic_slave_disable	This bit controls whether I2C has its slave disabled. The slave will be disabled, after reset. NOTE: Software should ensure that if this bit is written with 0, then bit [0] of this register should also be written with a 0. **Value** — **Description** 0x1 — slave disable 0x0 — slave enable	RW	0x1
5	ic_restart_en	Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several I2C operations. When RESTART is disabled, the master is prohibited from performing the following functions - Changing direction within a transfer (split), - Sending a START BYTE, - High-speed mode operation, - Combined format transfers in 7-bit addressing modes, - Read operation with a 10-bit address, - Sending multiple bytes per transfer, By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I2C transfers. If the above operations are performed, it will result in setting bit [6](tx_abort) of the Raw Interrupt Status Register. **Value** — **Description** 0x0 — restart master disable 0x1 — restart master enable	RW	0x1

Bit	Name	Description	Access	Reset
4	`ic_10bitaddr_master`	This bit controls whether the I2C starts its transfers in 7-or 10-bit addressing mode when acting as a master. **Value** — **Description** 0x0 — 7-bit addressing 0x1 — 10-bit addressing	RW	0x1
3	`ic_10bitaddr_slave`	When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses. In 7-bit addressing, only the lower 7 bits of the Slave Address Register are compared. The I2C responds will only respond to 10-bit addressing transfers that match the full 10 bits of the Slave Address register. **Value** — **Description** 0x0 — 7-bit addressing 0x1 — 10-bit addressing	RW	0x1
2:1	`speed`	These bits control at which speed the I2C operates, its setting is relevant only if one is operating the I2C in master mode. Hardware protects against illegal values being programmed by software. This field should be programmed only with standard or fast speed. **Value** — **Description** 0x1 — standard mode (100 kbit/s) 0x2 — fast mode (400 kbit/s)	RW	0x2
0	`master_mode`	This bit controls whether the i2c master is enabled. NOTE: Software should ensure that if this bit is written with '1', then bit 6 should also be written with a '1'. **Value** — **Description** 0x0 — master disabled 0x1 — master enabled	RW	0x1

ic_tar

This register can be written to only when the ic_enable register is set to 0. This register is 13 bits wide. All bits can be dynamically updated as long as any set of the following conditions are true, (Enable Register bit 0 is set to 0) or (Enable Register bit 0 is set to 1 AND (I2C is NOT engaged in any Master [tx, rx] operation [ic_status register mst_activity bit 5 is set to 0]) AND (I2C is enabled to operate in Master mode[ic_con bit[0] is set to one]) AND (there are NO entries in the TX FIFO Register [IC_STATUS bit [2] is set to 1])

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04004
i2c1	0xFFC05000	0xFFC05004
i2c2	0xFFC06000	0xFFC06004
i2c3	0xFFC07000	0xFFC07004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			ic_10bitaddr_master RW 0x1	special RW 0x0	gc_or_start RW 0x0	ic_tar RW 0x55									

ic_tar Fields

Bit	Name	Description	Access	Reset
12	ic_10bitaddr_master	This bit controls whether the i2c starts its transfers in 7-bit or 10-bit addressing mode when acting as a master. **Value** **Description** 0x0 Master Address, 7bit 0x1 Master Address, 10bit	RW	0x1
11	special	This bit indicates whether software performs a General Call or START BYTE command. **Value** **Description** 0x0 Ignore bit 10 gc_or_start and use ic_tar normally 0x1 Perform special I2C command as specified in gc_or_start	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>10</td><td>gc_or_start</td><td>If bit 11 (SPECIAL) of this Register is set to 1, then this bit indicates whether a General Call or START byte command is to be performed by the I2C or General Call Address after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the Raw Interrupt_Status register. The I2C remains in General Call mode until the special bit value (bit 11) is cleared.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>General Call</td></tr>
<tr><td>0x1</td><td>START Byte</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>9:0</td><td>ic_tar</td><td>This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the ic_tar and ic_sar are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only a slave.</td><td>RW</td><td>0x55</td></tr>
</table>

ic_sar

Holds Address of Slave

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04008
i2c1	0xFFC05000	0xFFC05008
i2c2	0xFFC06000	0xFFC06008
i2c3	0xFFC07000	0xFFC07008

Offset: 0x8

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">Reserved</td><td colspan="10">ic_sar
RW 0x55</td></tr>
</table>

ic_sar Fields

Bit	Name	Description	Access	Reset
9:0	ic_sar	The Slave Address register holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only Field Bits [6:0] of the Slave Address Register are used. This register can be written only when the I2C interface is disabled, which corresponds to field bit 0 of the Enable Register being set to 0. Writes at other times have no effect. Note, the default values cannot be any of the reserved address locations: that is, 0x00 to 0x07, or 0x78 to 0x7f. The correct operation of the device is not guaranteed if you program the Slave Address Register or Target Address Register to a reserved value.	RW	0x55

ic_data_cmd

This is the register the CPU writes to when filling the TX FIFO. Reading from this register returns bytes from RX FIFO.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04010
i2c1	0xFFC05000	0xFFC05010
i2c2	0xFFC06000	0xFFC06010
i2c3	0xFFC07000	0xFFC07010

Offset: 0x10

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">Reserved</td><td>restart
WO
0x0</td><td>stop
WO
0x0</td><td>cmd
WO
0x0</td><td colspan="8">dat
RW 0x0</td></tr>
</table>

ic_data_cmd Fields

Bit	Name	Description	Access	Reset
10	`restart`	This bit controls whether a RESTART is issued before the byte is sent or received. 1 = A RESTART is issued before the data is sent/received (according to the value of CMD), regardless of whether or not the transfer direction is changing from the previous command. 0 = A RESTART is issued only if the transfer direction is changing from the previous command. **Value** / **Description** 0x1 Issue Restart 0x0 Issue Restart On Direction Change	`WO`	`0x0`
9	`stop`	This bit controls whether a STOP is issued after the byte is sent or received. 1 = STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0 = STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/ receiving data bytes according to the value of the CMD bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO. **Value** / **Description** 0x1 Issue Stop 0x0 Do Not Issue Stop	`WO`	`0x0`

Bit	Name	Description	Access	Reset
8	cmd	This bit controls whether a read or a write is performed. This bit does not control the direction when the I2C acts as a slave. It controls only the direction when it acts as a master. When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a 'don't care' because writes to this register are not required. In slave-transmitter mode, a '0' indicates that the CPU data is to be transmitted. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a tx_abrt interrupt (bit 6 of the Raw Intr Status Register), unless bit 11 special in the Target Address Register has been cleared. If a '1' is written to this bit after receiving a RD_REQ interrupt, then a tx_abrt interrupt occurs. NOTE: It is possible that while attempting a master I2C read transfer on I2C, a RD_REQ interrupt may have occurred simultaneously due to a remote I2C master addressing I2C. In this type of scenario, I2C ignores the Data Cmd write, generates a tx_abrt interrupt, and waits to service the RD_REQ interrupt. **Value** **Description** 0x1 Master Read 0x0 Master Write	WO	0x0
7:0	dat	This Field contains the data to be transmitted or received on the I2C bus. If you are writing to these bits and want to perform a read, bits 7:0 (dat) are ignored by the I2C. However, when you read from this register, these bits return the value of data received on the I2C interface.	RW	0x0

ic_ss_scl_hcnt

This register sets the SCL clock high-period count for standard speed.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04014
i2c1	0xFFC05000	0xFFC05014
i2c2	0xFFC06000	0xFFC06014
i2c3	0xFFC07000	0xFFC07014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ic_ss_scl_hcnt RW 0x190															

ic_ss_scl_hcnt Fields

Bit	Name	Description	Access	Reset
15:0	ic_ss_scl_hcnt	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This field sets the SCL clock high-period count for standard speed. This register can be written only when the I2C interface is disabled which corresponds to the Enable Register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. It is readable and writeable. NOTE: This register must not be programmed to a value higher than 65525, because I2C uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of IC_SS_SCL_HCNT + 10.	RW	0x190

ic_ss_scl_lcnt

This register sets the SCL clock low-period count for standard speed

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04018
i2c1	0xFFC05000	0xFFC05018
i2c2	0xFFC06000	0xFFC06018
i2c3	0xFFC07000	0xFFC07018

Offset: 0x18

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ic_ss_scl_lcnt RW 0x1D6															

ic_ss_scl_lcnt Fields

Bit	Name	Description	Access	Reset
15:0	ic_ss_scl_lcnt	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This field sets the SCL clock low period count for standard speed. This register can be written only when the I2C interface is disabled which corresponds to the Enable Register register being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this from being written, and if attempted, results in 8 being set.	RW	0x1D6

ic_fs_scl_hcnt

This register sets the SCL clock high-period count for fast speed

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0401C
i2c1	0xFFC05000	0xFFC0501C
i2c2	0xFFC06000	0xFFC0601C
i2c3	0xFFC07000	0xFFC0701C

Offset: 0x1C

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ic_fs_scl_hcnt
RW 0x3C</td></tr>
</table>

ic_fs_scl_hcnt Fields

Bit	Name	Description	Access	Reset
15:0	`ic_fs_scl_hcnt`	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. This register goes away and becomes read-only returning 0s if in Standard Speed Mode. This register can be written only when the I2C interface is disabled, which corresponds to the Enable Register being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this from being written, and if attempted results in 6 being set.	RW	0x3C

ic_fs_scl_lcnt

This register sets the SCL clock low period count

Module Instance	Base Address	Register Address
`i2c0`	`0xFFC04000`	`0xFFC04020`
`i2c1`	`0xFFC05000`	`0xFFC05020`
`i2c2`	`0xFFC06000`	`0xFFC06020`
`i2c3`	`0xFFC07000`	`0xFFC07020`

Offset: `0x20`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">ic_fs_scl_lcnt
RW 0x82</td></tr>
</table>

ic_fs_scl_lcnt Fields

Bit	Name	Description	Access	Reset
15:0	ic_fs_scl_lcnt	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This field sets the SCL clock low period count for fast speed. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. This register can be written only when the I2C interface is disabled, which corresponds to the Enable Register being set to 0. Writes at other times have no effect.The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.	RW	0x82

ic_intr_stat

Each bit in this register has a corresponding mask bit in the Interrupt Mask Register. These bits are cleared by reading the matching Interrupt Clear Register. The unmasked raw versions of these bits are available in the Raw Interrupt Status Register.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0402C
i2c1	0xFFC05000	0xFFC0502C
i2c2	0xFFC06000	0xFFC0602C
i2c3	0xFFC07000	0xFFC0702C

Offset: 0x2C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				r_gen_call RO 0x0	r_start_det RO 0x0	r_stop_det RO 0x0	r_activity RO 0x0	r_rx_done RO 0x0	r_tx_abrt RO 0x0	r_rd_req RO 0x0	r_tx_empty RO 0x0	r_tx_over RO 0x0	r_rx_full RO 0x0	r_rx_over RO 0x0	r_rx_under RO 0x0

ic_intr_stat Fields

Bit	Name	Description	Access	Reset
11	`r_gen_call`	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the ic_clr_gen_call register. I2C stores the received data in the Rx buffer.	RO	0x0
10	`r_start_det`	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	RO	0x0
9	`r_stop_det`	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	RO	0x0
8	`r_activity`	This bit captures I2C activity and stays set until it is cleared. There are four ways to clear it: - Disabling the I2C - Reading the ic_clr_activity register - Reading the ic_clr_intr register - I2C reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.	RO	0x0
7	`r_rx_done`	When the I2C is acting as a slave-transmitter, this bit is set to 1, if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.	RO	0x0
6	`r_tx_abrt`	This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'.When this bit is set to 1, the ic_tx_abrt_source register indicates the reason why the transmit abort takes places. NOTE: The I2C flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register ic_clr_tx_abrt is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface.	RO	0x0

Bit	Name	Description	Access	Reset
5	`r_rd_req`	This bit is set to 1 when i2c is acting as a slave and another I2C master is attempting to read data from I2C. The I2C holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the IC_DATA_CMD register. This bit is set to 0 just after the processor reads the ic_clr_rd_req register.	RO	0x0
4	`r_tx_empty`	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the ic_tx_tl register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the ic_enable bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, this bit is set to 0.	RO	0x0
3	`r_tx_over`	Set during transmit if the transmit buffer is filled to 64 and the processor attempts to issue another I2C command by writing to the Data and Command Register. When the module is disabled, this bit keeps its level until the master or slave state machines goes into idle, then interrupt is cleared.	RO	0x0
2	`r_rx_full`	Set when the receive buffer reaches or goes above the Receive FIFO Threshold Value(rx_tl). It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled, Bit [0] of the Enable Register set to 0, the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the Enable Register Bit 0 is programmed with a 0, regardless of the activity that continues.	RO	0x0
1	`r_rx_over`	Set if the receive buffer is completely filled to 64 and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled, Enable Register bit[0] is set to 0 this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	RO	0x0

Bit	Name	Description	Access	Reset
0	r_rx_under	Set if the processor attempts to read the receive buffer when it is empty by reading from the Tx Rx Data and Command Register. If the module is disabled, Enable Register is set to 0, this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	RO	0x0

ic_intr_mask

These bits mask their corresponding interrupt status bits.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04030
i2c1	0xFFC05000	0xFFC05030
i2c2	0xFFC06000	0xFFC06030
i2c3	0xFFC07000	0xFFC07030

Offset: 0x30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				m_gen_call RW 0x1	m_start_det RW 0x0	m_stop_det RW 0x0	m_activity RW 0x0	m_rx_done RW 0x1	m_tx_abrt RW 0x1	m_rd_req RW 0x1	m_tx_empty RW 0x1	m_tx_over RW 0x1	m_rx_full RW 0x1	m_rx_over RW 0x1	m_rx_under RW 0x1

ic_intr_mask Fields

Bit	Name	Description	Access	Reset
11	m_gen_call	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the ic_clr_gen_call register. I2C stores the received data in the Rx buffer.	RW	0x1
10	m_start_det	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	RW	0x0

Bit	Name	Description	Access	Reset
9	m_stop_det	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether i2c is operating in slave or master mode.	RW	0x0
8	m_activity	This bit captures i2c activity and stays set until it is cleared. There are four ways to clear it: - Disabling the i2c - Reading the ic_clr_activity register - Reading the ic_clr_intr register - System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.	RW	0x0
7	m_rx_done	When the I2C is acting as a slave-transmitter, this bit is set to 1, if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.	RW	0x1
6	m_tx_abrt	This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the ic_tx_abrt_source register indicates the reason why the transmit abort takes places. NOTE: The I2C flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register ic_clr_tx_abrt is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface.	RW	0x1
5	m_rd_req	This bit is set to 1 when I2C is acting as a slave and another I2C master is attempting to read data from I2C. The I2C holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the ic_data_cmd register. This bit is set to 0 just after the processor reads the ic_clr_rd_req register.	RW	0x1

Bit	Name	Description	Access	Reset
4	`m_tx_empty`	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the ic_tx_tl register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the ic_enable bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then this bit is set to 0.	`RW`	`0x1`
3	`m_tx_over`	Set during transmit if the transmit buffer is filled to 64 and the processor attempts to issue another I2C command by writing to the ic_data_cmd register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	`RW`	`0x1`
2	`m_rx_full`	Set when the receive buffer reaches or goes above the RX_TL threshold in the ic_rx_tl register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled ic_enable[0]=0, the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the ic_enable bit 0 is programmed with a 0, regardless of the activity that continues.	`RW`	`0x1`
1	`m_rx_over`	Set if the receive buffer is completely filled to 64 and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled ic_enable[0]=0, this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	`RW`	`0x1`
0	`m_rx_under`	Set if the processor attempts to read the receive buffer when it is empty by reading from the ic_data_cmd register. If the module is disabled ic_enable[0]=0, this bit keeps its level until the master or slave state machines go into idle, and then this interrupt is cleared.	`RW`	`0x1`

ic_raw_intr_stat

Unlike the ic_intr_stat register, these bits are not masked so they always show the true status of the I2C.

Module Instance	Base Address	Register Address
`i2c0`	`0xFFC04000`	`0xFFC04034`
`i2c1`	`0xFFC05000`	`0xFFC05034`
`i2c2`	`0xFFC06000`	`0xFFC06034`

Module Instance	Base Address	Register Address
i2c3	0xFFC07000	0xFFC07034

Offset: 0x34

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				gen_call RO 0x0	start_det RO 0x0	stop_det RO 0x0	activity RO 0x0	rx_done RO 0x0	tx_abrt RO 0x0	rd_req RO 0x0	tx_empty RO 0x0	tx_over RO 0x0	rx_full RO 0x0	rx_over RO 0x0	rx_under RO 0x0

ic_raw_intr_stat Fields

Bit	Name	Description	Access	Reset
11	gen_call	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the ic_clr_gen_call register. I2C stores the received data in the Rx buffer.	RO	0x0
10	start_det	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	RO	0x0
9	stop_det	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.	RO	0x0
8	activity	This bit captures i2c activity and stays set until it is cleared. There are four ways to clear it: - Disabling the I2C - Reading the ic_clr_activity register - Reading the ic_clr_intr register - System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the i2c module is idle, this bit remains set until cleared, indicating that there was activity on the bus.	RO	0x0
7	rx_done	When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.	RO	0x0

Bit	Name	Description	Access	Reset
6	tx_abrt	This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a 'transmit abort'. When this bit is set to 1, the IC_TX_ABRT_SOURCE register indicates the reason why the transmit abort takes places. NOTE: The I2C flushes/resets/empties the TX FIFO whenever this bit is set. The TX FIFO remains in this flushed state until the register ic_clr_tx_abrt is read. Once this read is performed, the TX FIFO is then ready to accept more data bytes from the APB interface.	RO	0x0
5	rd_req	This bit is set to 1 when I2C is acting as a slave and another I2C master is attempting to read data from I2C. The i2c holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the ic_data_cmd register. This bit is set to 0 just after the processor reads the ic_clr_rd_req register.	RO	0x0
4	tx_empty	This bit is set to 1 when the transmit buffer is at or below the threshold value set in the ic_tx_tl register. It is automatically cleared by hardware when the buffer level goes above the threshold. When the IC_ENABLE bit 0 is 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer activity, then this bit is set to 0.	RO	0x0
3	tx_over	Set during transmit if the transmit buffer is filled to 64 and the processor attempts to issue another I2C command by writing to the ic_data_cmd register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	RO	0x0
2	rx_full	Set when the receive buffer reaches or goes above the RX_TL threshold in the ic_rx_tl register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled ic_enable[0]=0, the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once the ic_enable bit 0 is programmed with a 0, regardless of the activity that continues.	RO	0x0

Bit	Name	Description	Access	Reset
1	`rx_over`	Set if the receive buffer is completely filled to 64 and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled ic_enable[0]=0), this bit keeps its level until the master or slave state machines go into then, this interrupt is cleared.	RO	0x0
0	`rx_under`	Set if the processor attempts to read the receive buffer when it is empty by reading from the ic_data_cmd register. If the module is disabled ic_enable[0]=0, this bit keeps its level until the master or slave state machines go into idle, then this interrupt is cleared.	RO	0x0

ic_rx_tl

I2C Receive FIFO Threshold Register.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04038
i2c1	0xFFC05000	0xFFC05038
i2c2	0xFFC06000	0xFFC06038
i2c3	0xFFC07000	0xFFC07038

Offset: `0x38`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rx_tl RW 0x0							

ic_rx_tl Fields

Bit	Name	Description	Access	Reset
7:0	`rx_tl`	Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in IC_RAW_INTR_STAT register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.	RW	0x0

ic_tx_tl

Sets FIFO depth for Interrupt.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0403C
i2c1	0xFFC05000	0xFFC0503C
i2c2	0xFFC06000	0xFFC0603C
i2c3	0xFFC07000	0xFFC0703C

Offset: 0x3C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								tx_tl RW 0x0							

ic_tx_tl Fields

Bit	Name	Description	Access	Reset
7:0	tx_tl	Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in ic_raw_intr_stat register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries.	RW	0x0

ic_clr_intr

Controls Interrupts

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04040
i2c1	0xFFC05000	0xFFC05040
i2c2	0xFFC06000	0xFFC06040
i2c3	0xFFC07000	0xFFC07040

Offset: 0x40

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_intr RO 0x0

ic_clr_intr Fields

Bit	Name	Description	Access	Reset
0	clr_intr	Read this register to clear the combined interrupt, all individual interrupts, and the IC_TX_ABRT_SOURCE register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the ic_tx_abrt_source register for an exception to clearing ic_tx_abrt_source.	RO	0x0

ic_clr_rx_under

Rx Under Interrupt Bits.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04044
i2c1	0xFFC05000	0xFFC05044
i2c2	0xFFC06000	0xFFC06044
i2c3	0xFFC07000	0xFFC07044

Offset: 0x44

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_rx_under RO 0x0

ic_clr_rx_under Fields

Bit	Name	Description	Access	Reset
0	`clr_rx_under`	Read this register to clear the RX_UNDER interrupt bit 0 of the ic_raw_intr_stat register.	RO	0x0

ic_clr_rx_over

Clears Rx over Interrupt Bit

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04048
i2c1	0xFFC05000	0xFFC05048
i2c2	0xFFC06000	0xFFC06048
i2c3	0xFFC07000	0xFFC07048

Offset: `0x48`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_rx_over RO 0x0

ic_clr_rx_over Fields

Bit	Name	Description	Access	Reset
0	`clr_rx_over`	Read this register to clear the RX_OVER interrupt bit 1 of the ic_raw_intr_stat register.	RO	0x0

ic_clr_tx_over

Clears Over Interrupts

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0404C
i2c1	0xFFC05000	0xFFC0504C
i2c2	0xFFC06000	0xFFC0604C
i2c3	0xFFC07000	0xFFC0704C

Offset: 0x4C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_tx_over RO 0x0

ic_clr_tx_over Fields

Bit	Name	Description	Access	Reset
0	clr_tx_over	Read this register to clear the TX_OVER interrupt (bit 3) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_rd_req

Clear RD_REQ Interrupt Register

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04050
i2c1	0xFFC05000	0xFFC05050
i2c2	0xFFC06000	0xFFC06050
i2c3	0xFFC07000	0xFFC07050

Offset: 0x50

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_rd_req RO 0x0

ic_clr_rd_req Fields

Bit	Name	Description	Access	Reset
0	`clr_rd_req`	Read this register to clear the RD_REQ interrupt (bit 5) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_tx_abrt

Clear TX_ABRT Interrupt

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04054
i2c1	0xFFC05000	0xFFC05054
i2c2	0xFFC06000	0xFFC06054
i2c3	0xFFC07000	0xFFC07054

Offset: `0x54`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_tx_abort RO 0x0

ic_clr_tx_abrt Fields

Bit	Name	Description	Access	Reset
0	`clr_tx_abort`	Read this register to clear the TX_ABRT interrupt (bit 6) of the ic_raw_intr_stat register, and the ic_tx_abrt_source register. This also releases the TX FIFO from the flushed/reset state, allowing more writes to the TX FIFO. Refer to Bit 9 of the ic_tx_abrt_source register for an exception to clearing ic_tx_abrt_source.	RO	0x0

ic_clr_rx_done

Clear RX_DONE Interrupt Register

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04058

Module Instance	Base Address	Register Address
i2c1	0xFFC05000	0xFFC05058
i2c2	0xFFC06000	0xFFC06058
i2c3	0xFFC07000	0xFFC07058

Offset: 0x58

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_rx_done RO 0x0

ic_clr_rx_done Fields

Bit	Name	Description	Access	Reset
0	clr_rx_done	Read this register to clear the RX_DONE interrupt (bit 7) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_activity

Clears ACTIVITY Interrupt

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0405C
i2c1	0xFFC05000	0xFFC0505C
i2c2	0xFFC06000	0xFFC0605C
i2c3	0xFFC07000	0xFFC0705C

Offset: 0x5C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_activity RO 0x0

ic_clr_activity Fields

Bit	Name	Description	Access	Reset
0	clr_activity	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_stop_det

Clear Interrupts.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04060
i2c1	0xFFC05000	0xFFC05060
i2c2	0xFFC06000	0xFFC06060
i2c3	0xFFC07000	0xFFC07060

Offset: 0x60

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_stop_det RO 0x0

ic_clr_stop_det Fields

Bit	Name	Description	Access	Reset
0	clr_stop_det	Read this register to clear the clr_stop_det interrupt (bit 9) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_start_det

Clears START_DET Interrupt

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04064
i2c1	0xFFC05000	0xFFC05064
i2c2	0xFFC06000	0xFFC06064
i2c3	0xFFC07000	0xFFC07064

Offset: 0x64

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_start_det RO 0x0

ic_clr_start_det Fields

Bit	Name	Description	Access	Reset
0	clr_start_det	Read this register to clear the start_det interrupt (bit 10) of the ic_raw_intr_stat register.	RO	0x0

ic_clr_gen_call

Clear GEN_CALL Interrupt Register

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04068
i2c1	0xFFC05000	0xFFC05068
i2c2	0xFFC06000	0xFFC06068
i2c3	0xFFC07000	0xFFC07068

Offset: 0x68

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															clr_gen_call RO 0x0

ic_clr_gen_call Fields

Bit	Name	Description	Access	Reset
0	clr_gen_call	Read this register to clear the GEN_CALL interrupt (bit 11) of ic_raw_intr_stat register.	RO	0x0

ic_enable

Enable and disable i2c operation

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0406C
i2c1	0xFFC05000	0xFFC0506C
i2c2	0xFFC06000	0xFFC0606C
i2c3	0xFFC07000	0xFFC0706C

Offset: 0x6C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														txabort RW 0x0	enable RW 0x0

ic_enable Fields

Bit	Name	Description	Access	Reset
1	`txabort`	Write 1 does a TX abort. Self cleared on abort completion	RW	0x0
0	`enable`	Controls whether the I2C is enabled. Software can disable I2C while it is active. However, it is important that care be taken to ensure that I2C is disabled properly. When the I2C is disabled, the following occurs: The TX FIFO and RX FIFO get flushed. Status bits in the IC_INTR_STAT register are still active until I2C goes into IDLE state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the I2C stops the current transfer at the end of the current byte and does not acknowledge the transfer. The l4_sp_clk synchronizes pclk and ic_clk. The register ic_enable_status is added to allow software to determine when the hardware has completely shutdown in response to the IC_ENABLE register being set from 1 to 0. Only one register is required to be monitored. Procedure for Disabling I2C 1. Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by I2C. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c_poll is 25us. 2. Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported. 3. Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed. 4. The variable POLL_COUNT is initialized to zero. 5. Set IC_ENABLE to 0. 6. Read the IC_ENABLE_STATUS register and test the IC_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT >= MAX_T_POLL_COUNT, exit with the relevant error code. 7. If IC_ENABLE_STATUS[0] is 1, then sleep for ti2c_poll and proceed to the previous step. Otherwise, exit with a relevant success code. **Value** **Description** 0x0 Disables i2c. TX and RX FIFOs are held in an erased state 0x1 Enables i2c. Software can disable i2c while it is active	RW	0x0

ic_status

This is a read-only register used to indicate the current transfer status and FIFO status. The status register may be read at any time. None of the bits in this register request an interrupt.When the I2C is disabled by writing 0 in bit 0 of the ic_enable register: - Bits 1 and 2 are set to 1 - Bits 3 and 4 are set to 0 When the master or slave state machines goes to idle - Bits 5 and 6 are set to 0

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04070
i2c1	0xFFC05000	0xFFC05070
i2c2	0xFFC06000	0xFFC06070
i2c3	0xFFC07000	0xFFC07070

Offset: 0x70

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									slv_activity RO 0x0	mst_activity RO 0x0	rff RO 0x0	rfne RO 0x0	tfe RO 0x1	tfnf RO 0x1	activity RO 0x0

ic_status Fields

Bit	Name	Description	Access	Reset
6	slv_activity	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. **Value** **Description** 0x0 Slave FSM is in IDLE state so the Slave part of i2c is not Active 0x1 Slave FSM is not in IDLE state so the Slave part of i2c is Active	RO	0x0

Bit	Name	Description	Access	Reset
5	`mst_activity`	When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. Note:IC_STATUS[0]-that is, ACTIVITY bit-is the OR of SLV_ACTIVITY and MST_ACTIVITY bits. **Value** **Description** 0x0 Master FSM is in IDLE state. Master part of i2c is not Active 0x1 Master FSM is not in IDLE state. Master part of i2c is Active	`RO`	`0x0`
4	`rff`	Receive FIFO Completely Full. **Value** **Description** 0x0 Receive FIFO is not full 0x1 Receive FIFO is full	`RO`	`0x0`
3	`rfne`	Receive FIFO Not Empty. **Value** **Description** 0x0 Receive FIFO is empty 0x1 Receive FIFO is not empty	`RO`	`0x0`
2	`tfe`	Transmit FIFO Empty. **Value** **Description** 0x0 Transmit FIFO is not empty 0x1 Transmit FIFO is empty	`RO`	`0x1`
1	`tfnf`	Transmit Fifo Full **Value** **Description** 0x0 Transmit FIFO is full 0x1 Transmit FIFO is not full	`RO`	`0x1`
0	`activity`	I2C Activity.	`RO`	`0x0`

ic_txflr

This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever: - The I2C is disabled - There is a transmit abort that is, TX_ABRT bit is set in the ic_raw_intr_stat register. The slave bulk transmit mode is aborted The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04074
i2c1	0xFFC05000	0xFFC05074
i2c2	0xFFC06000	0xFFC06074
i2c3	0xFFC07000	0xFFC07074

Offset: 0x74

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									txflr RO 0x0						

ic_txflr Fields

Bit	Name	Description	Access	Reset
6:0	txflr	Transmit FIFO Level.Contains the number of valid data entries in the transmit FIFO.	RO	0x0

ic_rxflr

This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever: - The I2C is disabled - Whenever there is a transmit abort caused by any of the events tracked in ic_tx_abrt_source The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04078
i2c1	0xFFC05000	0xFFC05078
i2c2	0xFFC06000	0xFFC06078
i2c3	0xFFC07000	0xFFC07078

Offset: 0x78

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									rxflr RO 0x0						

ic_rxflr Fields

Bit	Name	Description	Access	Reset
6:0	`rxflr`	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.	RO	0x0

ic_sda_hold

This register controls the amount of time delay (in terms of number of l4_sp_clk clock periods) introduced in the falling edge of SCL, relative to SDA changing, when I2C services a read request in a slave-transmitter operation. The relevant I2C requirement is thd:DAT as detailed in the I2C Bus Specification.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0407C
i2c1	0xFFC05000	0xFFC0507C
i2c2	0xFFC06000	0xFFC0607C
i2c3	0xFFC07000	0xFFC0707C

Offset: 0x7C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ic_sda_hold RW 0x1															

ic_sda_hold Fields

Bit	Name	Description	Access	Reset
15:0	`ic_sda_hold`	Program to a minimum 0f 300ns.	RW	0x1

ic_tx_abrt_source

This register has 16 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the ic_clr_tx_abrt register or the ic_clr_intr register is read. To clear Bit 9, the source of the abrt_sbyte_norstrt must be fixed first; RESTART must be enabled (ic_con[5]=1), the special bit must be cleared (ic_tar[11]), or the gc_or_start bit must be cleared (ic_tar[10]). Once the source of the abrt_sbyte_norstrt is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the abrt_sbyte_norstrt is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04080

Module Instance	Base Address	Register Address
i2c1	0xFFC05000	0xFFC05080
i2c2	0xFFC06000	0xFFC06080
i2c3	0xFFC07000	0xFFC07080

Offset: 0x80

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
abrt_slvrd_intx RW 0x0	abrt_slv_arblost RW 0x0	abrt_slvflush_txfifo RW 0x0	arb_lost RW 0x0	abrt_master_dis RW 0x0	abrt_10b_rd_norstrt RW 0x0	abrt_sbyte_norstrt RW 0x0	abrt_hs_norstrt RW 0x0	abrt_sbyte_ackdet RW 0x0	abrt_hs_ackdet RW 0x0	abrt_gcall_read RW 0x0	abrt_gcall_noack RW 0x0	abrt_txdata_noack RW 0x0	abrt_10addr2_noack RW 0x0	abrt_10addr1_noack RW 0x0	abrt_7b_addr_noack RW 0x0

ic_tx_abrt_source Fields

Bit	Name	Description	Access	Reset
15	abrt_slvrd_intx	When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in CMD (bit 8) of IC_DATA_CMD register. Role of I2C: Slave-Transmitter	RW	0x0
14	abrt_slv_arblost	Slave lost the bus while transmitting data to a remote master. IC_TX_ABRT_SOURCE[12] is set at the same time. Note: Even though the slave never 'owns' the bus, something could go wrong on the bus. This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then i2c no longer own the bus. Role of I2C: Slave-Transmitter	RW	0x0
13	abrt_slvflush_txfifo	Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. Role of I2C: Slave-Transmitter	RW	0x0

Bit	Name	Description	Access	Reset
12	arb_lost	Master has lost arbitration, or if IC_TX_ABRT_SOURCE[14] is also set, then the slave transmitter has lost arbitration. Note: I2C can be both master and slave at the same time. Role of i2c: Master-Transmitter or Slave-Transmitter	RW	0x0
11	abrt_master_dis	User tries to initiate a Master operation with the Master mode disabled. Role of I2C: Master-Transmitter or Master-Receiver	RW	0x0
10	abrt_10b_rd_norstrt	The restart is disabled (ic_restart_en bit (ic_con[5]) =0) and the master sends a read command in 10-bit addressing mode. Role of I2C: Master-Receiver	RW	0x0
9	abrt_sbyte_norstrt	To clear Bit 9, the source of then abrt_sbyte_norstrt must be fixed first; restart must be enabled (ic_con[5]=1), the SPECIAL bit must be cleared (ic_tar[11]), or the GC_OR_START bit must be cleared (ic_tar[10]). Once the source of the abrt_sbyte_norstrt is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the abrt_sbyte_norstrt is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets reasserted. 1: The restart is disabled (IC_RESTART_EN bit (ic_con[5]) =0) and the user is trying to send a START Byte. Role of I2C: Master	RW	0x0
8	abrt_hs_norstrt	The restart is disabled (IC_RESTART_EN bit (IC_CON[5]) =0) and the user is trying to use the master to transfer data in High Speed mode. Role of i2c: Master-Transmitter or Master-Receiver	RW	0x0
7	abrt_sbyte_ackdet	Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Role of i2c: Master	RW	0x0
6	abrt_hs_ackdet	Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Role of i2c: Master	RW	0x0
5	abrt_gcall_read	i2c in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (IC_DATA_CMD[9] is set to 1). Role of i2c: Master-Transmitter	RW	0x0
4	abrt_gcall_noack	i2c in master mode sent a General Call and no slave on the bus acknowledged the General Call. Role of i2c: Master-Transmitter	RW	0x0

Bit	Name	Description	Access	Reset
3	`abrt_txdata_noack`	This is a master-mode only bit. Master has received an acknowledgement for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s). Role of i2c: Master-Transmitter	RW	0x0
2	`abrt_10addr2_noack`	Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Role of i2c: Master-Transmitter or Master-Receiver	RW	0x0
1	`abrt_10addr1_noack`	Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Role of i2c: Master-Transmitter or Master-Receiver	RW	0x0
0	`abrt_7b_addr_noack`	Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Role of i2c: Master-Transmitter or Master-Receiver	RW	0x0

ic_slv_data_nack_only

The register is used to generate a NACK for the data part of a transfer when i2c is acting as a slave-receiver.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04084
i2c1	0xFFC05000	0xFFC05084
i2c2	0xFFC06000	0xFFC06084
i2c3	0xFFC07000	0xFFC07084

Offset: 0x84

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															nack RW 0x0

ic_slv_data_nack_only Fields

Bit	Name	Description	Access	Reset
0	nack	This Bit control Nack generation **Value** **Description** 0x1 Generate NACK after data byte receive 0x0 Generate NACK/ACK normally	RW	0x0

ic_dma_cr

The register is used to enable the DMA Controller interface operation. There is a separate bit for transmit and receive. This can be programmed regardless of the state of IC_ENABLE.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04088
i2c1	0xFFC05000	0xFFC05088
i2c2	0xFFC06000	0xFFC06088
i2c3	0xFFC07000	0xFFC07088

Offset: 0x88

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														tdmae RW 0x0	rdmae RW 0x0

ic_dma_cr Fields

Bit	Name	Description	Access	Reset
1	tdmae	This bit enables/disables the transmit FIFO DMA channel. **Value** **Description** 0x0 Transmit DMA disable 0x1 Transmit DMA enabled	RW	0x0

Bit	Name	Description	Access	Reset
0	rdmae	This bit enables/disables the receive FIFO DMA channel. **Value** — **Description** 0x0 — Receive DMA disable 0x1 — Receive DMA enabled	RW	0x0

ic_dma_tdlr

This register supports DMA Transmit Operation.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0408C
i2c1	0xFFC05000	0xFFC0508C
i2c2	0xFFC06000	0xFFC0608C
i2c3	0xFFC07000	0xFFC0708C

Offset: 0x8C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										dmatdl RW 0x0					

ic_dma_tdlr Fields

Bit	Name	Description	Access	Reset
5:0	dmatdl	This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the i2c_dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.	RW	0x0

ic_dma_rdlr

DMA Control Signals Interface.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04090
i2c1	0xFFC05000	0xFFC05090

Module Instance	Base Address	Register Address
i2c2	0xFFC06000	0xFFC06090
i2c3	0xFFC07000	0xFFC07090

Offset: 0x90

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="10">Reserved</td><td colspan="6">dmardl
RW 0x0</td></tr>
</table>

ic_dma_rdlr Fields

Bit	Name	Description	Access	Reset
5:0	dmardl	This bit field controls the level at which a DMA request is made by the receive logic. The watermark level \= DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and RDMAE =1. For instance, when DMARDL is 0, then dma_rx_req is asserted when or more data entries are present in the receive FIFO.	RW	0x0

ic_sda_setup

This register controls the amount of time delay (in terms of number of l4_sp_clk clock periods) introduced in the rising edge of SCL relative to SDA changing by holding SCL low when I2C services a read request while operating as a slave-transmitter. The relevant I2C requirement is tSU:DAT (note 4) as detailed in the I2C Bus Specification. This register must be programmed with a value equal to or greater than 2. Note: The length of setup time is calculated using [(IC_SDA_SETUP - 1) * (l4_sp_clk)], so if the user requires 10 l4_sp_clk periods of setup time, they should program a value of 11. The IC_SDA_SETUP register is only used by the I2C when operating as a slave transmitter.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04094
i2c1	0xFFC05000	0xFFC05094
i2c2	0xFFC06000	0xFFC06094
i2c3	0xFFC07000	0xFFC07094

Offset: 0x94

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								sda_setup RW 0x64							

ic_sda_setup Fields

Bit	Name	Description	Access	Reset
7:0	`sda_setup`	It is recommended that if the required delay is 1000ns, then for an l4_sp_clk frequency of 10 MHz, ic_sda_setup should be programmed to a value of 11.	RW	0x64

ic_ack_general_call

The register controls whether i2c responds with a ACK or NACK when it receives an I2C General Call address.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC04098
i2c1	0xFFC05000	0xFFC05098
i2c2	0xFFC06000	0xFFC06098
i2c3	0xFFC07000	0xFFC07098

Offset: `0x98`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															ack_gen_call RW 0x1

ic_ack_general_call Fields

Bit	Name	Description	Access	Reset
0	`ack_gen_call`	When an ACK is asserted, (by asserting i2c_out_data) when it receives a General call. Otherwise, i2c responds with a NACK (by negating i2c_out_data). **Value** **Description** 0x0 I2C responds with a NACK 0x1 I2C responds with an ACK	RW	0x1

ic_enable_status

This register is used to report the i2c hardware status when the IC_ENABLE register is set from 1 to 0; that is, when i2c is disabled. If IC_ENABLE has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1. If IC_ENABLE has been set to 0, bits 2:1 are only valid as soon as bit 0 is read as '0'. Note: When ic_enable has been written with '0' a delay occurs for bit 0 to be read as '0' because disabling the i2c depends on I2C bus activities.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC0409C
i2c1	0xFFC05000	0xFFC0509C
i2c2	0xFFC06000	0xFFC0609C
i2c3	0xFFC07000	0xFFC0709C

Offset: `0x9C`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													slv_rx_data_lost RO 0x0	slv_disabled_while_busy RO 0x0	ic_en RO 0x0

ic_enable_status Fields

Bit	Name	Description	Access	Reset
2	`slv_rx_data_lost`	This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to the setting of IC ENABLE from 1 to 0. When read as 1, i2c is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the i2c has a chance to NACK a transfer, and ic_enable has been set to 0, then this bit is also set to 1. When read as 0, i2c is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.	RO	0x0
1	`slv_disabled_while_busy`	This bit indicates if a potential or active Slave operation has been aborted due to the setting of the ic_enable register from 1 to 0. This bit is set when the CPU writes a 0 to the ic_enable register while: (a) I2C is receiving the address byte of the Slave-Transmitter operation from a remote master; OR, (b) address and data bytes of the Slave-Receiver operation from a remote master. When read as 1, I2C is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2C address matches the slave address set in i2c (IC_SAR register) OR if the transfer is completed before IC_ENABLE is set to 0 but has not taken effect. NOTE: If the remote I2C master terminates the transfer with a STOP condition before the i2c has a chance to NACK a transfer, and IC_ENABLE has been set to 0, then this bit will also be set to 1. When read as 0, i2c is deemed to have been disabled when there is master activity, or when the I2C bus is idle. NOTE: The CPU can safely read this bit when IC_EN (bit 0) is read as 0.	RO	0x0
0	`ic_en`	This bit always reflects the value driven on the output port ic_en. Not used in current application. When read as 1, i2c is deemed to be in an enabled state. When read as 0, i2c is deemed completely inactive. NOTE: The CPU can safely read this bit anytime. When this bit is read as 0, the CPU can safely read slv_rx_data_lost (bit 2) and slv_disabled_while_busy (bit 1).	RO	0x0

ic_fs_spklen

This register is used to store the duration, measured in ic_clk cycles, of the longest spike that is filtered out by the spike suppression logic when the component is operating in SS or FS modes.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC040A0
i2c1	0xFFC05000	0xFFC050A0
i2c2	0xFFC06000	0xFFC060A0
i2c3	0xFFC07000	0xFFC070A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								spklen RW 0x2							

ic_fs_spklen Fields

Bit	Name	Description	Access	Reset
7:0	spklen	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic. This register can be written only when the I2C interface is disabled, which corresponds to the IC_ENABLE register being set to 0. Writes at other times have no effect. The minimum valid value is 1; hardware prevents values less than this being written, and if attempted results in 2 being set.	RW	0x2

ic_comp_param_1

This is a constant read-only register that contains encoded information about the component's parameter settings.

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC040F4
i2c1	0xFFC05000	0xFFC050F4
i2c2	0xFFC06000	0xFFC060F4
i2c3	0xFFC07000	0xFFC070F4

Offset: 0xF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								tx_buffer_depth RO 0x3F							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rx_buffer_depth RO 0x3F								add_encoded_params RO 0x1	has_dma RO 0x1	intr_io RO 0x1	hc_count_values RO 0x0	max_speed_mode RO 0x2		apb_data_width RO 0x2	

ic_comp_param_1 Fields

Bit	Name	Description	Access	Reset
23:16	tx_buffer_depth	Sets Tx FIFO Depth. **Value** **Description** 0x40 Tx Buffer Depth 64 Entries	RO	0x3F
15:8	rx_buffer_depth	Sets Rx FIFO Depth. **Value** **Description** 0x40 Rx Fifo Depth 64 Entries	RO	0x3F
7	add_encoded_params	By adding in the encoded parameters, this gives firmware an easy and quick way of identifying the DesignWare component within an I/O memory map. Some critical design-time options determine how a driver should interact with the peripheral. There is a minimal area overhead by including these parameters. Allows a single driver to be developed for each component which will be self-configurable. **Value** **Description** 0x1 Add Encoded Params	RO	0x1

Bit	Name	Description	Access	Reset
6	has_dma	This configures the inclusion of DMA handshaking interface signals. **Value** **Description** 0x1 Has DMA	RO	0x1
5	intr_io	All interrupt sources are combined in to a single output. **Value** **Description** 0x1 Combined Interrupt Output	RO	0x1
4	hc_count_values	This makes the *CNT registers readable and writable. **Value** **Description** 0x0 *CNT registers read/write	RO	0x0
3:2	max_speed_mode	The value of this field determines the maximum i2c bus interface speed. **Value** **Description** 0x2 Fast Mode (400 kbit/s)	RO	0x2
1:0	apb_data_width	Sets the APB Data Width. **Value** **Description** 0x2 APB Data Width is 32 Bits	RO	0x2

ic_comp_version

Describes the version of the I2C

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC040F8
i2c1	0xFFC05000	0xFFC050F8
i2c2	0xFFC06000	0xFFC060F8
i2c3	0xFFC07000	0xFFC070F8

Offset: 0xF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ic_comp_version RO 0x3132302A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ic_comp_version RO 0x3132302A															

ic_comp_version Fields

Bit	Name	Description	Access	Reset
31:0	`ic_comp_version`	Specifies I2C release number (encoded as 4 ASCII characters) **Value** **Description** 0x3132302a Version 1.20a	RO	0x31323 02A

ic_comp_type

Describes a unique ASCII value

Module Instance	Base Address	Register Address
i2c0	0xFFC04000	0xFFC040FC
i2c1	0xFFC05000	0xFFC050FC
i2c2	0xFFC06000	0xFFC060FC
i2c3	0xFFC07000	0xFFC070FC

Offset: `0xFC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ic_comp_type RO 0x44570140															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ic_comp_type RO 0x44570140															

ic_comp_type Fields

Bit	Name	Description	Access	Reset
31:0	`ic_comp_type`	Designware Component Type number = 0x44_57_01_40. This assigned unique hex value is constant and is derived from the two ASCII letters 'DW' followed by a 16-bit unsigned number.	RO	0x44570 140

Document Revision History

Table 20-5: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	HPS I^2C Signals for FPGA Routing table updated I^2C interface in FPGA Fabric diagram added Added Address Map and Register Descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting updates Added HPS I^2c Signals for FPGA routing to Interface pins section
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model, address map and register definitions, clocks, reset, and interface pins sections.
January 2012	1.0	Initial release.

UART Controller 21

2014.06.30

cv_54021 Subscribe Send Feedback

The hard processor system (HPS) provides two UART controllers for asynchronous serial communication. The UART controllers are based on an industry standard 16550 UART controller. The UART controllers are instances of the Synopsys® DesignWare® APB Universal Asynchronous Receiver/ Transmitter (DW_apb_uart) peripheral.[44]

UART Controller Features

The UART controller provides the following functionality and features:

- Programmable character properties, such as number of data bits per character, optional parity bits, and number of stop bits †
- Line break generation and detection †
- Direct memory access (DMA) controller interface
- Prioritized interrupt identification †
- Programmable baud rate
- False start bit detection †
- Automatic flow control mode per 16750 standard †
- Internal loopback mode support
- 128-byte transmit and receive FIFO buffers
  - FIFO buffer status registers †
  - FIFO buffer access mode (for FIFO buffer testing) enables write of receive FIFO buffer by master and read of transmit FIFO buffer by master †
- Shadow registers reduce software overhead and provide programmable reset †
- Transmitter holding register empty (THRE) interrupt mode †

[44] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

UART Controller Block Diagram and System Integration

Figure 21-1: UART Block Diagram

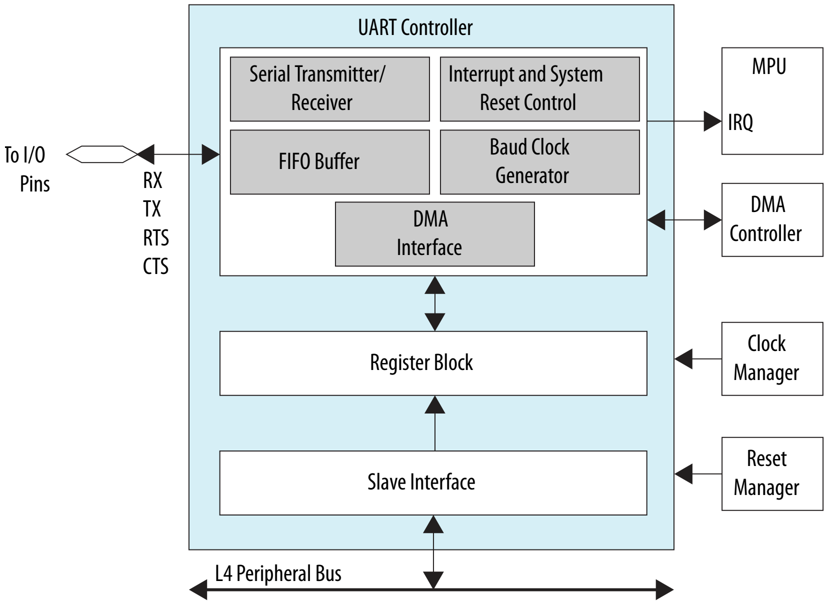


Table 21-1: UART Controller Block Descriptions

Block	Description
Slave interface	Slave interface between the component and L4 peripheral bus.
Register block	Provides main UART control, status, and interrupt generation functions.†
FIFO buffer	Provides FIFO buffer control and storage. †
Baud clock generator	Generates the transmitter and receiver baud clock. With a reference clock of 100 MHz, the UART controller supports transfer rates of 95 baud to 6.25 Mbaud. This supports communication with all known 16550 devices. The baud rate is controlled by programming the interrupt enable or divisor latch high (`IER_DLH`) and receive buffer, transmit holding, or divisor latch low (`RBR_THR_DLL`) registers.

Block	Description
Serial transmitter	Converts parallel data written to the UART into serial data and adds all additional bits, as specified by the control register, for transmission. This makeup of serial data, referred to as a character, exits the block in serial UART. †
Serial receiver	Converts the serial data character (as specified by the control register) received in the UART format to parallel form. Parity error detection, framing error detection and line break detection is carried out in this block. †
DMA interface	The UART controller includes a DMA controller interface to indicate when received data is available or when the transmit FIFO buffer requires data. The DMA requires two channels, one for transmit and one for receive. The UART controller supports single and burst transfers. You can use DMA in FIFO buffer and non-FIFO buffer mode.

Related Information

DMA Controller on page 16-1
For more information, refer to this *DMA Controller* chapter.

Functional Description of the UART Controller

The HPS UART is based on an industry-standard 16550 UART. The UART supports serial communication with a peripheral, modem (data carrier equipment), or data set. The master (CPU) writes data over the slave bus to the UART. The UART converts the data to serial format and transmits to the destination device. The UART also receives serial data and stores it for the master (CPU). †

The UART's registers control the character length, baud rate, parity generation and checking, and interrupt generation. The UART's single interrupt output signal is supported by several prioritized interrupt types that trigger assertion. You can separately enable or disable each of the interrupt types with the control registers. †

FIFO Buffer Support

The UART controller includes 128-byte FIFO buffers to buffer transmit and receive data. FIFO buffer access mode allows the master to write the receive FIFO buffer and to read the transmit FIFO buffer for test purposes. FIFO buffer access mode is enabled with the FIFO access register (FAR). Once enabled, the control portions of the transmit and receive FIFO buffers are reset and the FIFO buffers are treated as empty. †

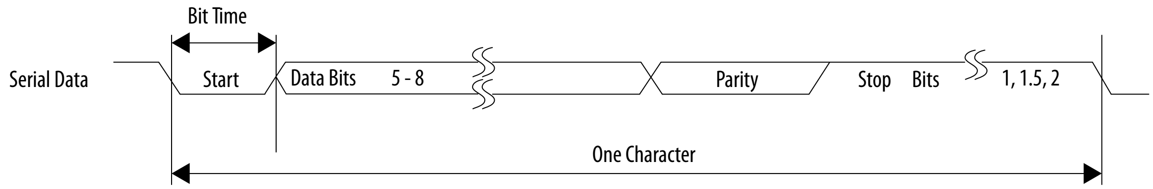
When FIFO buffer access mode is enabled, you can write data to the transmit FIFO buffer as normal; however, no serial transmission occurs in this mode and no data leaves the FIFO buffer. You can read back the data that is written to the transmit FIFO buffer with the transmit FIFO read (TFR) register. The TFR register provides the current data at the top of the transmit FIFO buffer. †

Similarly, you can also read data from the receive FIFO buffer in FIFO buffer access mode. Since the normal operation of the UART is halted in this mode, you must write data to the receive FIFO buffer to read it back. The receive FIFO write (`RFW`) register writes data to the receive FIFO buffer. The upper two bits of the 10-bit register write framing errors and parity error detection information to the receive FIFO buffer. Bit 9 of `RFW` indicates a framing error and bit 8 of `RFW` indicates a parity error. Although you cannot read these bits back from the receive buffer register, you can check the bits by reading the line status register (`LSR`), and by checking the corresponding bits when the data in question is at the top of the receive FIFO buffer. †

UART(RS232) Serial Protocol

Because the serial communication between the UART controller and the selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in below.†

Figure 21-2: Serial Data Format



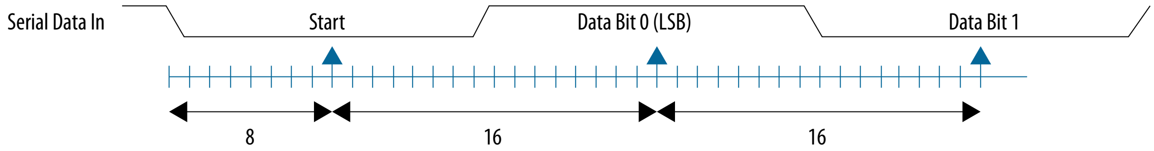
An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to provide the UART controller with the ability to perform simple error checking on the received data.†

The Control Register is used to control the serial character characteristics. The individual bits of the data word are sent after the start bit, starting with the least- significant bit (LSB). These are followed by the optional parity bit, followed by the stop bit(s), which can be 1, 1.5 or 2.†

All the bits in the transmission (with exception to the half stop bit when 1.5 stop bits are used) are transmitted for exactly the same time duration. This is referred to as a Bit Period or Bit Time. One Bit Time equals to 16 baud clocks. To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. As the exact number of baud clocks that each bit was transmitted for is known, calculating the midpoint for sampling is not difficult. That is, every 16 baud clocks after the midpoint sample of the start bit.†

Together with serial input debouncing, this feature also contributes to avoid the detection of false start bits. Short glitches are filtered out by debouncing, and no transition is detected on the line. If a glitch is wide enough to avoid filtering by debouncing, a falling edge is detected. However, a start bit is detected only if the line is sampled low again after half a bit time has elapsed. †

Figure 21-3: Receiver Serial Data Sample Points



The baud rate of the UART controller is controlled by the serial clock and the Divisor Latch Register (DLH and DLL).†

Automatic Flow Control

The UART includes 16750-compatible request-to-send (RTS) and clear-to-send (CTS) serial data automatic flow control mode. You enable automatic flow control with the modem control register (`MCR.AFCE`). †

Automatic RTS mode

Automatic RTS mode becomes active when the following conditions occur: †

- RTS (`MCR.RTS` bit and `MCR.AFCE` bit are both set)
- FIFO buffers are enabled (`FCR.FIFOE` bit is set)

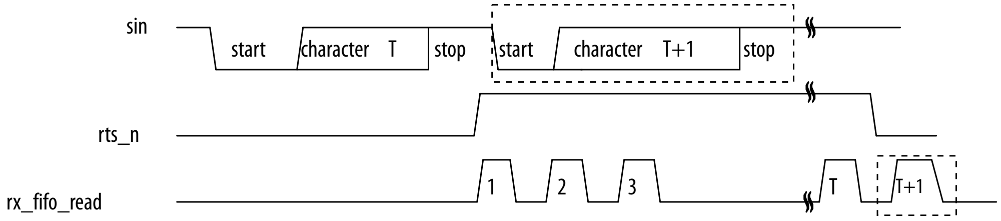
With automatic RTS enabled, the `rts_n` output pin is forced inactive (high) when the FIFO is almost full; where "almost full" refers to two available slots in the FIFO. When `rts_n` is connected to the `cts_n` input pin of another UART device, the other UART stops sending serial data until the receive FIFO buffer has available space (until it is completely empty). †

The selectable receive FIFO buffer threshold values are 1, ¼, ½, and 2 less than full. Because one additional character may be transmitted to the UART after `rts_n` is inactive (due to data already having entered the transmitter block in the other UART), setting the threshold to 2 less than full allows maximum use of the FIFO buffer with a margin of one character. †

Once the receive FIFO buffer is completely emptied by reading the receiver buffer register (`RBR_THR_DLL`), `rts_n` again becomes active (low), signaling the other UART to continue sending data.†

Even when you set the correct MCR bits, if the FIFO buffers are disabled through `FCR.FIFOE`, automatic flow control is also disabled. When auto RTS is not implemented or disabled, `rts_n` is controlled solely by `MCR.RTS`. In the Automatic RTS Timing diagram, the character T is received because `rts_n` is not detected prior to the next character entering the sending UART transmitter. †

Figure 21-4: Automatic RTS Timing



Automatic CTS mode

Automatic CTS mode becomes active when the following conditions occur: †

- AFCE (`MCR.AFCE` bit is set)
- FIFO buffers are enabled (through FIFO buffer control register `IIR_FCR.FIFOE`) bit

When automatic CTS is enabled (active), the UART transmitter is disabled whenever the `cts_n` input becomes inactive (high). This prevents overflowing the FIFO buffer of the receiving UART. †

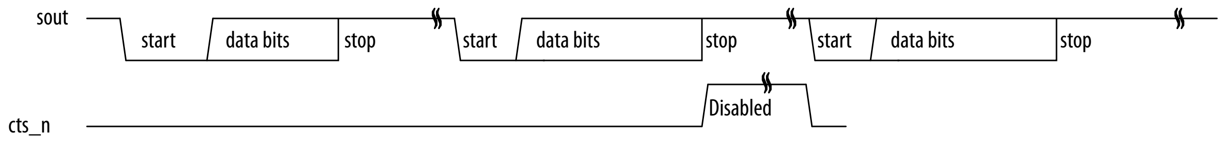
If the `cts_n` input is not deactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, you can continue to write and even overflow to the transmit FIFO buffer. †

Automatic CTS mode requires the following sequence:

1. The UART status register are read to verify that the transmit FIFO buffer is full (UART status register `USR.TFNF` set to zero). †
2. The current FIFO buffer level is read via the transmit FIFO level (`TFL`) register. †
3. Programmable THRE interrupt mode must be enabled to access the FIFO buffer full status from the LSR. †

When using the FIFO buffer full status, software can poll this before each write to the transmit FIFO buffer. When the `cts_n` input becomes active (low) again, transmission resumes. If the FIFO buffers are disabled with the `FCR.FIFOE` bit, automatic flow control is also disabled regardless of any other settings. When auto CTS is not implemented or disabled, the transmitter is unaffected by `cts_n`.†

Figure 21-5: Automatic CTS Timing



Interface Pins

Table 21-2: Interface Pins

Pin	Width	Direction	Description
RX	1 bit	Input	Serial Input
TX	1 bit	Output	Serial Output
CTS	1 bit	Input	Clear to send
RTS	1 bit	Output	Request to send

FPGA Routing

Table 21-3: Signals for FPGA Routing

Signal	Width	Direction	Description
uart_rxd	1 bit	Input	Serial input
uart_txd	1 bit	Output	Serial output
uart_cts	1 bit	Input	Clear to send

Signal	Width	Direction	Description
uart_rts	1 bit	Output	Request to send
uart_dsr	1 bit	Input	Data set ready
uart_dcd	1 bit	Input	Data carrier detect
uart_ri	1 bit	Input	Ring indicator
uart_dtr	1 bit	Output	Data terminal ready
uart_out1_n	1 bit	Output	User defined output 1
uart_out2_n	1 bit	Output	User defined output 2

Clocks

The UART controller is connected to the `l4_sp_clk` clock. The clock input is driven by the clock manager.

Related Information

Clock Manager on page 2-1
For more information, refer to the *Clock Manager* chapter.

Resets

The UART controller is connected to the `uart_rst_n` reset signal. The reset manager drives the signal on a cold or warm reset.

Related Information

Reset Manager on page 3-1
For more information, refer to the *Reset Manager* chapter.

Interrupts

The assertion of the UART interrupt output signal occurs when one of the following interrupt types are enabled and active: †

Table 21-4: Interrupt Types and Priority †

Interrupt Type	Priority	Source	Interrupt Reset Control
Receiver line status	Highest	Overrun, parity and framing errors, break condition.	Reading the line status Register.

Interrupt Type	Priority	Source	Interrupt Reset Control
Received data available	Second	Receiver data available (FIFOs disabled) or RCVR FIFO trigger level reached (FIFOs enabled).	Reading the receiver buffer register (FIFOs disabled) or the FIFO drops below the trigger level (FIFOs enabled)
Character timeout indication	Second	No characters in or out of the Receive FIFO during the last 4 character times and there is at least 1 character in it during this Time.	Reading the receiver buffer Register.
Transmit holding register empty	Third	Transmitter holding register empty (Programmable THRE Mode disabled) or Transmit FIFO at or below threshold (Programmable THRE Mode enabled).	Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or Programmable THRE Mode not enabled) or Transmit FIFO above threshold (FIFOs and Programmable THRE Mode enabled).
Modem Status	Fourth	Clear to send or data set ready or ring indicator or data carrier detect. If auto flow control mode is enabled, a change in CTS (that is, DCTS set) does not cause an interrupt.	Reading the Modem status Register.

You can enable the interrupt types with the interrupt enable register (`IER_DLH`).

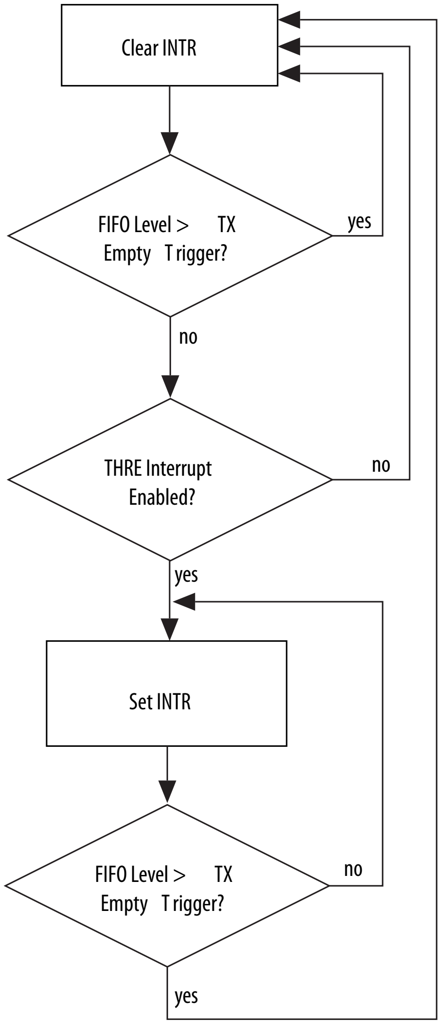
Note: Received Data Available" and "Character Timeout Indication" are enabled by a single bit in the IER_DLH register, because they have the same priority.

Once an interrupt is signaled, you can determine the interrupt source by reading the Interrupt Identity Register (IIR).

Programmable THRE Interrupt

The UART has a programmable THRE interrupt mode to increase system performance. You enable the programmable THRE interrupt mode with the interrupt enable register (`IER_DLH.PTIME`). When the THRE mode is enabled, THRE interrupts and the `dma_tx_req` signal are active at and below a programmed transmit FIFO buffer empty threshold level, as shown in the flowchart. †

Figure 21-6: Programmable THRE Interrupt

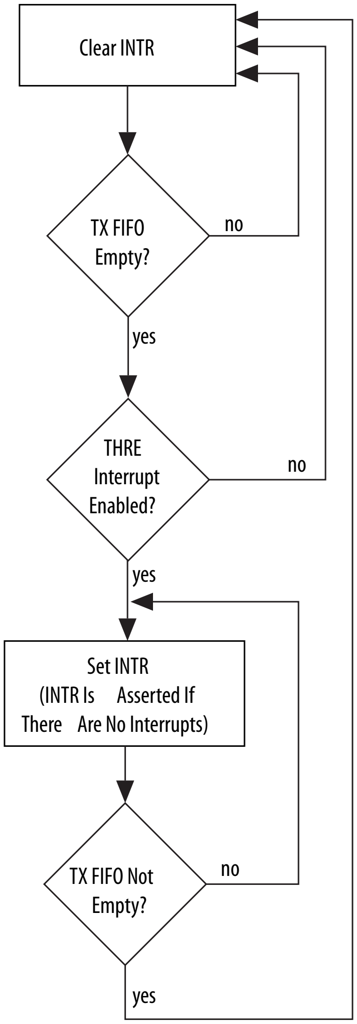

Clear INTR
FIFO Level > TX Empty Trigger?
yes
no
THRE Interrupt Enabled?
no
yes
Set INTR
FIFO Level > TX Empty Trigger?
no
yes

The threshold level is programmed into `FCR.TET`. The available empty thresholds are empty, 2, ¼, and ½. The optimum threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should prove optimum in increasing system performance by preventing the transmit FIFO buffer from running empty.

In addition to the interrupt change, line status register (`LSR.THRE`) also switches from indicating that the transmit FIFO buffer is empty, to indicating that the FIFO buffer is full. This change allows software to fill the FIFO buffer for each transmit sequence by polling `LSR.THRE` before writing another character. This directs the UART to fill the transmit FIFO buffer whenever an interrupt occurs and there is data to transmit, instead of waiting until the FIFO buffer is completely empty. Waiting until the FIFO buffer is empty reduces performance whenever the system is too busy to respond immediately. You can increase system efficiency when this mode is enabled in combination with automatic flow control.

When not selected or disabled, THRE interrupts and `LSR.THRE` function normally, reflecting an empty THR or FIFO buffer.

Figure 21-7: Interrupt Generation without Programmable THRE Interrupt Mode



DMA Controller Operation

The UART controller includes a DMA controller interface to indicate when the receive FIFO buffer data is available or when the transmit FIFO buffer requires data. The DMA requires two channels, one for transmit and one for receive. The UART controller supports both single and burst transfers.

The FIFO buffer depth (`FIFO_DEPTH`) for both the RX and TX buffers in the UART controller is 128 entries.

Related Information

DMA Controller on page 16-1
For more information, refer to this *DMA Controller* chapter.

Transmit FIFO Underflow

During UART serial transfers, transmit FIFO requests are made to the DMA controller whenever the number of entries in the transmit FIFO is less than or equal to the decoded level of the Transmit Empty

Trigger (TET) field in the FIFO Control Register (FCR), also known as the watermark level. The DMA controller responds by writing a burst of data to the transmit FIFO buffer, of length specified as DMA burst length. †

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously, that is, when the FIFO begins to empty, another DMA request should be triggered. Otherwise, the FIFO will run out of data (underflow) causing a STOP to be inserted on the UART bus. To prevent this condition, you must set the watermark level correctly. †

Related Information

DMA Controller on page 16-1
For more information, refer to this *DMA Controller* chapter.

Transmit Watermark Level

Consider the example where the following assumption is made: †

DMA burst length = `FIFO_DEPTH` - decoded watermark level of `IIR_FCR.TET` †

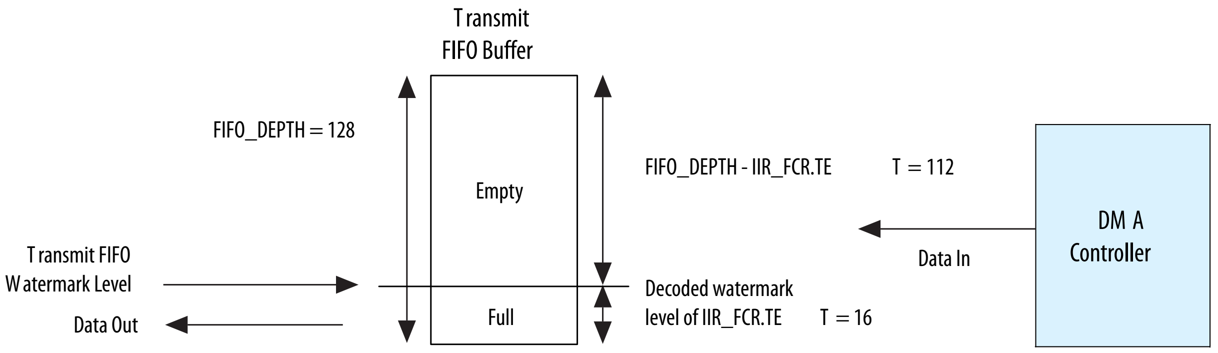
Here the number of data items to be transferred in a DMA burst is equal to the empty space in the transmit FIFO. Consider the following two different watermark level settings: †

IIR_FCR.TET = 1

`IIR_FCR.TET` = 1 decodes to a watermark level of 16.

- Transmit FIFO watermark level = decoded watermark level of `IIR_FCR.TET` = 16 †
- DMA burst length = `FIFO_DEPTH` - decoded watermark level of `IIR_FCR.TET` = 112 †
- UART transmit `FIFO_DEPTH` = 128 †
- Block transaction size =R** 448†

Figure 21-8: Transmit FIFO Watermark Level = 16



The number of burst transactions needed equals the block size divided by the number of data items per burst:

Block transaction size/DMA burst length = 448/112 = 4

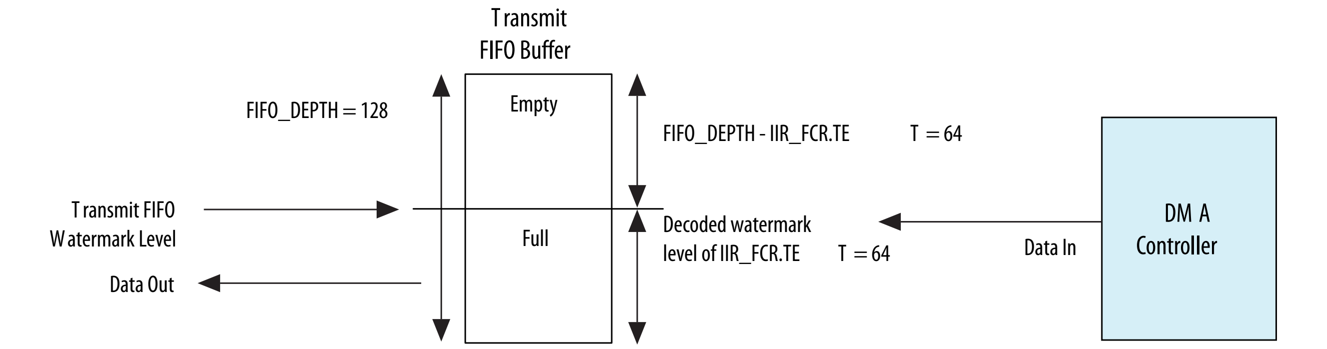
The number of burst transactions in the DMA block transfer is 4. But the watermark level, decoded level of `IIR_FCR.TET`, is quite low. Therefore, the probability of transmit underflow is high where the UART serial transmit line needs to transmit data, but there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the FIFO becomes empty.

IIR_FCR.TET = 3

`IIR_FCR.TET` = 3 decodes to a watermark level of 64.

- Transmit FIFO watermark level = decoded watermark level of `IIR_FCR.TET` = 64 †
- DMA burst length = `FIFO_DEPTH` - decoded watermark level of `IIR_FCR.TET` = 64†
- UART transmit `FIFO_DEPTH` = 128 †
- Block transaction size = 448 †

Figure 21-9: Transmit FIFO Watermark Level = 64



Number of burst transactions in block: †

Block transaction size/DMA burst length = 448/64 = 7 †

In this block transfer, there are 15 destination burst transactions in a DMA block transfer. But the watermark level, decoded level of `IIR_FCR.TET`, is high. Therefore, the probability of UART transmit underflow is low because the DMA controller has plenty of time to service the destination burst transaction request before the UART transmit FIFO becomes empty. †

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of bursts per block and worse bus utilization than the former case. †

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of the rate at which the UART transmits data to the rate at which the DMA can respond to destination burst requests. †

Transmit FIFO Overflow

Setting the DMA burst length to a value greater than the watermark level that triggers the DMA request might cause overflow when there is not enough space in the transmit FIFO to service the destination burst request. Therefore, the following equation must be adhered to in order to avoid overflow: †

DMA burst length <= `FIFO_DEPTH` - decoded watermark level of `IIR_FCR.TET`

In case 2: decoded watermark level of `IIR_FCR.TET` = 64, the amount of space in the transmit FIFO at the time of the burst request is made is equal to the DMA burst length. Thus, the transmit FIFO may be full, but not overflowed, at the completion of the burst transaction. †

Therefore, for optimal operation, DMA burst length should be set at the FIFO level that triggers a transmit DMA request; that is: †

DMA burst length = `FIFO_DEPTH` - decoded watermark level of `IIR_FCR.TET`

Adhering to this equation reduces the number of DMA bursts needed for block transfer, and this in turn improves bus utilization. †

The transmit FIFO will not be full at the end of a DMA burst transfer if the UART controller has successfully transmitted one data item or more on the UART serial transmit line during the transfer. †

Receive FIFO Overflow

During UART serial transfers, receive FIFO requests are made to the DMA whenever the number of entries in the receive FIFO is at or above the decoded level of Receive Trigger (`RT`) field in the FIFO Control Register (`IIR_FCR`). This is known as the watermark level. The DMA responds by fetching a burst of data from the receive FIFO. †

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously, that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise the FIFO will fill with data (overflow). To prevent this condition, the user must set the watermark level correctly. †

Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level, decoded watermark level of `IIR_FCR.RT`, should be set to minimize the probability of overflow, as shown in the Receive FIFO Buffer diagram. It is a tradeoff between the number of DMA burst transactions required per block versus the probability of an overflow occurring. †

Receive FIFO Underflow

Setting the source transaction burst length greater than the watermark level can cause underflow where there is not enough data to service the source burst request. Therefore, the following equation must be adhered to avoid underflow: †

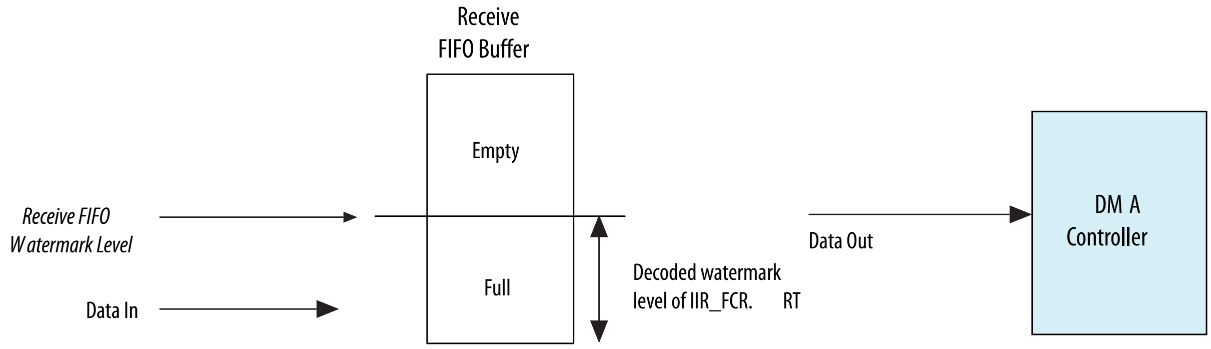
DMA burst length = decoded watermark level of `IIR_FCR.RT` + 1

If the number of data items in the receive FIFO is equal to the source burst length at the time of the burst request is made, the receive FIFO may be emptied, but not underflowed, at the completion of the burst transaction. For optimal operation, DMA burst length should be set at the watermark level, decoded watermark level of `IIR_FCR.RT`. †

Adhering to this equation reduces the number of DMA bursts in a block transfer, which in turn can avoid underflow and improve bus utilization. †

The receive FIFO will not be empty at the end of the source burst transaction if the UART controller has successfully received one data item or more on the UART serial receive line during the burst. †

Figure 21-10: Receive FIFO Buffer



UART Controller Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- UART Module 0
- UART Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

UART Module Address Map

Registers in the UART module

Module Instance	Base Address
uart0	0xFFC02000
uart1	0xFFC03000

UART Module

Register	Offset	Width	Access	Reset Value	Description
rbr_thr_dll on page 21-15	0x0	32	RW	0x0	Rx Buffer, Tx Holding, and Divisor Latch Low
ier_dlh on page 21-16	0x4	32	RW	0x0	Interrupt Enable and Divisor Latch High
iir on page 21-18	0x8	32	RO	0x1	Interrupt Identity Register (when read)
fcr on page 21-19	0x8	32	WO	0x0	FIFO Control (when written)
lcr on page 21-22	0xC	32	RW	0x0	Line Control Register (When Written)
mcr on page 21-24	0x10	32	RW	0x0	Modem Control Register
lsr on page 21-26	0x14	32	RO	0x60	Line Status Register
msr on page 21-29	0x18	32	RO	0x0	Modem Status Register
scr on page 21-33	0x1C	32	RW	0x0	Scratchpad Register
srbr on page 21-33	0x30	32	RW	0x0	Shadow Receive Buffer Register
sthr on page 21-34	0x34	32	RW	0x0	Shadow Transmit Buffer Register
far on page 21-35	0x70	32	RW	0x0	FIFO Access Register
tfr on page 21-36	0x74	32	RO	0x0	Transmit FIFO Read Register
RFW on page 21-37	0x78	32	WO	0x0	Receive FIFO Write

Register	Offset	Width	Access	Reset Value	Description
usr on page 21-37	0x7C	32	RO	0x6	UART Status Register
tfl on page 21-39	0x80	32	RO	0x0	Transmit FIFO Level
rfl on page 21-39	0x84	32	RO	0x0	Receive FIFO Level Write
srr on page 21-40	0x88	32	WO	0x0	Software Reset Register
srts on page 21-41	0x8C	32	RW	0x0	Shadow Request to Send
sbcr on page 21-42	0x90	32	RW	0x0	Shadow Break Control Register
sdmam on page 21-43	0x94	32	RW	0x0	Shadow DMA Mode
sfe on page 21-44	0x98	32	RW	0x0	Shadow FIFO Enable
srt on page 21-45	0x9C	32	RW	0x0	Shadow Rx Trigger
stet on page 21-45	0xA0	32	RW	0x0	Shadow Tx Empty Trigger
htx on page 21-46	0xA4	32	RW	0x0	Halt Tx
dmasa on page 21-47	0xA8	32	WO	0x0	DMA Software Acknowledge
cpr on page 21-47	0xF4	32	RO	0x373F32	Component Parameter Register
ucv on page 21-50	0xF8	32	RO	0x3331312A	Component Version
ctr on page 21-51	0xFC	32	RO	0x44570110	Component Type Register

rbr_thr_dll

This is a multi-function register. This register holds receives and transmit data and controls the least-signficant 8 bits of the baud rate divisor.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02000
uart1	0xFFC03000	0xFFC03000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								value RW 0x0							

rbr_thr_dll Fields

Bit	Name	Description	Access	Reset
7:0	value	Receive Buffer Register: This register contains the data byte received on the serial input port (uart_rxd). The data in this register is valid only if the Data Ready (bit [0] in the Line Status Register(LSR)) is set to 1. If FIFOs are disabled(bit[0] of Register FCR is set to 0) the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled(bit [0] of Register FCR is set to 1) this register accesses the head of the receive FIFO. If the receive FIFO is full, and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur. Transmit Holding Register: This register contains data to be transmitted on the serial output port. Data should only be written to the THR when the THR Empty bit [5] of the LSR Register is set to 1. If FIFOs are disabled (bit [0] of Register FCR) is set to 0 and THRE is set to 1, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled bit [0] of Register FCR is set to 1 and THRE is set up to 128 characters of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost. Divisor Latch Low: This register makes up the lower 8-bits of a 16-bit, Read/ write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit [7] of the LCR Register is set to 1. The output baud rate is equal to the serial clock l4_sp_clk frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor) Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 l4_sp_clk clock cycles should be allowed to pass before transmitting or receiving data.	RW	0x0

ier_dlh

This is a multi-function register. This register enables/disables receive and transmit interrupts and also controls the most-significant 8-bits of the baud rate divisor. Divisor Latch High Register: This register is accessed when the DLAB bit [7] of the LCR Register is set to 1.Bits[7:0] contain the high order 8-bits of the baud rate divisor.The output baud rate is equal to the serial clock l4_sp_clk frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor): Note that with the Divisor Latch Registers (DLLand DLH) set to zero, the baud clock is disabled and no serial communications will occur. Also, once the DLL is set, at least 8 l4_sp_clk clock cycles should be allowed to pass before transmitting or receiving data. Interrupt Enable Register: This register may only be accessed when the DLAB bit [7] of the LCR Register is set to 0.Allows control of the Interrupt Enables for transmit and receive functions.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02004
uart1	0xFFC03000	0xFFC03004

Offset: 0x4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ptime_dlh7 RW 0x0	dlh6 RW 0x0	dlh5 RW 0x0	dlh4 RW 0x0	edssi_dhl3 RW 0x0	elsi_dhl2 RW 0x0	etbei_dlhl RW 0x0	erbfi_dlh0 RW 0x0

ier_dlh Fields

Bit	Name	Description	Access	Reset
7	ptime_dlh7	Divisor Latch High Register: Bit 7 of DLH value. Interrupt Enable Register: This is used to enable/ disable the generation of THRE Interrupt. **Value** **Description** 0x0 disable tx-hold-reg-empty interrupt 0x1 enable tx-hold-reg-empty interrupt	RW	0x0
6	dlh6	Bit 6 of DLH value.	RW	0x0
5	dlh5	Bit 5 of DLH value.	RW	0x0
4	dlh4	Bit 4 of DLH value.	RW	0x0

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>3</td><td>edssi_dhl3</td><td>Divisor Latch High Register: Bit 3 of DLH value. Interrupt Enable Register: This is used to enable/disable the generation of Modem Status Interrupts. This is the fourth highest priority interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>disable modem status interrupt</td></tr>
<tr><td>0x1</td><td>enable modem status interrupt</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>2</td><td>elsi_dhl2</td><td>Divisor Latch High Register: Bit 2 of DLH value. Interrupt Enable Register: This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Disable interrupt line stat</td></tr>
<tr><td>0x1</td><td>Enable interrupt line stat</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>1</td><td>etbei_dlhl</td><td>Divisor Latch High Register: Bit 1 of DLH value. Interrupt Enable Register: Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Tx disable</td></tr>
<tr><td>0x1</td><td>Tx enable</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
<tr><td>0</td><td>erbfi_dlh0</td><td>Divisor Latch High Register: Bit 0 of DLH value. Interrupt Enable Register: Used to enable/disable the generation of the Receive Data Available Interrupt and the Character Timeout Interrupt(if FIFO's enabled). These are the second highest priority interrupts.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Interrupt Disable</td></tr>
<tr><td>0x1</td><td>Interrupt Enable</td></tr>
</table></td><td>RW</td><td>0x0</td></tr>
</table>

iir

Returns interrupt identification and FIFO enable/disable when read.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02008
uart1	0xFFC03000	0xFFC03008

Offset: 0x8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								fifoen RO 0x0		Reserved		id RO 0x1			

iir Fields

Bit	Name	Description	Access	Reset
7:6	fifoen	This is used to indicate whether the FIFO's are enabled or disabled. **Value** — **Description** 0x0 — FIFO disabled 0x3 — FIFO enabled	RO	0x0
3:0	id	This indicates the highest priority pending interrupt. **Value** — **Description** 0x0 — Modem status 0x1 — No Interrupt pending 0x2 — THR empty 0x4 — Receive data available 0x6 — Receive line status 0xc — Character timeout	RO	0x1

fcr

Controls FIFO Operations when written.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02008
uart1	0xFFC03000	0xFFC03008

fcr

Offset: `0x8`

Access: `WO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rt WO 0x0		tet WO 0x0		dmam WO 0x0	xfifor WO 0x0	rfifor WO 0x0	fifoe WO 0x0

fcr Fields

Bit	Name	Description	Access	Reset
7:6	rt	This register is configured to implement FIFOs. Bits[7:6], Rx Trigger (or RT): This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. In auto flow control mode it is used to determine when the uart_rts_n signal will be de-asserted. It also determines when the uart_dma_rx_req_n signal will be asserted when in certain modes of operation. **Value** — **Description** 0x0 — one character in fifo 0x1 — FIFO 1/4 full 0x2 — FIFO 1/2 full 0x3 — FIFO 2 less than full	WO	0x0
5:4	tet	This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. It also determines when the uart DMA transmit request signal uart_dma_tx_req_n will be asserted when in certain modes of operation. **Value** — **Description** 0x0 — FIFO empty 0x1 — Two characters in FIFO 0x2 — FIFO 1/4 full 0x3 — FIFO 1/2 full	WO	0x0

Bit	Name	Description	Access	Reset
3	dmam	This determines the DMA signalling mode used for the uart_dma_tx_req_n and uart_dma_rx_req_n output signals when additional DMA handshaking signals are not selected. DMA mode 0 supports single DMA data transfers at a time. In mode 0, the uart_dma_tx_req_n signal goes active low under the following conditions: -When the Transmitter Holding Register is empty in non-FIFO mode. -When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled. -When the transmitter FIFO is at or below the programmed threshold with Programmable THRE interrupt mode enabled. It goes inactive under the following conditions -When a single character has been written into the Transmitter Holding Register or transmitter FIFO with Programmable THRE interrupt mode disabled. -When the transmitter FIFO is above the threshold with Programmable THRE interrupt mode enabled. DMA mode 1 supports multi-DMA data transfers, where multiple transfers are made continuously until the receiver FIFO has been emptied or the transmit FIFO has been filled. In mode 1 the uart_dma_tx_req_n signal is asserted under the following conditions: -When the transmitter FIFO is empty with Programmable THRE interrupt mode disabled. -When the transmitter FIFO is at or below the programmed threshold with Programmable THRE interrupt mode enabled. **Value** — **Description** 0x0 — Single DMA Transfer Mode 0x1 — Multiple DMA Transfer Mode	WO	0x0
2	xfifor	Resets the control portion of the transmit FIFO and treats the FIFO as empty. This will also de-assert the DMA Tx request and single signals when additional DMA handshaking is used. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. **Value** — **Description** 0x0 — No Reset of Tx FIFO Control 0x1 — Resets Tx FIFO Control	WO	0x0

Bit	Name	Description	Access	Reset
1	rfifor	Resets the control portion of the receive FIFO and treats the FIFO as empty. This will also de-assert the DMA Rxrequest and single signals. Note that this bit is self-clearing' and it is not necessary to clear this bit. **Value** — **Description** 0x0 — No Reset of Rx FIFO Control 0x1 — Resets of Rx FIFO Control	WO	0x0
0	fifoe	Enables/disables the transmit (Tx) and receive (Rx) FIFO's. Whenever the value of this bit is changed both the Tx and Rx controller portion of FIFO's will be reset. **Value** — **Description** 0x0 — FIFOs disabled 0x1 — FIFOs enabled	WO	0x0

lcr

Formats serial data.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC0200C
uart1	0xFFC03000	0xFFC0300C

Offset: 0xC

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td>dlab
RW
0x0</td><td>break
RW
0x0</td><td>Reserved</td><td>eps
RW
0x0</td><td>pen
RW
0x0</td><td>stop
RW
0x0</td><td colspan="2">dls
RW 0x0</td></tr>
</table>

lcr Fields

Bit	Name	Description	Access	Reset
7	`dlab`	Used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.	`RW`	`0x0`
6	`break`	This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.	`RW`	`0x0`
4	`eps`	This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic '1's is transmitted or checked. If set to zero, an odd number of logic '1's is transmitted or checked. **Value** — **Description** 0x0 — odd parity 0x1 — even parity	`RW`	`0x0`
3	`pen`	This bit is used to enable and disable parity generation and detection in a transmitted and received data character. **Value** — **Description** 0x0 — parity disabled 0x1 — parity enabled	`RW`	`0x0`
2	`stop`	Number of stop bits. Used to select the number of stop bits per character that the peripheral will transmit and receive.Note that regardless of the number of stop bits selected the receiver will only check the first stop bit. **Value** — **Description** 0x0 — one stop bit 0x1 — 1.5 stop bits when DLS (LCR[1:0]) is zero	`RW`	`0x0`

Bit	Name	Description	Access	Reset
1:0	dls	Data Length Select.Selects the number of data bits per character that the peripheral will transmit and receive. **Value** **Description** 0x0 5 bits 0x1 6 bits 0x2 7 bits 0x3 8 bits	RW	0x0

mcr

Reports various operations of the modem signals

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02010
uart1	0xFFC03000	0xFFC03010

Offset: 0x10

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										afce RW 0x0	loopback RW 0x0	out2 RW 0x0	out1 RW 0x0	rts RW 0x0	dtr RW 0x0

mcr Fields

Bit	Name	Description	Access	Reset
5	afce	When FIFOs are enabled, the Auto Flow Control enable bits are active. **Value** **Description** 0x0 Auto Flow Control Mode disabled 0x1 Auto Flow Control Mode enabled	RW	0x0

Bit	Name	Description	Access	Reset
4	`loopback`	This is used to put the UART into a diagnostic mode for test purposes. If UART mode is NOT active, bit [6] of the modem control register MCR is set to zero, data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loopback mode, the modem control inputs (uart_dsr_n, uart_cts_n, uart_ri_n, uart_dcd_n) are disconnected and the modem control outputs (uart_dtr_n, uart_rts_n, uart_out1_n, uart_out2_n) are loopedback to the inputs, internally.	`RW`	`0x0`
3	`out2`	This is used to directly control the user-designated uart_out2_n output. The value written to this location is inverted and driven out on uart_out2_n Note: In Loopback mode bit 4 of the modem control register (MCR) is set to one, the uart_out2_n output is held inactive high while the value of this location is internally looped back to an input. **Value** — **Description** 0x0 — uart_out2_n de-asserted (logic 1) 0x1 — uart_out2_n asserted (logic 0)	`RW`	`0x0`
2	`out1`	The value written to this location is inverted and driven out on uart_out1_n pin. Note that in Loopback mode (MCR[4] set to one), the uart_out1_n output is held inactive high while the value of this location is internally looped back to an input. **Value** — **Description** 0x0 — uart_out1_n de-asserted (logic 1) 0x1 — uart_out1_n asserted (logic 0)	`RW`	`0x0`

Bit	Name	Description	Access	Reset
1	rts	This is used to directly control the Request to Send (uart_rts_n) output. The Request to Send (uart_rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto RTS Flow Control is not enabled (MCR[5] set to zero), the uart_rts_n signal is set low by programming MCR[1] (RTS) to a high. If Auto Flow Control is active (MCR[5] set to one) and FIFO's enable (FCR[0] set to one), the uart_rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (uart_rts_n is inactive high when above the threshold). The uart_rts_n signal will be de-asserted when MCR[1] is set low. Note that in Loopback mode (MCR[4] set to one), the uart_rts_n output is held inactive high while the value of this location is internally looped back to an input. **Value** / **Description** 0x0 uart_rts_n de-asserted (logic 1) 0x1 uart_rts_n asserted (logic 0)	RW	0x0
0	dtr	This is used to directly control the Data Terminal Ready output. The value written to this location is inverted and driven out on uart_dtr_n, that is: The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that Loopback mode bit [4] of MCR is set to one, the uart_dtr_n output is held inactive high while the value of this location is internally looped back to an input. **Value** / **Description** 0x0 uart_dtr_n de-asserted (logic 1) 0x1 uart_dtr_n asserted (logic 0)	RW	0x0

lsr

Reports status of transmit and receive.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02014
uart1	0xFFC03000	0xFFC03014

Offset: 0x14

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								rfe RO 0x0	temt RO 0x1	thre RO 0x1	bi RO 0x0	fe RO 0x0	pe RO 0x0	oe RO 0x0	dr RO 0x0

lsr Fields

Bit	Name	Description	Access	Reset
7	rfe	This bit is only relevant when FIFO's are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO. **Value** **Description** 0x0 no error in Rx FIFO 0x1 error in Rx FIFO	RO	0x0
6	temt	If in FIFO mode and FIFO's enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFO's are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty. **Value** **Description** 0x0 Transmit Empty not set 0x1 Transmit Empty set	RO	0x1
5	thre	If THRE mode is disabled (IER[7] set to zero) this bit indicates that the THR or Tx FIFO is empty. This bit is set whenever data is transferred from the THR or Tx FIFO to the transmitter shift register and no new data has been written to the THR or Tx FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If both THRE and FIFOs are enabled, both (IER[7] set to one and FCR[0] set to one respectively), the functionality will indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] thresholdsetting.	RO	0x1

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>4</td><td>bi</td><td>This is used to indicate the detection of a break sequence on the serial input data. Set whenever the serial input, sin, is held in a logic 0 state for longer than the sum of start time + data bits + parity + stop bits. A break condition on serial input causes one and only one character, consisting of all zeros, to be received by the UART. The character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BI bit.</td><td>RO</td><td>0x0</td></tr>
<tr><td>3</td><td>fe</td><td>This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs the UART will try to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit i.e. data, and/or parity and stop. It should be noted that the Framing Error (FE) bit(LSR[3]) will be set if a break interrupt has occurred, as indicated by a Break Interrupt BIT bit (LSR[4]). Reading the LSR clears the FE bit.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>no framing error</td></tr>
<tr><td>0x1</td><td>framing error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
<tr><td>2</td><td>pe</td><td>This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set. Since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO. It should be noted that the Parity Error (PE) bit (LSR[2]) will be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) bit (LSR[4]). Reading the LSR clears the PE bit.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>no parity error</td></tr>
<tr><td>0x1</td><td>no parity error</td></tr>
</table></td><td>RO</td><td>0x0</td></tr>
</table>

Bit	Name	Description	Access	Reset
1	oe	This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. In the non-FIFO mode, the OE bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost.Reading the LSR clears the OE bit. **Value** — **Description** 0x0 — no overrun error 0x1 — overrun error	RO	0x0
0	dr	This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. This bit is cleared when the RBR is read in the non-FIFO mode, or when the receiver FIFO is empty, in the FIFO mode. **Value** — **Description** 0x0 — no data ready 0x1 — data ready	RO	0x0

msr

It should be noted that whenever bits 0, 1, 2 or 3 are set to logic one, to indicate a change on the modem control inputs, a modem status interrupt will be generated if enabled via the IER regardless of when the change occurred. Since the delta bits (bits 0, 1, 3) can get set after a reset if their respective modem signals are active (see individual bits for details), a read of the MSR after reset can be performed to prevent unwanted interrupts.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02018
uart1	0xFFC03000	0xFFC03018

Offset: 0x18

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								dcd RO 0x0	ri RO 0x0	dsr RO 0x0	cts RO 0x0	ddcd RO 0x0	teri RO 0x0	ddsr RO 0x0	dcts RO 0x0

msr Fields

Bit	Name	Description	Access	Reset
7	dcd	This is used to indicate the current state of the modem control line uart_dcd_n. That is this bit is the complement uart_dcd_n. When the Data Carrier Detect input (uart_dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set. In Loopback Mode (MCR[4] set to one), DCD is the same as MCR[3] (uart_out2). **Value** **Description** 0x0 uart_dcd_n input is de-asserted (logic 1) 0x1 uart_dcd_n input is asserted (logic 0)	RO	0x0
6	ri	This bit is used to indicate the current state of the modem control line uart_ri_n. That is this bit is the complement uart_ri_n. When the Ring Indicator input (uart_ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set. In Loopback Mode bit [4] of register MCR set to one, RI is the same as bit [2] uart_out1_n of register MCR. **Value** **Description** 0x0 uart_ri_n input is de-asserted (logic 1) 0x1 uart_ri_n input is asserted (logic 0)	RO	0x0

Bit	Name	Description	Access	Reset
5	dsr	This is used to indicate the current state of the modem control line uart_dsr_n. That is this bit is the complement f uart_dsr_n. When the Data Set Ready input (uart_dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the uart. In Loopback Mode bit [4] of register MCR is set to one, DSR is the same as bit [0] (DTR) of register MCR. **Value** **Description** 0x0 uart_dsr_n input is de-asserted (logic 1) 0x1 uart_dsr_n input is asserted (logic 0)	RO	0x0
4	cts	This is used to indicate the current state of the modem control line uart_cts_n. That is, this bit is the complement uart_cts_n. When the Clear to Send input (uart_cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the uart. In Loopback Mode bit [4] of register MCR is set to one, CTS is the same as bit [1] RTS of register MCR. **Value** **Description** 0x0 uart_cts_n input is de-asserted (logic 1) 0x1 uart_cts_n input is asserted (logic 0)	RO	0x0
3	ddcd	This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read. Reading the MSR clears the DDCD bit. In Loopback Mode bit [4] of register MCR is set to one, DDCD reflects changes bit [3] uart_out2 of register MCR. Note: If the DDCD bit is not set and the uart_dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD bit will get set when the reset is removed if the uart_dcd_n signal remains asserted. **Value** **Description** 0x0 no change on uart_dcd_n since last read of MSR 0x1 change on uart_dcd_n since last read of MSR	RO	0x0

Bit	Name	Description	Access	Reset
2	`teri`	This is used to indicate that a change on the input uart_ri_n (from an active low, to an inactive high state) has occurred since the last time the MSR was read. Reading the MSR clears the TERI bit. In Loopback Mode bit [4] of register MCR is set to one, TERI reflects when bit [2] of register MCR has changed state from a high to a low. **Value** — **Description** 0x0 — no change on uart_ri_n since last read of MSR 0x1 — change on uart_ri_n since last read of MSR	RO	0x0
1	`ddsr`	This is used to indicate that the modem control line uart_dsr_n has changed since the last time the MSR was read. Reading the MSR clears the DDSR bit.In Loopback Mode (MCR[4] set to one), DDSR reflects changes on bit [0] DTR of register MCR . Note, if the DDSR bit is not set and the uart_dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit will get set when the reset is removed if the uart_dsr_n signal remains asserted. **Value** — **Description** 0x0 — no change on uart_dsr_n since last read of MSR 0x1 — change on uart_dsr_n since last read of MSR	RO	0x0
0	`dcts`	This is used to indicate that the modem control line uart_cts_n has changed since the last time the MSR was read. That is: Reading the MSR clears the DCTS bit. In Loopback Mode bit [4] of MCR set to one, DCTS reflects changes on bit [1] RTS of register MCR. Note: If the DCTS bit is not set and the uart_cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit will get set when the reset is removed if the uart_cts_n signal remains asserted. **Value** — **Description** 0x0 — no change on uart_cts_n since last read of MSR 0x1 — change on uart_cts_n since last read of MSR	RO	0x0

scr

Scratchpad Register

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC0201C
uart1	0xFFC03000	0xFFC0301C

Offset: 0x1C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								scr RW 0x0							

scr Fields

Bit	Name	Description	Access	Reset
7:0	scr	This register is for programmers to use as a temporary storage space.	RW	0x0

srbr

Used to accomadate burst accesses from the master.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02030
uart1	0xFFC03000	0xFFC03030

Offset: 0x30

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								srbr RW 0x0							

srbr Fields

Bit	Name	Description	Access	Reset
7:0	srbr	This is a shadow register for the RBR and has been allocated one 32-bit location so as to accommodate burst accesses from the master.This register contains the data byte received on the serial input port (sin). The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled, bit [0] of register FCR set to zero, the data in the RBR must be read before the next data arrives, otherwise it will be overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO will be preserved but any incoming data will be lost. An overrun error will also occur.	RW	0x0

sthr

Used to accomadate burst accesses from the master.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02034
uart1	0xFFC03000	0xFFC03034

Offset: 0x34

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="8">sthr
RW 0x0</td></tr>
</table>

sthr Fields

Bit	Name	Description	Access	Reset
7:0	`sthr`	This is a shadow register for the THR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains data to be transmitted on the serial output port (sout). Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If FIFO's are disabled bit [0] of register FCR set to zero and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFO's are enabled bit [0] of register FCR set to one and THRE is set, 128 characters of data may be written to the THR before the FIFO is full. The UART FIFO depth is configured for 128 characters. Any attempt to write data when the FIFO is full results in the write data being lost.	`RW`	`0x0`

far

This register is used in FIFO access testing.

Module Instance	Base Address	Register Address
`uart0`	`0xFFC02000`	`0xFFC02070`
`uart1`	`0xFFC03000`	`0xFFC03070`

Offset: `0x70`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`															`srbr_sthr` `RW 0x0`

far Fields

Bit	Name	Description	Access	Reset
0	`srbr_sthr`	This register is used to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFO's are enabled. When FIFO's are not enabled it allows the RBR to be written by the master and the THR to be read by the master Note: That when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFO's are treated as empty. **Value** — **Description** 0x0 — FIFO access mode disabled 0x1 — FIFO access mode enabled	`RW`	`0x0`

tfr

Used in FIFO Access test mode.

Module Instance	Base Address	Register Address
`uart0`	`0xFFC02000`	`0xFFC02074`
`uart1`	`0xFFC03000`	`0xFFC03074`

Offset: `0x74`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`Reserved`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`Reserved`								`tfr` `RO 0x0`							

tfr Fields

Bit	Name	Description	Access	Reset
7:0	`tfr`	These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO. When FIFO's are not enabled, reading this register gives the data in the THR.	`RO`	`0x0`

RFW

Used only with FIFO access test mode.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02078
uart1	0xFFC03000	0xFFC03078

Offset: 0x78

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						RFFE WO 0x0	rfpe WO 0x0	rfwd WO 0x0							

RFW Fields

Bit	Name	Description	Access	Reset
9	RFFE	These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are enabled, this bit is used to write framing error detection information to the receive FIFO. When FIFO's are not enabled, this bit is used to write framing error detection information to the RBR.	WO	0x0
8	rfpe	These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are enabled, this bit is used to write parity error detection information to the receive FIFO. When FIFO's are not enabled, this bit is used to write parity error detection information to the RBR.	WO	0x0
7:0	rfwd	These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFO's are enabled, the data that is written to the RFWD is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFO's are not enabled, the data that is written to the RFWD is pushed into the RBR.	WO	0x0

usr

Status of FIFO Operations.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC0207C

Module Instance	Base Address	Register Address
uart1	0xFFC03000	0xFFC0307C

Offset: 0x7C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rff RO 0x0	rfne RO 0x0	tfe RO 0x1	tfnf RO 0x1	Reserved

usr Fields

Bit	Name	Description	Access	Reset
4	rff	This Bit is used to indicate that the receive FIFO is completely full. This bit is cleared when the Rx FIFO is no longer full. **Value** **Description** 0x0 Receiive FIFO not full 0x1 Transmit FIFO is full	RO	0x0
3	rfne	This Bit is used to indicate that the receive FIFO contains one or more entries. This bit is cleared when the Rx FIFO is empty. **Value** **Description** 0x0 Receiive FIFO is empty 0x1 Receive FIFO is not empty	RO	0x0
2	tfe	This is used to indicate that the transmit FIFO is completely empty. This bit is cleared when the Tx FIFO is no longer empty. **Value** **Description** 0x0 Transmit FIFO is not empty 0x1 Transmit FIFO is empty	RO	0x1

Bit	Name	Description	Access	Reset
1	tfnf	This Bit is used to indicate that the transmit FIFO in not full. This bit is cleared when the Tx FIFO is full. **Value** **Description** 0x0 Transmit FIFO is full 0x1 Transmit FIFO is not full	RO	0x1

tfl

This register is used to specify the number of data entries in the Tx FIFO. Status Bits in USR register monitor the FIFO state.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02080
uart1	0xFFC03000	0xFFC03080

Offset: 0x80

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											tfl RO 0x0				

tfl Fields

Bit	Name	Description	Access	Reset
4:0	tfl	This indicates the number of data entries in the transmit FIFO.	RO	0x0

rfl

This register is used to specify the number of data entries in the Tx FIFO. Status Bits in USR register monitor the FIFO state.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02084
uart1	0xFFC03000	0xFFC03084

Offset: 0x84

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											rfl RO 0x0				

rfl Fields

Bit	Name	Description	Access	Reset
4:0	rfl	This indicates the number of data entries in the receive FIFO.	RO	0x0

srr

Provides Software Resets for Tx/Rx FIFO's and the uart.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02088
uart1	0xFFC03000	0xFFC03088

Offset: 0x88

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													xfr WO 0x0	rfr WO 0x0	ur WO 0x0

srr Fields

Bit	Name	Description	Access	Reset
2	xfr	This is a shadow register forthe Tx FIFO Reset bit (FCR[2]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the transmit FIFO.This resets the control portion of the transmit FIFO and treats the FIFO as empty. This will also de-assert the DMA Tx request and single signals. **Value** — **Description** 0x0 — No reset Tx FIFO 0x1 — Reset Tx FIFO	WO	0x0

Bit	Name	Description	Access	Reset
1	rfr	This is a shadow register for the Rx FIFO Reset bit (FCR[1]). This can be used to remove the burden on software having to store previously written FCR values (which are pretty static) just to reset the receive FIFO. This resets the control portion of the receive FIFO and treats the FIFO as empty. This will also de-assert the DMA Rx request and single signals. Note that this bit is 'self-clearing' and it is not necessary to clear this bit. **Value** — **Description** 0x0 — No reset Rx FIFO 0x1 — Reset Rx FIFO	WO	0x0
0	ur	This asynchronously resets the UART and synchronously removes the reset assertion. **Value** — **Description** 0x0 — No reset Uart 0x1 — Reset Uart	WO	0x0

srts

This is a shadow register for the RTS status (MCR[1]), this can be used to remove the burden of having to performing a read modify write on the MCR.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC0208C
uart1	0xFFC03000	0xFFC0308C

Offset: 0x8C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															srts RW 0x0

srts Fields

Bit	Name	Description	Access	Reset
0	srts	This is used to directly control the Request to Send (uart_rts_n) output. The Request to Send (uart_rts_n) output is used to inform the modem or data set that the UART is read to exchange data. The uart_rts_n signal is set low by programming MCR[1] (RTS) to a high. In Auto Flow Control, (MCR[5] set to one) and FIFO's are enabled (FCR[0] set to one), the uart_rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (uart_rts_n is inactive high when above the threshold). Note that in Loopback mode (MCR[4] set to one), the uart_rts_n output is held inactive high while the value of this location is internally looped back to an input. **Value** / **Description** 0x1 uart_rts_n logic0 0x0 uart_rts_n logic1	RW	0x0

sbcr

This is a shadow register for the Break bit [6] of the register LCR. This can be used to remove the burden of having to performing a read modify write on the LCR.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02090
uart1	0xFFC03000	0xFFC03090

Offset: 0x90

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sbcr RW 0x0

sbcr Fields

Bit	Name	Description	Access	Reset
0	sbcr	This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the uart_txd line is forced low until the Break bit is cleared. When in Loopback Mode, the break condition is internally looped back to the receiver. **Value** — **Description** 0x0 — no break 0x1 — break serial output spacing	RW	0x0

sdmam

This is a shadow register for the DMA mode bit (FCR[3]).

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC02094
uart1	0xFFC03000	0xFFC03094

Offset: `0x94`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															sdmam RW 0x0

sdmam Fields

Bit	Name	Description	Access	Reset
0	`sdmam`	This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the DMA Mode bit gets updated. **Value** **Description** 0x0 Single DMA Transfer Mode 0x1 Multiple DMA Transfer Mode	`RW`	`0x0`

sfe

This is a shadow register for the FIFO enable bit [0] of register FCR.

Module Instance	Base Address	Register Address
`uart0`	`0xFFC02000`	`0xFFC02098`
`uart1`	`0xFFC03000`	`0xFFC03098`

Offset: `0x98`

Access: `RW`

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="15">Reserved</td><td>sfe
RW 0x0</td></tr>
</table>

sfe Fields

Bit	Name	Description	Access	Reset
0	`sfe`	This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the FIFO enable bit gets updated. This enables/ disables the transmit (Tx) and receive (Rx) FIFO's. If this bit is set to zero (disabled) after being enabled then both the Tx and Rx controller portion of FIFO's will be reset. **Value** **Description** 0x0 Disable Rx/Tx 0x1 Enable Rx/Tx	`RW`	`0x0`

srt

This is a shadow register for the Rx trigger bits (FCR[7:6]).

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC0209C
uart1	0xFFC03000	0xFFC0309C

Offset: 0x9C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														srt RW 0x0	

srt Fields

Bit	Name	Description	Access	Reset
1:0	srt	This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the Rx trigger bit gets updated. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt will be generated. It also determines when the uart_dma_rx_req_n signal will be asserted when DMA Mode (FCR[3]) is set to one. The enum below shows trigger levels that are supported. **Value** / **Description** 0x0 one character in fifo 0x1 FIFO 1/4 full 0x2 FIFO 1/2 full 0x3 FIFO 2 less than full	RW	0x0

stet

This is a shadow register for the Tx empty trigger bits (FCR[5:4]).

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC020A0
uart1	0xFFC03000	0xFFC030A0

Offset: 0xA0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														stet RW 0x0	

stet Fields

Bit	Name	Description	Access	Reset
1:0	stet	This can be used to remove the burden of having to store the previously written value to the FCR in memory and having to mask this value so that only the Tx empty trigger bit gets updated. This is used to select the empty threshold level at which the THRE Interrupts will be generated when the mode is active. These threshold levels are also described in. The enum trigger levels are supported. **Value** **Description** 0x0 FIFO empty 0x1 Two characters in FIFO 0x2 FIFO quarter full 0x3 FIFO half full	RW	0x0

htx

Used to halt transmission for testing.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC020A4
uart1	0xFFC03000	0xFFC030A4

Offset: 0xA4

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															htx RW 0x0

htx Fields

Bit	Name	Description	Access	Reset
0	htx	This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFO's are enabled. Note, if FIFO's are not enabled, the setting of the halt Tx register will have no effect on operation. **Value** **Description** 0x0 Halt Tx disabled 0x1 Halt Tx enabled	RW	0x0

dmasa

DMA Operation Control

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC020A8
uart1	0xFFC03000	0xFFC030A8

Offset: 0xA8

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															dmasa WO 0x0

dmasa Fields

Bit	Name	Description	Access	Reset
0	dmasa	This register is used to perform DMA software acknowledge if a transfer needs to be terminated due to an error condition. For example, if the DMA disables the channel, then the uart should clear its request. This will cause the Tx request, Tx single, Rx request and Rx single signals to de-assert. Note that this bit is 'self-clearing' and it is not necessary to clear this bit.	WO	0x0

cpr

Describes various fixed hardware setups states.

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC020F4
uart1	0xFFC03000	0xFFC030F4

Offset: 0xF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								fifo_mode RO 0x37							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		dma_extra RO 0x1	uart_add_encoded_param RO 0x1	shadow RO 0x1	fifo_stat RO 0x1	fifo_access RO 0x1	additional_feat RO 0x1	sir_lp_mode RO 0x0	sir_mode RO 0x0	thre_mode RO 0x1	afce_mode RO 0x1	Reserved		apbdatawidth RO 0x2	

cpr Fields

Bit	Name	Description	Access	Reset
23:16	fifo_mode	Receiver and Transmitter FIFO depth in bytes. **Value** **Description** 0x80 FIFO Depth 128 bytes	RO	0x37
13	dma_extra	Configures the peripheral to have four additional DMA signals on the interface. **Value** **Description** 0x1 DMA Extra Supported	RO	0x1
12	uart_add_encoded_param	Configures the peripheral to have a configuration identification register. **Value** **Description** 0x1 ID register present	RO	0x1

Bit	Name	Description	Access	Reset
11	shadow	Configures the peripheral to have seven additional registers that shadow some of the existing register bits that are regularly modified by software. These can be used to reduce the software overhead that is introduced by having to perform read-modify writes. **Value** **Description** 0x1 Shadow Supported	RO	0x1
10	fifo_stat	Configures the peripheral to have three additional FIFO status registers. **Value** **Description** 0x1 FIFO Stat Supported	RO	0x1
9	fifo_access	Configures the peripheral to have a programmable FIFO access mode. This is used for test purposes, to allow the receiver FIFO to be written and the transmit FIFO to be read when FIFOs are implemented and enabled. **Value** **Description** 0x1 FIFO Access Supported	RO	0x1
8	additional_feat	Configures the uart to include fifo status register, shadow registers and encoded parameter register. **Value** **Description** 0x1 Additional Features Supported	RO	0x1
7	sir_lp_mode	LP Sir Mode not used in this application. **Value** **Description** 0x0 LP Sir Mode Not Supported	RO	0x0
6	sir_mode	Sir mode not used in this application. **Value** **Description** 0x0 Sir Mode Not Supported	RO	0x0

Bit	Name	Description	Access	Reset
5	`thre_mode`	Programmable Transmitter Hold Register Empty interrupt **Value** **Description** 0x1 Programmable Tx Hold Reg. Empty interrupt present	RO	0x1
4	`afce_mode`	Allows auto flow control. **Value** **Description** 0x1 Auto Flow	RO	0x1
1:0	`apbdatawidth`	Fixed to support an ABP data bus width of 32-bits. **Value** **Description** 0x2 APB Data Width = 32-bits	RO	0x2

ucv

Used only with Additional Features

Module Instance	Base Address	Register Address
uart0	0xFFC02000	0xFFC020F8
uart1	0xFFC03000	0xFFC030F8

Offset: `0xF8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
uart_component_version RO 0x3331312A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
uart_component_version RO 0x3331312A															

ucv Fields

Bit	Name	Description	Access	Reset
31:0	`uart_component_version`	ASCII value for each number in the version, followed by *For example 32_30_31_2A represents the version 2.01a	RO	0x3331312A

ctr

Describes a hex value associated with the component.

Module Instance	Base Address	Register Address
`uart0`	`0xFFC02000`	`0xFFC020FC`
`uart1`	`0xFFC03000`	`0xFFC030FC`

Offset: `0xFC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`peripheral_id` `RO 0x44570110`															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`peripheral_id` `RO 0x44570110`															

ctr Fields

Bit	Name	Description	Access	Reset
31:0	`peripheral_id`	This register contains the peripherals identification code.	`RO`	`0x44570110`

Document Revision History

Table 21-5: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• UART(RS232) Serial Protocol topic added • Interrupts section updated • Updated Interrupt type table • Added address map and register descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting updates.
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model, address map and register definitions, and reset sections.

Date	Version	Changes
January 2012	1.0	Initial release.

General-Purpose I/O Interface 22

2014.06.30

cv_54022 Subscribe Send Feedback

The hard processor system (HPS) provides three general-purpose I/O (GPIO) interface modules. The GPIO modules are instances of the Synopsys® DesignWare® APB General Purpose Programming I/O (DW_apb_gpio) peripheral.† [45]

Features of the GPIO Interface

The GPIO interface offers the following features:

- Supports digital de-bounce
- Configurable interrupt mode
- Supports up to 67 I/O pins and 14 input-only pins

GPIO Interface Block Diagram and System Integration

The figure below shows a block diagram of the GPIO interface. The following table shows a pin table of the GPIO interface

[45] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

Figure 22-1: Cyclone V SoC GPIO

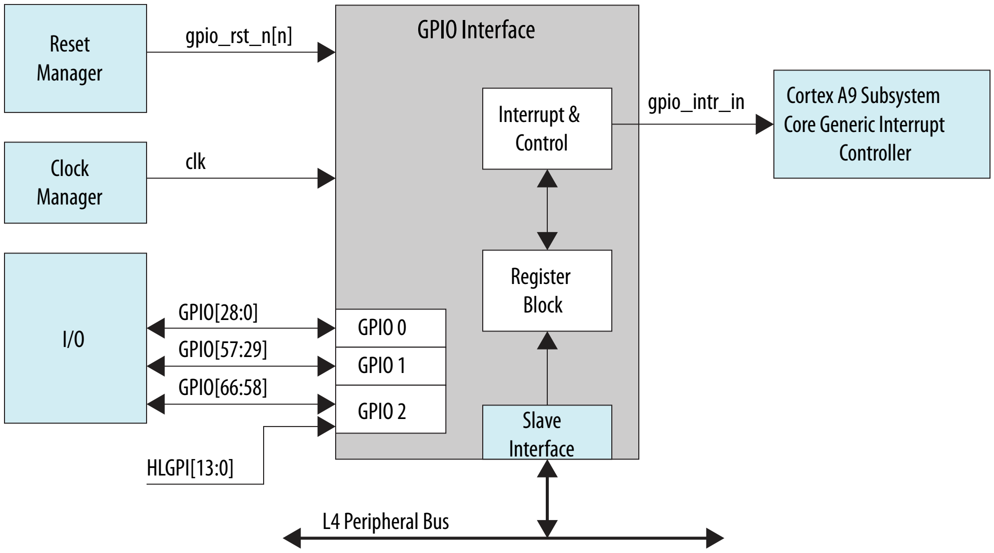


Table 22-1: GPIO Interface pin table

Pin Mux Name	Mapped to signal name	Comments
GPIO [28:0]	GPIO 0 [28:0]	Input / Output
GPIO [57:29]	GPIO 1 [28:0]	Input / Output
GPIO [66:58]	GPIO 2 [8:0]	Input / Output
HLGPI [13:0]	GPIO 2 [26:13]	Input only

Related Information

http://www.altera.com/literature/hb/cyclone-v/cv_52005.pdf
For more information on I/O banks locations on device, refer to *Cyclone V I/O Features*.

Functional Description of the GPIO Interface

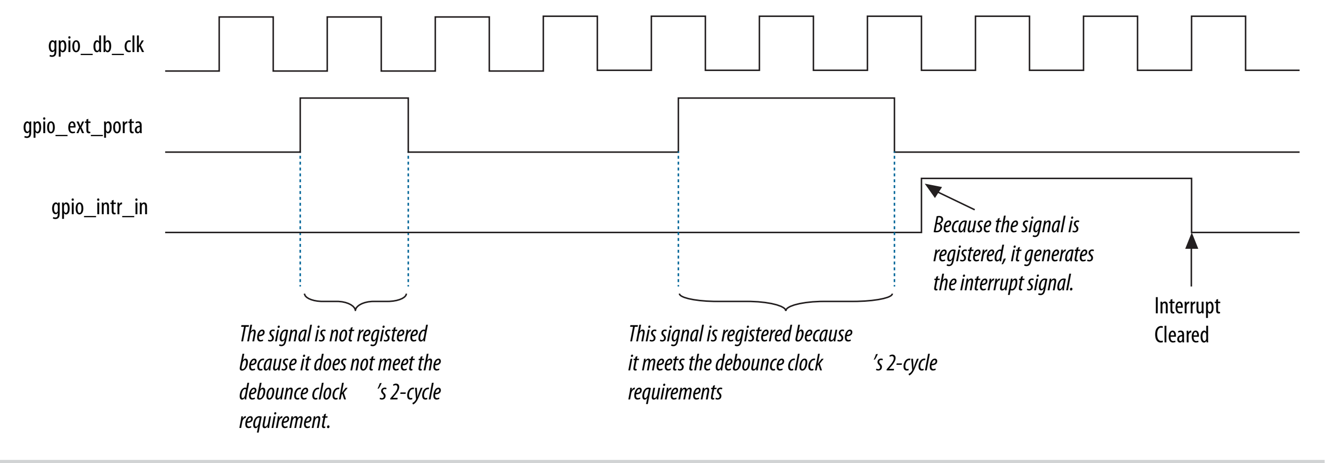
Debounce Operation

The GPIO modules provided in the HPS include optional debounce capabilities. The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock, `gpio_db_clk`. †

When input signals are debounced using the `gpio_db_clk` debounce clock, the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are filtered out. If the input signal pulse width is between one and two debounce clock widths it may or may not be filtered out, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered. †

The figure below shows a timing diagram of the debounce circuitry for both cases: a bounced input signal, and later, a propagated input signal.

Figure 22-2: Debounce Timing With Asychronous Reset Flip-Flops


gpio_db_clk
gpio_ext_porta
gpio_intr_in
Because the signal is registered, it generates the interrupt signal.
Interrupt Cleared
The signal is not registered because it does not meet the debounce clock 's 2-cycle requirement.
This signal is registered because it meets the debounce clock 's 2-cycle requirements

Note: Enabling the debounce circuitry increases interrupt latency by two clock cycles of the debounce clock.

Pin Directions

The pins `GPIO0` through `GPIO66` can be configured to be either input or output signals. The pins `HLGPI0` through `HLGPI13` share pins with the HPS DDR controller and are input-only signals.

GPIO Interface Programming Model

Debounce capability for each of the input signals can be enabled or disabled under software control by setting the corresponding bits in the `gpio_debounce` register, accordingly. The debounce clock must be stable and operational before the debounce capability is enabled.

Under software control, the direction of the external I/O pad is controlled by a write to the `gpio_swportx_ddr` register. When configured as input mode, reading `gpio_ext_porta` would read the values on the signal of the external I/O pad. When configured as output mode, the data written to the `gpio_swporta_dr` register drives the output buffer of the I/O pad. The same pins are shared for both input and output modes, so they cannot be configured as input and output modes at the same time. †

GPIO Interface Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- GPIO Module 0
- GPIO Module 1
- GPIO Module 2

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

GPIO Module Address Map

Registers in the GPIO module

Module Instance	Base Address
gpio0	0xFF708000
gpio1	0xFF709000
gpio2	0xFF70A000

GPIO Module

Register	Offset	Width	Acces s	Reset Value	Description
gpio_swporta_dr on page 22-5	0x0	32	RW	0x0	Port A Data Register
gpio_swporta_ddr on page 22-5	0x4	32	RW	0x0	Port A Data Direction Register
gpio_inten on page 22-6	0x30	32	RW	0x0	Interrupt Enable Register
gpio_intmask on page 22-7	0x34	32	RW	0x0	Interrupt Mask Register
gpio_inttype_level on page 22-8	0x38	32	RW	0x0	Interrupt Level Register
gpio_int_polarity on page 22-8	0x3C	32	RW	0x0	Interrupt Polarity Register
gpio_intstatus on page 22-9	0x40	32	RW	0x0	Interrupt Status Register
gpio_raw_intstatus on page 22-10	0x44	32	RW	0x0	Raw Interrupt Status Register
gpio_debounce on page 22-10	0x48	32	RW	0x0	Debounce Enable Register
gpio_porta_eoi on page 22-11	0x4C	32	WO	0x0	Clear Interrupt Register
gpio_ext_porta on page 22-12	0x50	32	RO	0x0	External Port A Register
gpio_ls_sync on page 22-13	0x60	32	RW	0x0	Synchronization Level Register
gpio_id_code on page 22-13	0x64	32	RO	0x0	ID Code Register
gpio_ver_id_code on page 22-14	0x6C	32	RO	0x3230382A	GPIO Version Register
gpio_config_reg2 on page 22-14	0x70	32	RO	0x39CFC	Configuration Register 2

Register	Offset	Width	Access	Reset Value	Description
gpio_config_reg1 on page 22-16	0x74	32	RO	0x1FF0F2	Configuration Register 1

gpio_swporta_dr

This GPIO Data register is used to input or output data Check the GPIO chapter in the handbook for details on how GPIO2 is implemented.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708000
gpio1	0xFF709000	0xFF709000
gpio2	0xFF70A000	0xFF70A000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_swporta_dr RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_swporta_dr RW 0x0															

gpio_swporta_dr Fields

Bit	Name	Description	Access	Reset
28:0	gpio_swporta_dr	Values written to this register are output on the I/O signals of the GPIO Data Register, if the corresponding data direction bits for GPIO Data Direction Field are set to Output mode. The value read back is equal to the last value written to this register. Check the GPIO chapter in the handbook for details on how GPIO2 is implemented.	RW	0x0

gpio_swporta_ddr

This register establishes the direction of each corresponding GPIO Data Field Bit. Check the GPIO chapter in the handbook for details on how GPIO2 is implemented.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708004
gpio1	0xFF709000	0xFF709004
gpio2	0xFF70A000	0xFF70A004

Offset: 0x4

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="13">gpio_swporta_ddr
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">gpio_swporta_ddr
RW 0x0</td></tr>
</table>

gpio_swporta_ddr Fields

Bit	Name	Description	Access	Reset
28:0	gpio_swporta_ddr	Values written to this register independently control the direction of the corresponding data bit in the Port A Data Register. Check the GPIO chapter in the handbook for details on how GPIO2 is implemented. **Value** — **Description** 0x0 — Input Direction 0x1 — Output Direction	RW	0x0

gpio_inten

The Interrupt enable register allows interrupts for each bit of the Port A data register.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708030
gpio1	0xFF709000	0xFF709030
gpio2	0xFF70A000	0xFF70A030

Offset: 0x30

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="3">Reserved</td><td colspan="13">gpio_inten
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">gpio_inten
RW 0x0</td></tr>
</table>

gpio_inten Fields

Bit	Name	Description	Access	Reset
28:0	gpio_inten	Allows each bit of Port A Data Register to be configured for interrupt capability. Interrupts are disabled on the corresponding bits of Port A Data Register if the corresponding data direction register is set to Output. **Value** — **Description** 0x0 — Disable Interrupt on Port A 0x1 — Enable Interrupt on Port A	RW	0x0

gpio_intmask

Controls which pins cause interrupts on Port A Data Register inputs.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708034
gpio1	0xFF709000	0xFF709034
gpio2	0xFF70A000	0xFF70A034

Offset: `0x34`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_intmask RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_intmask RW 0x0															

gpio_intmask Fields

Bit	Name	Description	Access	Reset
28:0	gpio_intmask	Controls whether an interrupt on Port A Data Register can generate an interrupt to the interrupt controller by not masking it. The unmasked status can be read as well as the resultant status after masking. **Value** — **Description** 0x0 — Interrupt bits are unmasked 0x1 — Mask Interrupt	RW	0x0

gpio_inttype_level

The interrupt level register defines the type of interrupt (edge or level).

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708038
gpio1	0xFF709000	0xFF709038
gpio2	0xFF70A000	0xFF70A038

Offset: 0x38

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_inttype_level RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_inttype_level RW 0x0															

gpio_inttype_level Fields

Bit	Name	Description	Access	Reset
28:0	gpio_inttype_level	This field controls the type of interrupt that can occur on the Port A Data Register. **Value** **Description** 0x0 Level-sensitive 0x1 Edge-sensitive	RW	0x0

gpio_int_polarity

Controls the Polarity of Interrupts that can occur on inputs of Port A Data Register

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF70803C
gpio1	0xFF709000	0xFF70903C
gpio2	0xFF70A000	0xFF70A03C

Offset: 0x3C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_int_polarity RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_int_polarity RW 0x0															

gpio_int_polarity Fields

Bit	Name	Description	Access	Reset
28:0	gpio_int_polarity	Controls the polarity of edge or level sensitivity that can occur on input of Port A Data Register. **Value** — **Description** 0x0 — Active low 0x1 — Active high	RW	0x0

gpio_intstatus

The Interrupt status is reported for all Port A Data Register Bits.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708040
gpio1	0xFF709000	0xFF709040
gpio2	0xFF70A000	0xFF70A040

Offset: 0x40

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_intstatus RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_intstatus RW 0x0															

gpio_intstatus Fields

Bit	Name	Description	Access	Reset
28:0	gpio_intstatus	Interrupt status of Port A Data Register. **Value** **Description** 0x0 Inactive 0x1 Active	RW	0x0

gpio_raw_intstatus

This is the Raw Interrupt Status Register for Port A Data Register. It is used with the Interrupt Mask Register to allow interrupts from the Port A Data Register.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708044
gpio1	0xFF709000	0xFF709044
gpio2	0xFF70A000	0xFF70A044

Offset: 0x44

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_raw_intstatus RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_raw_intstatus RW 0x0															

gpio_raw_intstatus Fields

Bit	Name	Description	Access	Reset
28:0	gpio_raw_intstatus	Raw interrupt of status of Port A Data Register (premasking bits) **Value** **Description** 0x0 Inactive 0x1 Active	RW	0x0

gpio_debounce

Debounces each IO Pin

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708048

Module Instance	Base Address	Register Address
gpio1	0xFF709000	0xFF709048
gpio2	0xFF70A000	0xFF70A048

Offset: 0x48

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_debounce RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_debounce RW 0x0															

gpio_debounce Fields

Bit	Name	Description	Access	Reset
28:0	gpio_debounce	Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. A signal must be valid for two periods of an external clock (gpio_db_clk) before it is internally processed. **Value** **Description** 0x0 No debounce 0x1 Enable debounce	RW	0x0

gpio_porta_eoi

Port A Data Register interrupt handling.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF70804C
gpio1	0xFF709000	0xFF70904C
gpio2	0xFF70A000	0xFF70A04C

Offset: 0x4C

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_porta_eoi WO 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_porta_eoi WO 0x0															

gpio_porta_eoi Fields

Bit	Name	Description	Access	Reset
28:0	gpio_porta_eoi	Controls the clearing of edge type interrupts from the Port A Data Register. **Value** **Description** 0x0 No interrupt clear 0x1 Clear interrupt	WO	0x0

gpio_ext_porta

The external port register is used to input data to the metastability flops.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708050
gpio1	0xFF709000	0xFF709050
gpio2	0xFF70A000	0xFF70A050

Offset: 0x50

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			gpio_ext_porta RO 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_ext_porta RO 0x0															

gpio_ext_porta Fields

Bit	Name	Description	Access	Reset
28:0	gpio_ext_porta	When Port A Data Register is configured as Input, then reading this location reads the values on the signals. When the data direction of Port A Data Register is set as Output, reading this location reads Port A Data Register	RO	0x0

gpio_ls_sync

The Synchronization level register is used to synchronize input with l4_mp_clk

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708060
gpio1	0xFF709000	0xFF709060
gpio2	0xFF70A000	0xFF70A060

Offset: 0x60

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															gpio_ls_sync RW 0x0

gpio_ls_sync Fields

Bit	Name	Description	Access	Reset
0	gpio_ls_sync	The level-sensitive interrupts is synchronized to l4_mp_clk. **Value** **Description** 0x0 No synchronization to l4_mp_clk 0x1 Synchronize to l4_mp_clk	RW	0x0

gpio_id_code

GPIO ID code.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708064
gpio1	0xFF709000	0xFF709064
gpio2	0xFF70A000	0xFF70A064

Offset: 0x64

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio_id_code RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_id_code RO 0x0															

gpio_id_code Fields

Bit	Name	Description	Access	Reset
31:0	gpio_id_code	Chip identification	RO	0x0

gpio_ver_id_code

GPIO Component Version

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF70806C
gpio1	0xFF709000	0xFF70906C
gpio2	0xFF70A000	0xFF70A06C

Offset: 0x6C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
gpio_ver_id_code RO 0x3230382A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_ver_id_code RO 0x3230382A															

gpio_ver_id_code Fields

Bit	Name	Description	Access	Reset
31:0	gpio_ver_id_code	ASCII value for each number in the version, followed by *. For example. 32_30_31_2A represents the version 2.01	RO	0x3230382A

gpio_config_reg2

Specifies the bit width of port A.

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708070
gpio1	0xFF709000	0xFF709070

Module Instance	Base Address	Register Address
gpio2	0xFF70A000	0xFF70A070

Offset: 0x70

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												encoded_id_pwidth_d RO 0x7			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
encoded_id_pwidth_d RO 0x7	encoded_id_pwidth_c RO 0x7					encoded_id_pwidth_b RO 0x7					encoded_id_pwidth_a RO 0x1C				

gpio_config_reg2 Fields

Bit	Name	Description	Access	Reset
19:15	encoded_id_pwidth_d	Specifies the width of GPIO Port D. Ignored because there is no Port D in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0x1c — Width (less 1) of 29 bits	RO	0x7
14:10	encoded_id_pwidth_c	Specifies the width of GPIO Port C. Ignored because there is no Port C in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0x1c — Width (less 1) of 29 bits	RO	0x7
9:5	encoded_id_pwidth_b	Specifies the width of GPIO Port B. Ignored because there is no Port B in the GPIO. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0x1c — Width (less 1) of 29 bits	RO	0x7

Bit	Name	Description	Access	Reset
4:0	encoded_id_pwidth_a	Specifies the width of GPIO Port A. The value 28 represents the 29-bit width less one. **Value** — **Description** 0x7 — Width (less 1) of 8 bits 0x1c — Width (less 1) of 29 bits	RO	0x1C

gpio_config_reg1

Reports settings of various GPIO configuration parameters

Module Instance	Base Address	Register Address
gpio0	0xFF708000	0xFF708074
gpio1	0xFF709000	0xFF709074
gpio2	0xFF70A000	0xFF70A074

Offset: 0x74

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved											encoded_id_width RO 0x1F				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_id RO 0x1	add_encoded_params RO 0x1	debounce RO 0x1	porta_intr RO 0x1	Reserved			hw_porta RO 0x0	portd_single_ctl RO 0x1	portc_single_ctl RO 0x1	portb_single_ctl RO 0x1	porta_single_ctl RO 0x1	num_ports RO 0x0		apb_data_width RO 0x2	

gpio_config_reg1 Fields

Bit	Name	Description	Access	Reset
20:16	encoded_id_width	This value is fixed at 32 bits. **Value** — **Description** 0x1f — Width of ID Field	RO	0x1F

Bit	Name	Description	Access	Reset
15	gpio_id	Provides an ID code value **Value** **Description** 0x1 GPIO ID Code	RO	0x1
14	add_encoded_params	Fixed to allow the indentification of the Designware IP component. **Value** **Description** 0x1 Enable IP indentification	RO	0x1
13	debounce	The value of this field is fixed to allow debouncing of the Port A signals. **Value** **Description** 0x1 Debounce is Enabled	RO	0x1
12	porta_intr	The value of this field is fixed to allow interrupts on Port A. **Value** **Description** 0x1 Port A Interrupts Enabled	RO	0x1
8	hw_porta	The value is fixed to enable Port A configuration to be controlled by software only. **Value** **Description** 0x0 Software Configuration Control Enabled	RO	0x0
7	portd_single_ctl	Indicates the mode of operation of Port D to be software controlled only. Ignored because there is no Port D in the GPIO. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1
6	portc_single_ctl	Indicates the mode of operation of Port C to be software controlled only. Ignored because there is no Port C in the GPIO. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1

Bit	Name	Description	Access	Reset
5	`portb_single_ctl`	Indicates the mode of operation of Port B to be software controlled only. Ignored because there is no Port B in the GPIO. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1
4	`porta_single_ctl`	Indicates the mode of operation of Port A to be software controlled only. **Value** **Description** 0x1 Software Enabled Individual Port Control	RO	0x1
3:2	`num_ports`	The value of this register is fixed at one port (Port A). **Value** **Description** 0x0 Number of GPIO Ports = 1	RO	0x0
1:0	`apb_data_width`	Fixed to support an ABP data bus width of 32-bits. **Value** **Description** 0x2 APB Data Width = 32-bits	RO	0x2

Document Revision History

Table 22-2: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added Address Map and Register Descriptions
February 2014	2014.02.28	Updated content in sections: • Features of the GPIO Interface • GPIO Interface Block Diagram and System Integration • Debounce Operation

Date	Version	Changes
December 2013	2013.12.30	Minor formatting updates Updated GPIO interface block diagram and GPIO interface pin table
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model section.
January 2012	1.0	Initial release.

Timer 23

2014.06.30

cv_54023 Subscribe Send Feedback

The hard processor system (HPS) provides four 32-bit general-purpose timers connected to the level 4 (L4) peripheral bus.The timers optionally generate an interrupt when the 32-bit binary count-down timer reaches zero. The timers are instances of the Synopsys® DesignWare® APB Timers (DW_apb_timers) peripheral.†[46]

Related Information

Cortex-A9 MPCore on page 9-4
The MPU subsystem provides additional timers. For more information about the timers in the MPU, refer to the *Cortex-A9 Microprocessor Unit Subsystem* chapter.

Features of the Timer

- Supports interrupt generation
- Supports free-running mode
- Supports user-defined count mode

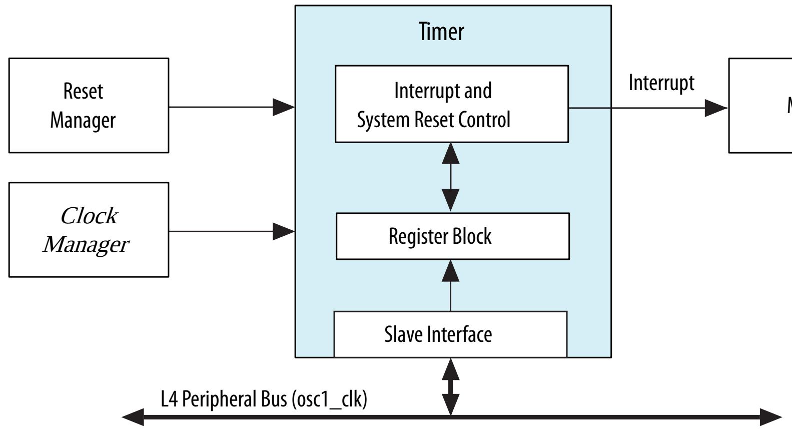
Timer Block Diagram and System Integration

Each timer includes a slave interface for control and status register (CSR) access, a register block, and a programmable 32-bit down counter that generates interrupts on reaching zero. The timer operates on a single clock domain driven by the clock manager.

[46] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

Figure 23-1: Timer Block Diagram



Functional Description of the Timer

The 32-bit timer counts down from a programmed value and generates an interrupt when the count reaches zero. The timer has an independent clock input connected to the system clock signal or to an external clock source. †

The timer supports the following modes of operation:

- Free-running mode—decrementing from the maximum value (0xFFFFFFFF). Reloads maximum value upon reaching zero.
- User-defined count mode—generates a periodic interrupt. Decrements from the user-defined count value loaded from the timer1 load count register (`timer1loadcount`). Reloads the user-defined count upon reaching zero.

The initial value for the timer (that is, the value from which it counts down) is loaded into the timer by the `timer1loadcount` register. The following events can cause a timer to load the initial count from the `timer1loadcount` register: †

- Timer is enabled after being reset or disabled
- Timer counts down to 0

Clocks

Table 23-1: Timer Clock Characteristics

Timer	System Clock	Notes
OSC1 timer 0	`osc1_clk`	—
OSC1 timer 1		
SP timer 0	`l4_sp_clk`	Timer must be disabled if clock frequency changes
SP timer 1		

The timers above are labeled according to the clock it receives. OSC timers receive the oscillator clock `osc1_clk` and the SP timers receives the l4 slave peripheral clock `l4_sp_clk` .

SP timer 0 and SP timer 1 must be disabled before `l4_sp_clk` is changed to another frequency. You can then re-enable the timer once the clock frequency change takes effect. You cannot change the frequency of OSC1 timer 0 and OSC1 timer 1.

Related Information

Clock Manager on page 2-1
For more information about clock performance, refer to the *Clock Manager* chapter.

Resets

The timers are reset by a cold or warm reset. Resetting the timers produces the following results in the following order:

1. The timer is disabled.
2. The interrupt is enabled.
3. The timer enters free-running mode.
4. The timer count load register value is set to zero.

Interrupts

The timer1 interrupt status (`timer1intstat`) and timer1 end of interrupt (`timer1eoi`) registers handle the interrupts. The `timer1intstat` register allows you to read the status of the interrupt. Reading from the `timer1eoi` register clears the interrupt. †

The timer1 control register (`timer1controlreg`) contains the timer1 interrupt mask bit (`timer1_interrupt_mask`) to mask the interrupt. In both the free-running and user-defined count modes of operation, the timer generates an interrupt signal when the timer count reaches zero and the interrupt mask bit of the control register is high.

If the timer interrupt is set, then it is cleared when the timer is disabled.

FPGA Interface

The timer interrupts can be routed to the FPGA interface. You can configure and route the interrupts when you instantiate the HPS component in Qsys.

Related Information

Interrupts on page 27-6
For more information about configuring and routing timer interrupts, refer to the interrupt section of the*Instantiating the HPS Component* chapter in the *Hard Processor System Technical Reference Manual.*

Timer Programming Model

Initialization

To initialize the timer, perform the following steps: †

1. Initialize the timer through the `timer1controlreg` register: †
    - Disable the timer by writing a 0 to the timer1 enable bit (`timer1_enable`) of the `timer1controlreg` register. †

Note: Before writing to a timer1 load count register (`timer1loadcount`), you must disable the timer by writing a 0 to the `timer1_enable` bit of the `timer1controlreg` register to avoid potential synchronization problems. †

- Program the timer mode—user-defined count or free-running—by writing a 0 or 1, respectively, to the timer1 mode bit (`timer1_mode`) of the `timer1controlreg` register. †
- Set the interrupt mask as either masked or not masked by writing a 1 or 0, respectively, to the `timer1_interrupt_mask` bit of the `timer1controlreg` register. †

2. Load the timer counter value into the `timer1loadcount` register. †
3. Enable the timer by writing a 1 to the `timer1_enable` bit of the `timer1controlreg` register. †

Enabling the Timer

When a timer transitions to the enabled state, the current value of `timer1loadcount` register is loaded into the timer counter. †

1. To enable the timer, write a 1 to the `timer1_enable` bit of the `timer1controlreg` register.

Disabling the Timer

When the timer enable bit is cleared to 0, the timer counter and any associated registers in the timer clock domain, are asynchronously reset. †

1. To disable the timer, write a 0 to the `timer1_enable` bit. †

Loading the Timer Countdown Value

When a timer counter is enabled after being reset or disabled, the count value is loaded from the `timer1loadcount` register; this occurs in both free-running and user-defined count modes. †

When a timer counts down to 0, it loads one of two values, depending on the timer operating mode: †

- User-defined count mode—timer loads the current value of the `timer1loadcount` register. Use this mode if you want a fixed, timed interrupt. Designate this mode by writing a 1 to the `timer1_mode` bit of the `timer1controlreg` register. †
- Free-running mode—timer loads the maximum value (0xFFFFFFFF). The timer max count value allows for a maximum amount of time to reprogram or disable the timer before another interrupt occurs. Use this mode if you want a single timed interrupt. Enable this mode by writing a 0 to the `timer1_mode` bit of the `timer1controlreg` register. †

Servicing Interrupts

Clearing the Interrupt

An active timer interrupt can be cleared in two ways.

1. If you clear the interrupt at the same time as the timer reaches 0, the interrupt remains asserted. This action happens because setting the timer interrupt takes precedence over clearing the interrupt. †
2. To clear an active timer interrupt, read the `timer1eoi` register or disable the timer. When the timer is enabled, its interrupt remains asserted until it is cleared by reading the `timer1eoi` register. †

Checking the Interrupt Status

You can query the interrupt status of the timer without clearing its interrupt.

1. To check the interrupt status, read the `timer1intstat` register. †

Masking the Interrupt

The timer interrupt can be masked using the `timer1controlreg` register.

To mask an interrupt, write a 1 to the `timer1_interrupt_mask` bit of the `timer1controlreg` register. †

Timer Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridges consist of the following regions:

- OSC1 Timer Module 0
- OSC1 Timer Module 1
- SP Timer Module 0
- SP Timer Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

Timer Module Address Map

Registers in the timer module. The timer IP core supports multiple timers but it is configured for just one timer. The term Timer1 refers to this one timer in the IP core and not the module instance.

Module Instance	Base Address
sptimer0	0xFFC08000
sptimer1	0xFFC09000
osc1timer0	0xFFD00000
osc1timer1	0xFFD01000

Timer Module

Register	Offset	Width	Access	Reset Value	Description
timer1loadcount on page 23-6	0x0	32	RW	0x0	Timer1 Load Count Register
timer1currentval on page 23-6	0x4	32	RO	0x0	Timer1 Current Value Register
timer1controlreg on page 23-7	0x8	32	RW	0x0	Timer1 Control Register

Register	Offset	Width	Access	Reset Value	Description
timer1eoi on page 23-8	0xC	32	RO	0x0	Timer1 End-of-Interrupt Register
timer1intstat on page 23-9	0x10	32	RO	0x0	Timer1 Interrupt Status Register
timersintstat on page 23-10	0xA0	32	RO	0x0	Timers Interrupt Status Register
timerseoi on page 23-11	0xA4	32	RO	0x0	Timers End-of-Interrupt Register
timersrawintstat on page 23-12	0xA8	32	RO	0x0	Timers Raw Interrupt Status Register
timerscompversion on page 23-13	0xAC	32	RO	0x3230352A	Timers Component Version Register

timer1loadcount

Used to load counter value into Timer1

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC08000
sptimer1	0xFFC09000	0xFFC09000
osc1timer0	0xFFD00000	0xFFD00000
osc1timer1	0xFFD01000	0xFFD01000

Offset: 0x0

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">timer1loadcount
RW 0x0</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">timer1loadcount
RW 0x0</td></tr>
</table>

timer1loadcount Fields

Bit	Name	Description	Access	Reset
31:0	timer1loadcount	Value to be loaded into Timer1. This is the value from which counting commences. Any value written to this register is loaded into the associated timer.	RW	0x0

timer1currentval

Provides current value of Timer1

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC08004
sptimer1	0xFFC09000	0xFFC09004
osc1timer0	0xFFD00000	0xFFD00004
osc1timer1	0xFFD01000	0xFFD01004

Offset: 0x4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timer1currentval RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
timer1currentval RO 0x0															

timer1currentval Fields

Bit	Name	Description	Access	Reset
31:0	timer1currentval	Current value of Timer1.	RO	0x0

timer1controlreg

This register controls enabling, operating mode (free-running or user-defined-count), and interrupt mask of Timer1. You can program this register to enable or disable Timer1 and to control its mode of operation.

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC08008
sptimer1	0xFFC09000	0xFFC09008
osc1timer0	0xFFD00000	0xFFD00008
osc1timer1	0xFFD01000	0xFFD01008

Offset: 0x8

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													timer1_interrupt_mask RW 0x0	timer1_mode RW 0x0	timer1_enable RW 0x0

timer1controlreg Fields

Bit	Name	Description	Access	Reset
2	timer1_interrupt_mask	Timer1 interrupt mask **Value** **Description** 0x0 interrupt not masked (enabled) 0x1 interrupt masked (disabled)	RW	0x0
1	timer1_mode	Sets operating mode. NOTE: You must set the timer1loadcount register to all ones before enabling the timer in free-running mode. **Value** **Description** 0x0 Free-running mode 0x1 User-defined count mode	RW	0x0
0	timer1_enable	Timer1 enable/disable bit. **Value** **Description** 0x0 Timer1 Disabled 0x1 Timer1 Enabled	RW	0x0

timer1eoi

Clears Timer1 interrupt when read.

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC0800C
sptimer1	0xFFC09000	0xFFC0900C

Module Instance	Base Address	Register Address
osc1timer0	0xFFD00000	0xFFD0000C
osc1timer1	0xFFD01000	0xFFD0100C

Offset: `0xC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															timer1eoi RO 0x0

timer1eoi Fields

Bit	Name	Description	Access	Reset
0	timer1eoi	Reading from this register clears the interrupt from Timer1 and returns 0.	RO	0x0

timer1intstat

Provides the interrupt status of Timer1 after masking.

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC08010
sptimer1	0xFFC09000	0xFFC09010
osc1timer0	0xFFD00000	0xFFD00010
osc1timer1	0xFFD01000	0xFFD01010

Offset: `0x10`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															timer1intstat RO 0x0

timer1intstat Fields

Bit	Name	Description	Access	Reset
0	`timer1intstat`	Provides the interrupt status for Timer1. The status reported is after the interrupt mask has been applied. Reading from this register does not clear any active interrupts. **Value** **Description** 0x0 Timer1 interrupt is not active 0x1 Timer1 interrupt is active	RO	0x0

timersintstat

Provides the interrupt status for all timers after masking. Because there is only Timer1 in this module instance, this status is the same as timer1intstat.

Module Instance	Base Address	Register Address
`sptimer0`	`0xFFC08000`	`0xFFC080A0`
`sptimer1`	`0xFFC09000`	`0xFFC090A0`
`osc1timer0`	`0xFFD00000`	`0xFFD000A0`
`osc1timer1`	`0xFFD01000`	`0xFFD010A0`

Offset: `0xA0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															timersintstat RO 0x0

timersintstat Fields

Bit	Name	Description	Access	Reset
0	timersintstat	Provides the interrupt status for Timer1. Because there is only Timer1 in this module instance, this status is the same as timer1intstat. The status reported is after the interrupt mask has been applied. Reading from this register does not clear any active interrupts. **Value** / **Description** 0x0 timer_intr is not active 0x1 timer_intr is active	RO	0x0

timerseoi

Clears Timer1 interrupt when read. Because there is only Timer1 in this module instance, reading this register has the same effect as reading timer1eoi.

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC080A4
sptimer1	0xFFC09000	0xFFC090A4
osc1timer0	0xFFD00000	0xFFD000A4
osc1timer1	0xFFD01000	0xFFD010A4

Offset: 0xA4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															timerseoi RO 0x0

timerseoi Fields

Bit	Name	Description	Access	Reset
0	timerseoi	Reading from this register clears the interrupt all timers and returns 0. Because there is only Timer1 in this module instance, reading this register has the same effect as reading timer1eoi.	RO	0x0

timersrawintstat

Provides the interrupt status for all timers before masking. Note that there is only Timer1 in this module instance.

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC080A8
sptimer1	0xFFC09000	0xFFC090A8
osc1timer0	0xFFD00000	0xFFD000A8
osc1timer1	0xFFD01000	0xFFD010A8

Offset: 0xA8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															timersrawintstat RO 0x0

timersrawintstat Fields

Bit	Name	Description	Access	Reset
0	`timersrawintstat`	Provides the interrupt status for Timer1. Because there is only Timer1 in this module instance, this status is the same as timer1intstat. The status reported is before the interrupt mask has been applied. Reading from this register does not clear any active interrupts. **Value** — **Description** 0x0 — Timer1 interrupt is not active 0x1 — Timer1 interrupt is active	RO	0x0

timerscompversion

Module Instance	Base Address	Register Address
sptimer0	0xFFC08000	0xFFC080AC
sptimer1	0xFFC09000	0xFFC090AC
osc1timer0	0xFFD00000	0xFFD000AC
osc1timer1	0xFFD01000	0xFFD010AC

Offset: `0xAC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
timerscompversion RO 0x3230352A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
timerscompversion RO 0x3230352A															

timerscompversion Fields

Bit	Name	Description	Access	Reset
31:0	`timerscompversion`	Current revision number of the timers component.	RO	0x3230352A

Document Revision History

Table 23-2: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• "FPGA Interface" section added • Added address map and register descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting upates
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model and address map and register definitions sections.
January 2012	1.0	Initial release.

Watchdog Timer 24

2014.06.30

cv_54024 Subscribe Send Feedback

The watchdog timers are peripherals you can use to recover from system lockup that might be caused by software or system related issues. The hard processor system (HPS) provides two programmable watchdog timers, which are connected to the level 4 (L4) peripheral bus. The watchdog timers are instances of the Synopsys® DesignWare® APB Watchdog Timer (DW_apb_wdt) peripheral. [47]

The microprocessor unit (MPU) subsystem provides two additional watchdog timers.

Related Information

Cortex-A9 MPCore on page 9-4
For more information about the watchdog timers in the MPU, refer to *Cortex A9 Microprocessor Unit Subsystem* chapter.

Features of the Watchdog Timer

The following list describes the features of the watchdog timer:

[47] Portions © 2014 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.

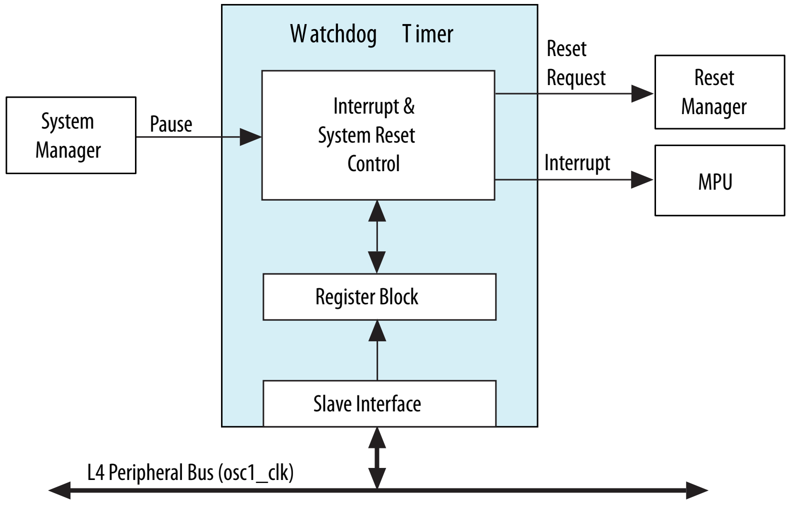
- Programmable 32-bit timeout range
- Timer counts down from a preset value to zero, then performs one of the following user-configurable operations:
  - Generates a system reset †
  - Generates an interrupt, restarts the timer, and if the timer is not cleared before a second timeout occurs, generates a system reset
- Dual programmable timeout period, used when the time to wait after the first start is different than that required for subsequent restarts †
- Prevention of accidental restart of the watchdog counter †
- Prevention of accidental disabling of the watchdog counter †
- Pause mode for debugging

Watchdog Timer Block Diagram and System Integration

Each watchdog timer consists of a slave interface for control and status register (CSR) access, a register block, and a 32-bit down counter that operates on the slave interface clock (`osc1_clk`). A pause input, driven by the system manager, optionally pauses the counter when a CPU is being debugged.

The watchdog timer drives an interrupt request to the MPU and a reset request to the reset manager.

Figure 24-1: Watchdog Timer Block Diagram



Related Information

- **Reset Manager** on page 3-1
  For more information, refer to the *Reset Manager* chapter.
- **Cortex-A9 MPCore** on page 9-4
  For more information about the watchdog timers in the MPU, refer to *Cortex A9 Microprocessor Unit Subsystem* chapter.

Functional Description of the Watchdog Timer

Watchdog Timer Counter

Each watchdog timer is a programmable, little-endian down counter that decrements by one on each clock cycle. The watchdog timer supports 16 fixed timeout period values, Software chooses which timeout periods are desired. A timeout period is 2<*n*> `osc1_clk` clock periods, where *n* is an integer from 16 to 31 inclusive.

Software must regularly restart the timer (which reloads the counter with the restart timeout period value) to indicate that the system is functioning normally. Software can reload the counter at any time by writing to the restart register. If the counter reaches zero, the watchdog timer has timed out, indicating an unrecoverable error has occurred and a system reset is needed.

Software configures the watchdog timer to one of the following output response modes:

- On timeout, generate a reset request.
- On timeout, assert an interrupt request and restart the watchdog timer. Software must service the interrupt and reset the watchdog timer before a second timeout occurs. Otherwise, generate a reset request.

If a restart occurs at the same time the watchdog counter reaches zero, an interrupt is not generated.

Related Information

- **Watchdog Timer Clocks** on page 24-3
- **Setting the Timeout Period Values** on page 24-4
- **Selecting the Output Response Mode** on page 24-4
- **Reloading a Watchdog Counter** on page 24-5

Watchdog Timer Pause Mode

The watchdog timers can be paused during debugging. The watchdog timer pause mode is controlled by the system manager. The following options are available:

- Pause the timer while either CPU0 or CPU1 is in debug mode
- Pause the timer while only CPU1 is in debug mode
- Pause the timer while only CPU0 is in debug mode
- Do not pause the timer

When pause mode is enabled, the system manager pauses the watchdog timer while debugging. When pause mode is disabled, the watchdog timer runs while debugging.

At reset, the watchdog pausing feature is enabled for both CPUs by default.

Related Information

Pausing a Watchdog Timer on page 24-5

Watchdog Timer Clocks

Each watchdog timer is connected to the `osc1_clk` clock so that timer operation is not dependent on the phase-locked loops (PLLs) in the clock manager. This independence allows recovery from software that inadvertently programs the PLLs in the clock manager incorrectly.

Related Information

Clock Manager on page 2-1
For more information, refer to the *Clock Manager* chapter.

Watchdog Timer Resets

Watchdog timers are reset by a cold or warm reset from the reset manager, and are disabled when exiting reset. †

Related Information

Reset Manager on page 3-1
For more information, refer to the *Reset Manager* chapter.

FPGA Interface

The watchdog timer interrupts can be routed to the FPGA interface. You can configure and route the interrupts when you instantiate the HPS component in Qsys

Related Information

Interrupts on page 27-6
For more information, refer to the *Instantiating the HPS Component* chapter in the *Hard Processor System Technical Reference Manual.*

Watchdog Timer Programming Model

Setting the Timeout Period Values

The watchdog timers have a dual timeout period. The counter uses the initial start timeout period value the first the timer is started. All subsequent restarts use the restart timeout period. The valid values are $2^{(16+<i>)}$ – 1 clock cycles, where i is an integer from 0 to 15. To set the programmable timeout periods, perform the following actions in no specific order:

Note: Set the timeout values before enabling the timer.

- To set the initial start timeout period, write i to the timeout period for the initialization field (`top_init`) of the watchdog timeout range register (`wdt_torr`).
- To set the restart timeout period, write i to the timeout period field (`top`) of the `wdt_torr` register

Selecting the Output Response Mode

The watchdog timers have two output response modes. To select the desired mode, perform one of the following actions:

- To generate a system reset request when a timeout occurs, write 0 to the output response mode bit (`rmod`) of the watchdog timer control register (`wdt_cr`).
- To generate an interrupt and restart the timer when a timeout occurs, write 1 to the `rmod` field of the `wdt_cr` register.

If a restart occurs at the same time the watchdog counter reaches zero, a system reset is not generated. †

Related Information

Watchdog Timer Counter on page 24-3

Enabling and Initially Starting a Watchdog Timer

To enable and start a watchdog timer, write the value 1 to the watchdog timer enable bit (`wdt_en`) of the `wdt_cr` register.

Reloading a Watchdog Counter

To reload a watchdog counter, write the value 0x76 to the counter restart register (`wdt_crr`). This unique 8-bit value is used as a safety feature to prevent accidental restarts.

Pausing a Watchdog Timer

Pausing the watchdog timers is controlled by the L4 watchdog debug register (`wddbg`) in the system manager.

Related Information

Features of the System Manager on page 5-1
For more information, refer to the *System Manager* chapter.

Disabling and Stopping a Watchdog Timer

The watchdog timers are disabled and stopped by resetting them from the reset manager.

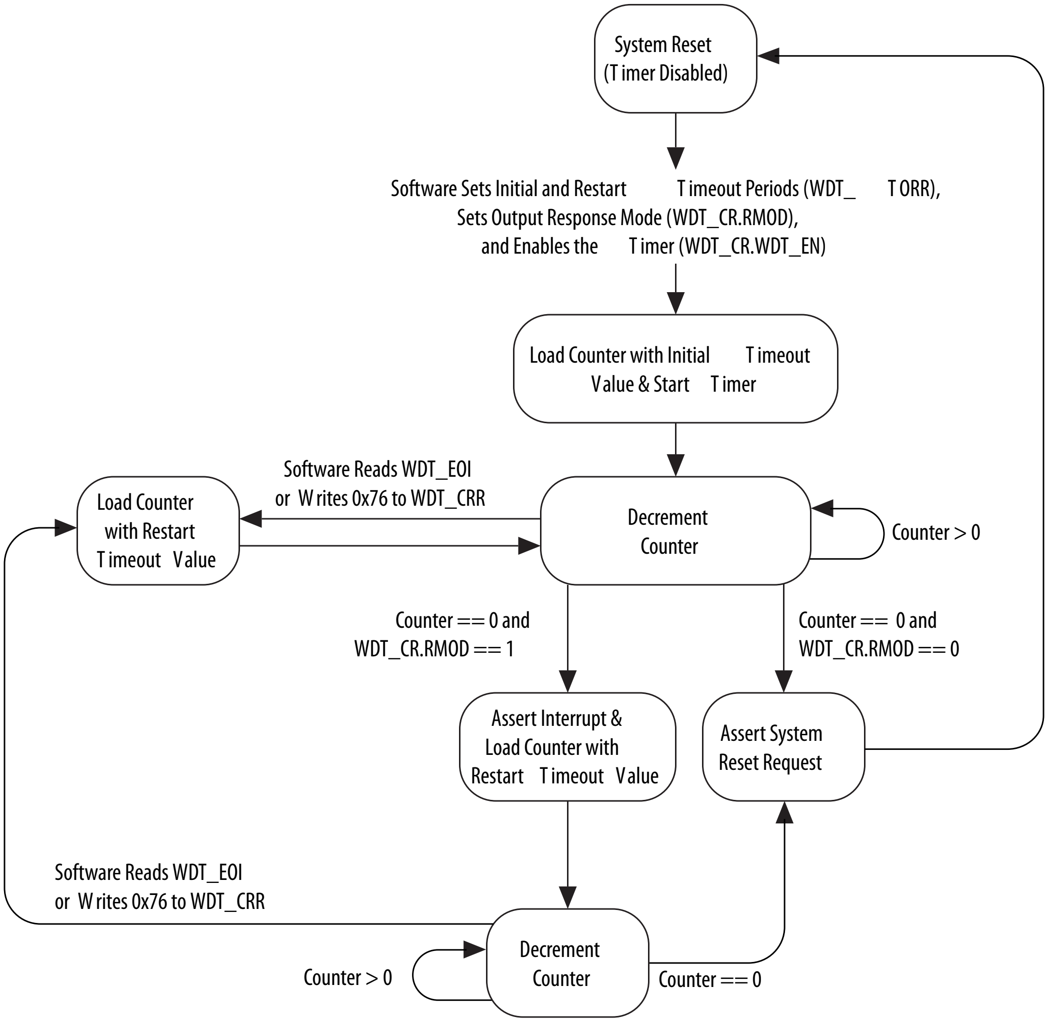
Related Information

Reset Manager on page 3-1
For more information, refer to the *Reset Manager* chapter.

Watchdog Timer State Machine

The following figure illustrates the behavior of the watchdog timer, including the behavior of both output response modes. Once initialized, the counter decrements at every clock cycle. The state machine remains in the Decrement Counter state until the counter reaches zero, or the watchdog timer is restarted. If software reads the interrupt clear register (`wdt_eoi`), or writes 0x76 to the `wdt_crr` register, the state changes from Decrement Counter to Load Counter with Restart Timeout Value. In this state, the watchdog counter gets reloaded with the restart timeout value, and then the state changes back to Decrement Counter.

Figure 24-2: Watchdog Timer State Machine


System Reset (Timer Disabled)
Software Sets Initial and Restart Timeout Periods (WDT_TORR), Sets Output Response Mode (WDT_CR.RMOD), and Enables the Timer (WDT_CR.WDT_EN)
Load Counter with Initial Timeout Value & Start Timer
Software Reads WDT_EOI or Writes 0x76 to WDT_CRR
Load Counter with Restart Timeout Value
Decrement Counter
Counter > 0
Counter == 0 and WDT_CR.RMOD == 1
Counter == 0 and WDT_CR.RMOD == 0
Assert Interrupt & Load Counter with Restart Timeout Value
Assert System Reset Request
Software Reads WDT_EOI or Writes 0x76 to WDT_CRR
Counter > 0
Decrement Counter
Counter == 0

If the counter reaches zero, the state changes based on the value of the output response mode setting defined in the rmod bit of the wdt_cr register. If the rmod bit of the wdt_cr register is 0, the output response mode is to generate a system reset request. In this case, the state changes to Assert System Reset Request. In response, the reset manager resets and disables the watchdog timer, and gives software the opportunity to reinitialize the timer.

If the rmod bit of the wdt_cr register is 1, the output response mode is to generate an interrupt. In this case, the state changes to Assert Interrupt and Load Counter with Restart Timeout Value. An interrupt to the processor is generated, and the watchdog counter is reloaded with the restart timeout value. The state then changes to the second Decrement Counter state, and the counter resumes decrementing. If software reads the wdt_eoi register, or writes 0x76 to the wdt_crr register, the state changes from Decrement Counter to Load Counter with Restart Timeout Value. In this state, the watchdog counter gets reloaded with the restart timeout value, and then the state changes back to the first Decrement Counter state. If the counter again reaches zero, the state changes to Assert System Reset Request. In response, the reset manager resets the watchdog timer, and gives software the opportunity to reinitialize the timer.

Watchdog Timer Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridge consist of the following regions:

- L4 Watchdog Module 0
- L4 Watchdog Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  For more information, refer to the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

L4 Watchdog Module Address Map

Registers in the L4 Watchdog module

Module Instance	Base Address
l4wd0	0xFFD02000
l4wd1	0xFFD03000

L4 Watchdog Module

Register	Offset	Width	Access	Reset Value	Description
wdt_cr on page 24-8	0x0	32	RW	0x2	Control Register
wdt_torr on page 24-9	0x4	32	RW	0xFF	Timeout Range Register
wdt_ccvr on page 24-11	0x8	32	RO	0x7FFFFFFF	Current Counter Value Register
wdt_crr on page 24-12	0xC	32	WO	0x0	Counter Restart Register
wdt_stat on page 24-12	0x10	32	RO	0x0	Interrupt Status Register.
wdt_eoi on page 24-13	0x14	32	RO	0x0	Interrupt Clear Register
cp_wdt_user_top_max on page 24-14	0xE4	32	RO	0x0	Component Parameters Register 5
cp_wdt_user_top_init_max on page 24-14	0xE8	32	RO	0x0	Component Parameters Register 4
cd_wdt_top_rst on page 24-15	0xEC	32	RO	0xFF	Component Parameters Register 3
cp_wdt_cnt_rst on page 24-15	0xF0	32	RO	0x7FFFFFFF	Component Parameters Register 2
wdt_comp_param_1 on page 24-16	0xF4	32	RO	0x10FF0254	Component Parameters Register 1
wdt_comp_version on page 24-19	0xF8	32	RO	0x3130362A	Component Version Register

Register	Offset	Width	Acces s	Reset Value	Description
wdt_comp_type on page 24-19	0xFC	32	RO	0x44570120	Component Type Register

wdt_cr

Contains fields that control operating functions.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD02000
l4wd1	0xFFD03000	0xFFD03000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														rmod RW 0x1	wdt_en RW 0x0

wdt_cr Fields

Bit	Name	Description	Access	Reset
1	rmod	Selects the output response generated to a timeout. **Value** **Description** 0x0 Generate a warm reset request 0x1 First generate an interrupt, and if it is not cleared by the time a second timeout occurs, then generate a warm reset request.	RW	0x1
0	wdt_en	This bit is used to enable and disable the watchdog. When disabled, the counter does not decrement. Thus, no interrupts or warm reset requests are generated. Once this bit has been enabled, it can only be cleared only by resetting the watchdog. **Value** **Description** 0x0 Watchdog disabled 0x1 Watchdog enabled	RW	0x0

wdt_torr

Contains fields that determine the watchdog timeout.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD02004
l4wd1	0xFFD03000	0xFFD03004

Offset: 0x4

Access: RW

<table>
<tr><th colspan="16">Bit Fields</th></tr>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">Reserved</td><td colspan="4">top_init
RW 0xF</td><td colspan="4">top
RW 0xF</td></tr>
</table>

wdt_torr Fields

<table>
<tr><th>Bit</th><th>Name</th><th>Description</th><th>Access</th><th>Reset</th></tr>
<tr><td>7:4</td><td>top_init</td><td>Used to select the timeout period that the watchdog counter restarts from for the first counter restart (kick). This register should be written after reset and before the watchdog is enabled. A change of the TOP_INIT is seen only once the watchdog has been enabled, and any change after the first kick is not seen as subsequent kicks use the period specified by the TOP bits. The timeout period (in clocks) is: t = 2**(16 + top_init)
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>Timeout = 65536 osc1_clk</td></tr>
<tr><td>0x1</td><td>Timeout = 131072 osc1_clk</td></tr>
<tr><td>0x2</td><td>Timeout = 262144 osc1_clk</td></tr>
<tr><td>0x3</td><td>Timeout = 524288 osc1_clk</td></tr>
<tr><td>0x4</td><td>Timeout = 1048576 osc1_clk</td></tr>
<tr><td>0x5</td><td>Timeout = 2097152 osc1_clk</td></tr>
<tr><td>0x6</td><td>Timeout = 4194304 osc1_clk</td></tr>
<tr><td>0x7</td><td>Timeout = 8388608 osc1_clk</td></tr>
<tr><td>0x8</td><td>Timeout = 16777216 osc1_clk</td></tr>
<tr><td>0x9</td><td>Timeout = 33554432 osc1_clk</td></tr>
<tr><td>0xa</td><td>Timeout = 67108864 osc1_clk</td></tr>
<tr><td>0xb</td><td>Timeout = 134217728 osc1_clk</td></tr>
<tr><td>0xc</td><td>Timeout = 268435456 osc1_clk</td></tr>
<tr><td>0xd</td><td>Timeout = 536870912 osc1_clk</td></tr>
<tr><td>0xe</td><td>Timeout = 1073741824 osc1_clk</td></tr>
<tr><td>0xf</td><td>Timeout = 2147483648 osc1_clk</td></tr>
</table></td><td>RW</td><td>0xF</td></tr>
</table>

Bit	Name	Description	Access	Reset
3:0	`top`	This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick). The timeout period (in clocks) is: t = 2**(16 + top) **Value** **Description** 0x0 Timeout = 65536 osc1_clk 0x1 Timeout = 131072 osc1_clk 0x2 Timeout = 262144 osc1_clk 0x3 Timeout = 524288 osc1_clk 0x4 Timeout = 1048576 osc1_clk 0x5 Timeout = 2097152 osc1_clk 0x6 Timeout = 4194304 osc1_clk 0x7 Timeout = 8388608 osc1_clk 0x8 Timeout = 16777216 osc1_clk 0x9 Timeout = 33554432 osc1_clk 0xa Timeout = 67108864 osc1_clk 0xb Timeout = 134217728 osc1_clk 0xc Timeout = 268435456 osc1_clk 0xd Timeout = 536870912 osc1_clk 0xe Timeout = 1073741824 osc1_clk 0xf Timeout = 2147483648 osc1_clk	`RW`	`0xF`

wdt_ccvr

See Field Description

Module Instance	Base Address	Register Address
`l4wd0`	`0xFFD02000`	`0xFFD02008`
`l4wd1`	`0xFFD03000`	`0xFFD03008`

Offset: `0x8`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wdt_ccvr RO 0x7FFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wdt_ccvr RO 0x7FFFFFFF															

wdt_ccvr Fields

Bit	Name	Description	Access	Reset
31:0	wdt_ccvr	This register provides the current value of the internal counter.	RO	0x7FFFFFFF

wdt_crr

Restarts the watchdog.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD0200C
l4wd1	0xFFD03000	0xFFD0300C

Offset: 0xC

Access: WO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								wdt_crr WO 0x0							

wdt_crr Fields

Bit	Name	Description	Access	Reset
7:0	wdt_crr	This register is used to restart the watchdog counter. As a safety feature to prevent accidental restarts, the kick value of 0x76 must be written. A restart also clears the watchdog interrupt. **Value** **Description** 0x76 Value to write to restart watchdog timer	WO	0x0

wdt_stat

Provides interrupt status

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD02010
l4wd1	0xFFD03000	0xFFD03010

Offset: 0x10

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															wdt_stat RO 0x0

wdt_stat Fields

Bit	Name	Description	Access	Reset
0	wdt_stat	Provides the interrupt status of the watchdog. **Value** **Description** 0x1 Interrupt is active 0x0 Interrupt is inactive	RO	0x0

wdt_eoi

Clears the watchdog interrupt when read.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD02014
l4wd1	0xFFD03000	0xFFD03014

Offset: 0x14

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															wdt_eoi RO 0x0

wdt_eoi Fields

Bit	Name	Description	Access	Reset
0	`wdt_eoi`	Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter.	RO	0x0

cp_wdt_user_top_max

This is a constant read-only register that contains encoded information about the component's parameter settings.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020E4
l4wd1	0xFFD03000	0xFFD030E4

Offset: 0xE4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cp_wdt_user_top_max RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cp_wdt_user_top_max RO 0x0															

cp_wdt_user_top_max Fields

Bit	Name	Description	Access	Reset
31:0	`cp_wdt_user_top_max`	Upper limit of Timeout Period parameters.	RO	0x0

cp_wdt_user_top_init_max

This is a constant read-only register that contains encoded information about the component's parameter settings

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020E8
l4wd1	0xFFD03000	0xFFD030E8

Offset: 0xE8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cp_wdt_user_top_init_max RO 0x0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cp_wdt_user_top_init_max RO 0x0															

cp_wdt_user_top_init_max Fields

Bit	Name	Description	Access	Reset
31:0	cp_wdt_user_top_init_max	Upper limit of Initial Timeout Period parameters.	RO	0x0

cd_wdt_top_rst

This is a constant read-only register that contains encoded information about the component's parameter settings.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020EC
l4wd1	0xFFD03000	0xFFD030EC

Offset: 0xEC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cd_wdt_top_rst RO 0xFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cd_wdt_top_rst RO 0xFF															

cd_wdt_top_rst Fields

Bit	Name	Description	Access	Reset
31:0	cd_wdt_top_rst	Contains the reset value of the WDT_TORR register.	RO	0xFF

cp_wdt_cnt_rst

This is a constant read-only register that contains encoded information about the component's parameter settings.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020F0
l4wd1	0xFFD03000	0xFFD030F0

Offset: 0xF0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
cp_wdt_cnt_rst RO 0x7FFFFFFF															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cp_wdt_cnt_rst RO 0x7FFFFFFF															

cp_wdt_cnt_rst Fields

Bit	Name	Description	Access	Reset
31:0	cp_wdt_cnt_rst	The timeout period range is fixed. The range increments by the power of 2 from 2 to the 16 to 2 to the 31.	RO	0x7FFFFFFF

wdt_comp_param_1

This is a constant read-only register that contains encoded information about the component's parameter settings.

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020F4
l4wd1	0xFFD03000	0xFFD030F4

Offset: 0xF4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			cp_wdt_cnt_width RO 0x10					cp_wdt_dflt_top_init RO 0xF				cp_wdt_dflt_top RO 0xF			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			cp_wdt_dflt_rpl RO 0x0			cp_wdt_apb_data_width RO 0x2		cp_wdt_pause RO 0x0	cp_wdt_use_fix_top RO 0x1	cp_wdt_hc_top RO 0x0	cp_wdt_hc_rpl RO 0x1	cp_wdt_hc_rmod RO 0x0	cp_wdt_dual_top RO 0x1	cp_wdt_dflt_rmod RO 0x0	cp_wdt_always_en RO 0x0

wdt_comp_param_1 Fields

Bit	Name	Description	Access	Reset
28:24	cp_wdt_cnt_width	Width of counter in bits less 16. **Value** **Description** 0x10 Counter width is 32 bits	RO	0x10
23:20	cp_wdt_dflt_top_init	Specifies the initial timeout period that is available directly after reset. **Value** **Description** 0xf Initial timeout period is 15 (2**31 cycles).	RO	0xF
19:16	cp_wdt_dflt_top	Specifies the timeout period that is available directly after reset. **Value** **Description** 0xf Timeout period is 15 (2**31 cycles).	RO	0xF
12:10	cp_wdt_dflt_rpl	Specifies the reset pulse length in cycles. **Value** **Description** 0x0 Reset pulse length of 2 cycles.	RO	0x0

Bit	Name	Description	Access	Reset
9:8	`cp_wdt_apb_data_width`	APB Bus Width **Value** **Description** 0x2 APB Data Width is 32 Bits	RO	0x2
7	`cp_wdt_pause`	Should specify if the pause input is included or not. However, this field is always hardwired to 0 so you can't figure this out by reading this field. The pause input is included and can be used to pause the watchdog when the MPU is in debug mode.	RO	0x0
6	`cp_wdt_use_fix_top`	Specifies if the watchdog uses the pre-defined timeout values or if these were overriden with customer values when the watchdog was configured. **Value** **Description** 0x1 Use pre-defined (fixed) timeout values (range from 2**16 to 2**31)	RO	0x1
5	`cp_wdt_hc_top`	Specifies if the timeout period is programmable or hardcoded. **Value** **Description** 0x0 Timeout period is programmable.	RO	0x0
4	`cp_wdt_hc_rpl`	Specifies if the reset pulse length is programmable or hardcoded. **Value** **Description** 0x1 Reset pulse length is hardcoded.	RO	0x1
3	`cp_wdt_hc_rmod`	Specifies if response mode (when counter reaches 0) is programmable or hardcoded. **Value** **Description** 0x0 Output response mode is programmable.	RO	0x0
2	`cp_wdt_dual_top`	Specifies whether a second timeout period that is used for initialization prior to the first kick is present or not. **Value** **Description** 0x1 Second timeout period is present	RO	0x1

Bit	Name	Description	Access	Reset
1	cp_wdt_dflt_rmod	Specifies default output response mode after reset. **Value** **Description** 0x0 Generate a warm reset request (don't generate an interrupt first)	RO	0x0
0	cp_wdt_always_en	Specifies whether watchdog starts after reset or not. **Value** **Description** 0x0 Watchdog disabled on reset	RO	0x0

wdt_comp_version

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020F8
l4wd1	0xFFD03000	0xFFD030F8

Offset: 0xF8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wdt_comp_version RO 0x3130362A															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wdt_comp_version RO 0x3130362A															

wdt_comp_version Fields

Bit	Name	Description	Access	Reset
31:0	wdt_comp_version	ASCII value for each number in the version, followed by *. For example, 32_30_31_2A represents the version 2.01*.	RO	0x31303 62A

wdt_comp_type

Module Instance	Base Address	Register Address
l4wd0	0xFFD02000	0xFFD020FC
l4wd1	0xFFD03000	0xFFD030FC

Offset: 0xFC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
wdt_comp_type RO 0x44570120															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wdt_comp_type RO 0x44570120															

wdt_comp_type Fields

Bit	Name	Description	Access	Reset
31:0	wdt_comp_type	Designware Component Type number = 0x44_57_01_20.	RO	0x44570120

Document Revision History

Table 24-1: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• "FPGA Interface" section added • Added address map and register descriptions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting updates
November 2012	1.2	Minor updates.
May 2012	1.1	Added programming model and address map and register definitions sections.
January 2012	1.0	Initial release.

CAN Controller Introduction 25

2014.06.30

cv_54025 Subscribe Send Feedback

The hard processor system (HPS) provides two controller area network (CAN) controllers for serial communication with the Cortex-A9 microprocessor unit (MPU) subsystem host processor and the direct memory access (DMA) controller using the CAN protocol. The CAN controllers are instances of the Bosch D_CAN controller and compliant with ISO 11898-1.

Features of the CAN Controller

The CAN controllers offer the following features:

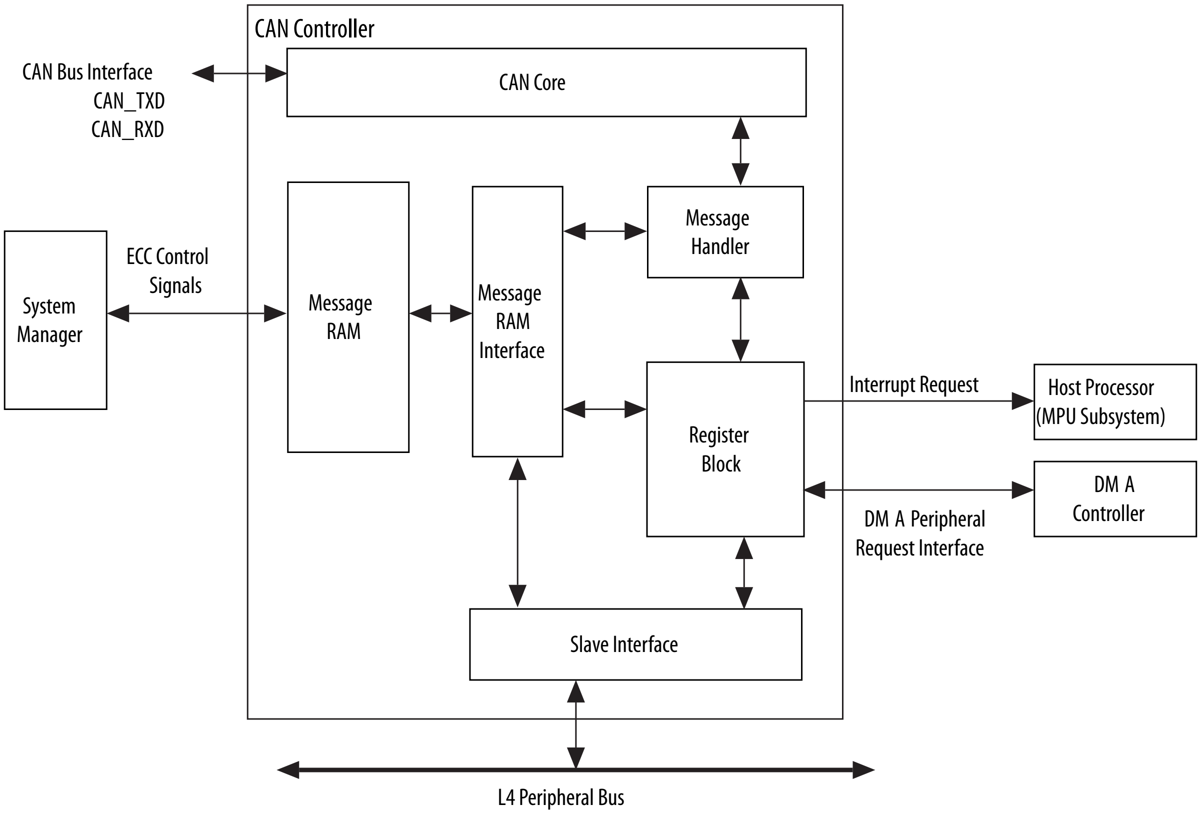
- Compliant with CAN Protocol Specification 2.0 parts A and B, available from the Bosch website.
- Programmable communication rate up to 1 Mbps
- Holds up to 128 messages
- Error correction code (ECC)
- 11-bit standard and 29-bit extended identifiers
- Programmable loopback mode
- External direct memory access (DMA) controller for large data transfers
- Automatic retransmission

Related Information

http://www.bosch-semiconductors.de/en/ubk_semiconductors/homepage/homepage_1.html

CAN Controller Block Diagram and System Integration

Figure 25-1: CAN Controller Block Diagram



The CAN controller consists of the following modules and interfaces:

- CAN core
  - Connects to the CAN bus interface
  - Handles all ISO 11898-1 protocol functions
- Message handler
  - State machine that controls the data transfer between the message RAM and CAN core.
  - Handles acceptance filtering and the interrupt generation
- Message RAM
  - Storage for up to 128 messages objects
  - Single bit error correction and double bit error detection
- Message RAM interface
  - Two separate interfaces, `IF1` and `IF2`
- Register block
  - Control and status registers (CSR) for module setup and indirect message object access.
  - All host processor accesses to the message RAM are relayed through the message RAM interface.
- Level 4 (L4) slave interface for CSR accesses

Functional Description of the CAN Controller

The CAN controllers perform communication according to the CAN protocol version 2.0 parts A and B. All communication on the CAN bus is through message objects. The CAN controller stores message objects in its internal message RAM. The host processor cannot access the message RAM directly, instead the `IF1` and `IF2` message interface register sets provide the host processor access to the messages. Messages are passed between the message RAM and the CAN core by the message handler. The message handler is also responsible for message level responsibilities such as acceptance filtering, interrupt generation, and transmission request generation.

Message Object

The message RAM can store up to 128 message objects. To avoid conflicts between host processor accesses to the message RAM and CAN message reception and transmission, the host processor does not access the message objects directly. Accesses are handled through the `IF1` and `IF2` message interface registers.

The following table shows the structure of a message object, where the first row contains the message object control flags, the second row contains the message object masks, and the third row is the CAN message.

Table 25-1: Message Object Structure

MsgVal				NewDat	MsgLst	IntPnd		TxIE	RxIE	RmtEn	TxRqst	EoB
UMask	Msk [28:0]	MXtd	MDir									

	ID [28:0]	Xtd	Dir	DLC [3:0]	Data 0 [7:0]	Data 1 [7:0]	Data 2 [7:0]	Data 3 [7:0]	Data 4 [7:0]	Data 5 [7:0]	Data 6 [7:0]	Data 7 [7:0]

Message Object Control Flags

Message Valid (`MsgVal`)

- 0= The message object is ignored by the message handler
- 1= The message object is configured and should be considered by the message handler

The host processor must set the `MsgVal` bit of all unused message objects to 0 before it resets the initialization bit (`Init`) of the CAN control register (`CCTRL`) to initialize the CAN controller. `MsgVal` must also be set to 0 when the message object is no longer in use.

The `MsgVal` field is directly readable from the message valid registers (`MOVALA`, `MOVALB`, `MOVALC`, `MOVALD`, and `MOVALX`). However, to write to the `MsgVal` field for a specific message object, the host processor must write to the message interface registers.

New Data (`NewDat`)

- 0= No new data has been written into the data portion of this message object by the message handler since last time this flag was cleared by the host processor.
- 1= The message handler or the host processor has written new data to the data portion of this message object.

The `NewDat` field is directly readable from the new data registers (`MONDA`, `MONDB`, `MONDC`, `MONDD`, and `MONDX`). However, to write to the `NewDat` field for a specific message object, the host processor must write to the message interface registers.

Message Lost (`MsgLst`)

- 0= No message lost since last time this bit is was last reset by the host processor
- 1= The message handler stored a new message into this object when the `NewDat` bit was still set, indicating the host processor has lost a message.

`MsgLst` is valid only in message objects with the message direction bit (`Dir`) set to receive.

Interrupt Pending (IntPnd)

- 0= This message object is not the source of an interrupt.
- 1= This message object is the source of an interrupt. The interrupt identifier field in the CAN interrupt register (CIR) points to this message object if there is no other interrupt source with higher priority.

The IntPnd field is directly readable from the interrupt pending registers (MOIPA, MOIPB, MOIPC, MOIPD, and MOIPX). However, to write to the IntPnd field for a specific message object, the host processor must write to the message interface registers.

Transmit Interrupt Enable (TxIE)

- 0= TxIE is disabled. IntPnd is left unchanged after the successful transmission of a frame.
- 1= TxIE is enabled. IntPnd is set after a successful transmission of a frame.

Receive Interrupt Enable (RxIE)

- 0= RxIE is disabled. IntPnd is left unchanged after a successful reception of a frame.
- 1= RxIE is enabled. IntPnd is set after a successful reception of a frame.

Remote Enable (RmtEn)

- 0= RmtEn is disabled. At the reception of a remote frame, TxRqst is left unchanged.
- 1= RmtEn is enabled. At the reception of a remote frame, TxRqst is set.

Transmit Request (TxRqst)

- 0= This message object is not waiting for transmission.
- 1= The transmission of this message object is requested and is not complete.

The TxRqst field is directly readable from the transmission request registers (MOTRA, MOTRB, MOTRC, MOTRD, and MOTRX). However, to write to the TxRqst field for a specific message object, the host processor must write to the message interface registers.

End of Block (EoB)

- 0= Message object belongs to a FIFO buffer block and is not the last message object of that FIFO buffer block.
- 1= Single message object or last message object of a FIFO buffer block.

This bit is used to concatenate two or more message objects (up to 128) to build a FIFO buffer. For single message objects (not belonging to a FIFO buffer), this bit must always be set to 1.

Message Object Mask Bits

The message object mask bits, together with the arbitration bits, are used for acceptance filtering of incoming messages.

Use Acceptance Mask (UMask) on page 25-6

Identifier Mask (Msk[28:0]) on page 25-6

Extended Identifier Mask (MXtd) on page 25-6

Mask Message Direction (MDir) on page 25-6

Use Acceptance Mask (UMask)

- 0= Ignore mask. Msk[28:0], MXtd, and Mdir have no effect on acceptance filtering. For an incoming message to be accepted, all the following conditions must be met:
  - The received message is a data frame with message direction set to 0 (receive) or is a remote frame with message direction set to 1 (transmit).
  - The received message identifier matches the message identifier (ID[28:0]) of the message object.
  - The received identifier extension bit matches the identifier extension bit (Xtd) of the message object.
- 1= Use mask (Msk[28:0], MXtd, and MDir) for acceptance filtering if the respective mask bits are set up for acceptance filtering. For an incoming message to be accepted, all the following conditions must be met:
  - The received message is a data frame with message direction set to 0 (receive) or is a remote frame with message direction set to 1 (transmit) with the MDir mask bit enabled.
  - The received message identifier matches the message identifier (ID[28:0]) of the message object with the Msk[28:0] mask bits enabled
  - The received identifier extension bit matches the identifier extension bit (Xtd) of the message object, with the MXtd mask bit enabled.

Note: If the UMask bit is set to 1, the message object's mask bits have to be programmed during the initialization of the message object before MsgVal is set to 1.

Identifier Mask (Msk[28:0])

The identifier mask filters the corresponding bits in ID[28:0].

- 0= The corresponding identifier bit has no effect on the acceptance filter.
- 1= The corresponding identifier bit is used for acceptance filtering.

Extended Identifier Mask (MXtd)

- 0= The extended frame identifier bit (Xtd) has no effect on the acceptance filtering
- 1= The extended frame identifier bit (Xtd) is used for acceptance filtering

When 11-bit (standard) identifiers are used for a message object, the identifiers of received data frames are written to bits ID28 to ID18. For acceptance filtering, only these bits, together with mask bits Msk28 to Msk18, are used.

Mask Message Direction (MDir)

- 0= The message direction bit (Dir) has no effect on the acceptance filtering.
- 1= The message direction bit (Dir) is used for acceptance filtering.

Important: Altera recommends always setting MDir to 1. Ignoring the message direction bit is an advanced technique that must be handled with great care.

CAN Message Bits

The arbitration fields ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask fields Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored to the valid message object with matching identifier when the direction is set to receive a data frame or transmit a remote frame. Extended frames can be stored only in message objects with Xtd is set to 1, standard frames in message objects with Xtd is set to 0. If a received message (data frame or remote frame) matches more than one valid message object, it is stored to the object with the lowest message number.

Message Identifier (ID[28:0])

- ID28-ID0: 29-bit identifier (extended frame)
- ID28-ID18: 11-bit identifier (standard frame)

Extended Frame Identifier (Xtd)

- 0= The 11-bit (standard) identifier is used for this message object.
- 1= The 29-bit (extended) identifier is used for this message object.

Message Direction (Dir)

- 0= receive direction. When TxRqst is set to 1, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, that message is stored in this message object.
- 1= transmit direction. When TxRqst is set to 1, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TxRqst bit of this message object is set to 1 (if RmtEn = 1).

Data Length Code (DLC[3:0])

DLC specifies the number of data bytes in the data frame. The maximum number is eight.

The data length code (DLC) of a message object must be defined the same as in all the corresponding objects with the same identifier in all CAN devices. When the message handler stores a data frame, it sets the DLC field to the value provided in the received message.

Data Bytes 0-7 (Data 0[7:0] - Data 7[7:0])

The data bytes in a CAN data frame.

Message Interface Registers

There are two sets of message interface registers, IF1 and IF2, that provide a means for a host processor or DMA controller to access any message object indirectly. Message objects are transferred between the message RAM and the message buffer registers as a single, atomic operation maintaining data consistency across the message.

Table 25-2: Message Interface Register Set

This table lists the structure of each message interface register set, where *x* represents either set 1 or set 2.

Register Type	Register	Name	Description
Command	IFxCMR	IFx command register	Specifies the transfer direction and selects the portions of the message object to transfer

Register Type	Register	Name	Description
Message buffer	IFxMSK	IFx mask register	Provides access to the Msk, MDir, and MXtd mask fields of the message object
	IFxARB	IFx arbitration register	Provides access to the ID, Dir, Xtd, and MsgVal arbitration fields of the message object
	IFxMCTR	IFx message control register	Provides access to the DLC, EoB, TxRqst, RmtEn, RxIE, TxIE, UMask, IntPnd, MsgLst, and NewDat fields of the message object
	IFxDA	IFx data A register	Provides access to data bytes 0-3 of the message object
	IFxDB	IFx data B register	Provides access to data bytes 4-7 of the message object

DMA Mode

The CAN controller can issue DMA controller requests to transfer data between one or both of the message interface registers and system memory. The CAN controller has two DMA request interfaces, called can_if1dma and can_if2dma. The CAN peripheral request interfaces are shared with the FPGA DMA peripheral request interfaces. To use the DMA peripheral request interface, the host processor must access the CAN control register (CCTRL) in the protocol group (protogrp). The peripheral request interface is selected through the system manager.

To activate the DMA support feature and initiate a transfer, write a 1 to the DMAactive bit in the appropriate IF command register (IFxCMR) in the message interface group (msgifgrp). After the message object transfer is completed, the CAN controller issues a DMA peripheral request to perform the next message object transfer. The request remains active until the first read or write to the message interface register.

Related Information

- **DMA Controller** on page 16-1
- **Features of the System Manager** on page 5-1

Automatic Retransmission

The CAN controller provides a means for automatic retransmission of frames that have lost arbitration or have errors during transmission. Retransmission happens automatically without user intervention or notification. Normal confirmation is given when the transmission is successfully complete.

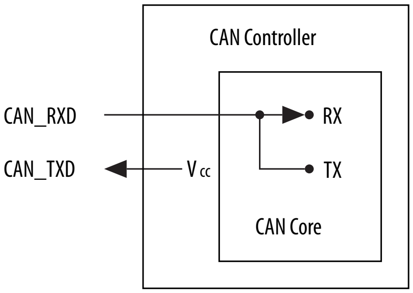
Test Mode

To enable test mode, set the test mode enable bit (Test) in the CCTRL register to 1. This action activates write access to the CAN test register (CTR).

Silent Mode

The CAN controller is set in the silent mode by programming the silent mode (`Silent`) bit in the test register (`CTR`) to 1. In silent mode, the CAN controller is able to receive valid data frames and valid remote frames, but it holds the `CAN_TXD` pin high, sending no data to the CAN bus. The silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (acknowledge bits, error frames). In ISO 11898-1, the silent mode is called the *bus monitoring mode*.

Figure 25-2: CAN Core in Silent Mode

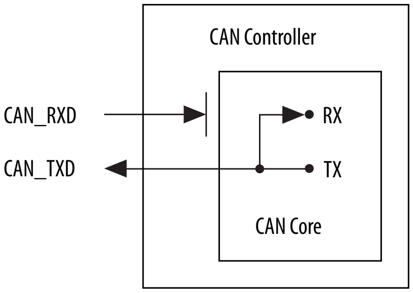


Loopback Mode

The CAN controller is set in loopback mode by programming the loopback mode (`LBack`) bit in the test register (`CTR`) to 1. In loopback mode, the CAN controller treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) to a receive buffer.

To be independent from external stimulation, the CAN controller ignores acknowledge errors in loopback mode. In this mode, the CAN controller provides internal feedback from its transmit (TX) output to its receive (RX) input. The actual value of the input pin is disregarded by the CAN controller.

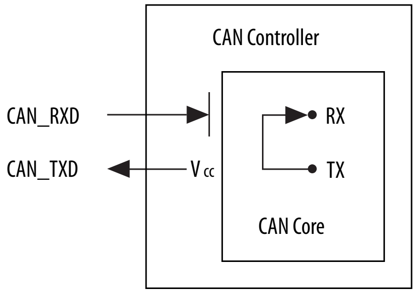
Figure 25-3: CAN Core in Loopback Mode



Combined Mode

The CAN controller is set in combined loopback and silent mode by programming the silent mode (`Silent`) and loopback mode (`LBack`) bits in the test register (`CTR`) to 1. Combined mode can be used for testing the CAN hardware without affecting other devices connected to the CAN bus. In this mode, the `CAN_RXD` pin is disconnected from the CAN core and the `CAN_TXD` pin is held high.

Figure 25-4: CAN Core in Combined Mode



L4 Slave Interface

The host processor accesses data, control, and status information of the CAN controller through the L4 slave interface. The slave interface supports 32-bit accesses only.

Note: This interface does not support error responses.

Clocks

The CAN controller operates on the `l4_sp_clk` and `can_clk` clock inputs. The `l4_sp_clk` clock is used by the L4 slave interface and the `can_clk` is used to operate the CAN core.

The `can_clk` clock must be programmed to be at least eight times the CAN bus interface speed. For example, for a CAN bus interface operating at a 1 Mbps baud rate, the `can_clk` clock must be set to at least 8 MHz. The `l4_sp_clk` clock can operate at a clock frequency that is equal to or greater than the `can_clk` frequency.

Related Information

Clock Manager on page 2-1
For more information about the `l4_sp_clk` and `can_clk` clocks, refer to the *Clock Manager* chapter.

Software Reset

Software initialization is done by setting the `Init` bit in the CAN control register (`CCTRL`) in the protocol group (`protogrp`) in the CAN controller register map. This bit is set through the CAN protocol when a bus off condition occurs on the CAN link. The bit is also set through the hardware reset input described in Hardware Reset.

Due to the synchronization mechanism between the two clock domains, there might be a delay until the value written to the `Init` bit can be read back. To assure that the previous value written has been accepted, read the `Init` bit before setting it to a new value.

The bus off recovery sequence cannot be shortened by setting or resetting the `Init` bit. For more information about bus off, refer to the CAN Protocol Specification 2.0 parts A and B, available from the Bosch website.

Related Information

- **Hardware Reset** on page 25-11
- **http://www.bosch-semiconductors.de/en/ubk_semiconductors/homepage/homepage_1.html**

Hardware Reset

Each CAN controller has a separate reset signal. The reset manager drives the signals on a cold or warm reset. The reset signal is synchronized to both clock domains and applied to the appropriate logic within the CAN controllers.

Related Information

Reset Manager on page 3-1
For more information, refer to the *Reset Manager* chapter.

Interrupts

Each CAN controller generates two interrupt signals. One signal indicates error and status interrupts and the other signal indicates message object interrupts. Both interrupt signals connect to the global interrupt controller (GIC). Interrupts are enabled in the CAN control register (`CCTRL`) in the protocol group (`protogrp`). The CAN interrupt register (`CIR`) in the protocol group (`protogrp`) indicates the highest priority interrupt that is pending.

Error Interrupts

The following error conditions generate interrupts:

- Bus off—when the transmit error count is equal to or greater than 256, the bus off (`BOff`) bit in the CAN status register (`CSTS`) in the protocol group (`protogrp`) is set to 1.
- Error warning—when either the transmit error counter or the receive error counters reaches 96, the error warning status (`EWarn`) bit in the CAN status register (`CSTS`) in the protocol group (`protogrp`) is set to 1.

Status Interrupts

The following status conditions generate interrupts:

- Receive OK—when the CAN controller receives a message successfully, the `RxOK` bit in the CAN status register (`CSTS`) in the protocol group (`protogrp`) is set to 1.
- Transmit OK—when the CAN controller transmits a message successfully, the `TxOK` bit in the CAN status register (`CSTS`) in the protocol group (`protogrp`) is set to 1.
- Last error code—when a message is received or transmitted with an error, the `LEC` bits in the CAN status register (`CSTS`) in the protocol group (`protogrp`) are set according to the error type.

Message Object Interrupts

The `IntPnd` bits from the message objects can generate interrupts when the corresponding message object `TxIE` bit or `RxIE` bit is set to 1. The table lists the location of message object interrupt information in the interrupt pending registers. The interrupt pending registers are located in the message handler group (`msghandgrp`).

Table 25-3: Message Object Interrupt Registers

Register	Title	Message Objects
`MOIPA`	Interrupt pending A register	1 to 32

Register	Title	Message Objects
MOIPB	Interrupt pending B register	33 to 64
MOIPC	Interrupt pending C register	65 to 96
MOIPD	Interrupt pending D register	97 to 128

The MOIPX register allows software to quickly detect which message object group has a pending interrupt.

CAN Controller Programming Model

Software Initialization

The software initialization is started by setting the Init bit in the CAN control register (CCTRL) to 1. While the Init bit is 1, messages are not transferred to or from the CAN bus, and the CAN_TXD CAN bus output is held in the high state. Setting the Init bit does not change any configuration registers.

To initialize the CAN controller, the host processor must program the CAN bit timing (CBT) register and message objects that will be used for CAN communication. If a message object is not needed, it is sufficient to set the MsgVal bit of the message object to not valid (0), which is the default after RAM initialization. You must set up the entire message object before setting MsgVal bit to valid (1). The message objects are set up through either message interface register set.

Access to the CAN bit timing (CBT) register is only enabled when the configuration change enable (CCE) and Init bits in the CAN control register (CCTRL) are both set to 1.

Setting the Init bit to 0 finishes the software initialization. The CAN core synchronizes itself to the data transfer on the CAN bus by waiting for the bus to reach an idle state before it can take part in bus activities and message transfers.

The initialization of the message objects is independent of the CAN controller initialization and can be done anytime, but the message objects should all be configured to particular identifiers or set to not valid before message transferring begins.

On power up, the message RAM has to be initialized. To initialize RAM, set the Init bit to 1, then set the RAMInit bit in the CAN function register (CFR) in the protocol group (protogrp) to 1. The RAMInit bit returns to 0 when RAM initialization completes. During RAM initialization, all message objects are cleared to zero and the RAM ECC bits are initialized. Access to RAM is not allowed prior to or during RAM initialization.

CAN Message Transfer

Once the CAN controller is initialized, the CAN controller synchronizes itself to the CAN bus and starts transferring messages.

Received messages are stored to their appropriate message objects, if they pass the message handler's acceptance filtering. The entire message, including all arbitration bits, Xtd, Dir, DLC, eight data bytes, the mask and control bits UMask, MXtd, MDir, EoB, MsgLst, RxIE, TxIE, and RmtEn, is stored in the message object. Masked arbitration bits might change in the message object when a received message is stored.

The host processor may read or update each message at any time using the message interface registers. The message handler guarantees data consistency when the host processor accesses the message object at the same time the message is being transferred to or from RAM.

Messages to be transmitted are updated by the host processor. If a permanent message object (arbitration and control bits set up during configuration are unchanged for multiple CAN transfers) exists for the message, only the data bytes need to be updated. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time they are transmitted, according to their internal priority. The message object numbers are 1 to 128, with 1 being the lowest internal priority and 128 the highest priority. Messages may be updated or set to invalid (`MsgVal`=0) at anytime, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started.

Depending on the configuration on the message object, the transmission of a message may be requested automatically by the reception of a remote frame with a matching identifier. Remote frames are frames used to request a particular message on the CAN network.

Note: For ease in programming, Altera recommends using one `IF` message interface for all receive direction activity and the other `IF` message interface for all transmit direction activity.

Message Object Reconfiguration for Frame Reception

To configure a message object to receive data frames, set the `Dir` field to 0.

To configure a message object to receive remote frames, set the `Dir` field to 1, set `UMask` to 1, and set `RmtEn` to 0.

To avoid modifying an object while it is being transmitted, you must set `MsgVal` to 0 before changing any of the following configuration and control bits:

- `ID[28:0]`
- `Xtd`
- `DLC[3:0]`
- `RxIE`
- `TxIE`
- `RmtEn`
- `EoB`
- `UMask`
- `Msk[28:0]`
- `MXtd`
- `MDir`

The following fields of a message object can be changed without clearing `MsgVal`:

- `Data0[7:0] to Data7[7:0]`
- `TxRqst`
- `NewDat`
- `MsgLst`
- `IntPnd`

Message Object Reconfiguration for Frame Transmission

To configure a message object to transmit data frames, set the `Dir` field to 1, and either set `UMask` to 0 or set `RmtEn` to 1.

Before changing any of the following configuration and control bits, you must set `MsgVal` to 0:

- `Dir`
- `RxIE`
- `TxIE`
- `RmtEn`
- `EoB`
- `UMask`
- `Msk[28:0]`
- `MXtd`
- `MDir`

The following fields of a message object can be changed without clearing `MsgVal`:

- `ID[28:0]`
- `Xtd`
- `DLC[3:0]`
- `Data0[7:0] to Data7[7:0]`
- `TxRqst`
- `NewDat`
- `MsgLst`
- `IntPnd`

CAN Controller Address Map and Register Definitions

The address map and register definitions for the HPS-FPGA bridge consist of the following regions:

- CAN Controller Module 0
- CAN Controller Module 1

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
  The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter.
- **http://www.altera.com/literature/hb/cyclone-v/hps.html**

CAN Controller Module Address Map

Registers in the CAN Controller module NOTE: These descriptions apply only to SoC devices that support the CAN module.

Module Instance	Base Address
can0	0xFFC00000
can1	0xFFC01000

Protocol Group

Register	Offset	Width	Access	Reset Value	Description
CCTRL on page 25-17	0x0	32	RW	0x1	Control Register
CSTS on page 25-21	0x4	32	RO	0x7	Status Register
CERC on page 25-24	0x8	32	RO	0x0	Error Counter Register
CBT on page 25-25	0xC	32	RW	0x2301	Bit Timing / BRP Extension Register
CIR on page 25-26	0x10	32	RO	0x0	Interrupt Register
CTR on page 25-27	0x14	32	RW	0x0	Test Register
CFR on page 25-29	0x18	32	RW	0x0	Function Register
CRR on page 25-30	0x20	32	RO	0x11161128	Core Release Register
HWS on page 25-30	0x24	32	RO	0x3	Hardware Configuration Status Register

Message Handler Group

Register	Offset	Width	Access	Reset Value	Description
MOTRX on page 25-33	0x84	32	RO	0x0	Transmission Request X Register
MOTRA on page 25-39	0x88	32	RO	0x0	Transmission Request A Register
MOTRB on page 25-48	0x8C	32	RO	0x0	Transmission Request B Register
MOTRC on page 25-56	0x90	32	RO	0x0	Transmission Request C Register
MOTRD on page 25-65	0x94	32	RO	0x0	Transmission Request D Register
MONDX on page 25-74	0x98	32	RO	0x0	New Data X Register
MONDA on page 25-79	0x9C	32	RO	0x0	New Data A Register
MONDB on page 25-90	0xA0	32	RO	0x0	New Data B Register
MONDC on page 25-101	0xA4	32	RO	0x0	New Data C Register
MONDD on page 25-112	0xA8	32	RO	0x0	New Data D Register
MOIPX on page 25-124	0xAC	32	RO	0x0	Interrupt Pending X Register
MOIPA on page 25-130	0xB0	32	RO	0x0	Interrupt Pending A Register

Register	Offset	Width	Acces s	Reset Value	Description
MOIPB on page 25-138	0xB4	32	RO	0x0	Interrupt Pending B Register
MOIPC on page 25-147	0xB8	32	RO	0x0	Interrupt Pending C Register
MOIPD on page 25-156	0xBC	32	RO	0x0	Interrupt Pending D Register
MOVALX on page 25-164	0xC0	32	RO	0x0	Message Valid X Register
MOVALA on page 25-170	0xC4	32	RO	0x0	Message Valid A Register
MOVALB on page 25-179	0xC8	32	RO	0x0	Message Valid B Register
MOVALC on page 25-187	0xCC	32	RO	0x0	Message Valid C Register
MOVALD on page 25-196	0xD0	32	RO	0x0	Message Valid D Register

Message Interface Group

Register	Offset	Width	Acces s	Reset Value	Description
IF1CMR on page 25-207	0x100	32	RW	0x1	IF1 Command Register
IF1MSK on page 25-211	0x104	32	RW	0xFFFFFFFF	IF1 Mask Register
IF1ARB on page 25-213	0x108	32	RW	0x0	IF1 Arbitration Register
IF1MCTR on page 25-214	0x10C	32	RW	0x0	IF1 Message Control Register
IF1DA on page 25-217	0x110	32	RW	0x0	IF1 Data A Register
IF1DB on page 25-218	0x114	32	RW	0x0	IF1 Data B Register
IF2CMR on page 25-219	0x120	32	RW	0x1	IF2 Command Register
IF2MSK on page 25-223	0x124	32	RW	0xFFFFFFFF	IF2 Mask Register
IF2ARB on page 25-225	0x128	32	RW	0x0	IF2 Arbitration Register
IF2MCTR on page 25-226	0x12C	32	RW	0x0	IF2 Message Control Register
IF2DA on page 25-229	0x130	32	RW	0x0	IF2 Data A Register
IF2DB on page 25-230	0x134	32	RW	0x0	IF2 Data B Register

Protocol Group Register Descriptions

These registers are related to the CAN protocol controller in the CAN Core. They control the operating modes and the configuration of the CAN bit timing and provide status information.

Offset: 0x0

CCTRL on page 25-17
Control Register

CSTS on page 25-21
Status Register

CERC on page 25-24
Error Counter Register

CBT on page 25-25
This register is only writable if bits CCTRL.CCE and CCTRL.Init are set. The CAN bit time may be programed in the range of [4 .. 25] time quanta. The CAN time quantum may be programmed in the range of [1 .. 1024] CAN_CLK periods. For details see Application Note 001 "Configuration of Bit Timing". The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. TSeg1 is the sum of Prop_Seg and Phase_Seg1. TSeg2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [TSeg1 + TSeg2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

CIR on page 25-26
If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If IntID is different from 0x00 and CCTRL.MIL is set, the interrupt port CAN_INT_MO is active. The interrupt port remains active until IntID is back to value 0x00 (the cause of the interrupt is reset) or until CCTRL.MIL is reset. If CCTRL.ILE is set and CCTRL.MIL is reseted the Message Object interrupts will be routed to interrupt port CAN_INT_STATUS. The interrupt port remains active until IntID is back to value 0x00 (the cause of the interrupt is reset) or until CCTRL.MIL is set or CCTRL.ILE is reset. The Message Object's interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the Message Object's IntPnd bit.

CTR on page 25-27
The Test Mode is entered by setting bit CCTRL.Test to one. In Test Mode the bits EXL, Tx1, Tx0, LBack and Silent in the Test Register are writable. Bit Rx monitors the state of pin CAN_RXD and therefore is only readable. All Test Register functions are disabled when bit Test is reset to zero. Loop Back Mode and CAN_TXD Control Mode are hardware test modes, not to be used by application programs. Note: This register is only writable if bit CCTRL.Test is set.

CFR on page 25-29
The Function Register controls the features RAM_Initialisation and Power_Down also by application register. The CAN module can be prepared for Power_Down by setting the port CAN_CLKSTOP_REQ to one or writing to CFR.ClkStReq a one. The power down state is left by setting port CAN_CLKSTOP_REQ to zero or writing to CFR.ClkStReq a zero, acknowledged by CAN_CLKSTOP_ACK is going to zero as well as CFR.ClkStAck. The CCTRL.Init bit is left one and has to be written by the application to re-enable CAN transfers. Note: It's recommended to use either the ports CAN_CLKSTOP_REQ and CAN_CLKSTOP_ACK or the CCTRL.ClkStReq and CFR.ClkStAck. The application CFR.ClkStReq showsalso the actual status of the portCAN_CLKSTOP_REQ.

CRR on page 25-30
Core Release Register

HWS on page 25-30
Hardware Configuration Status Register

CCTRL

Control Register

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00000
can1	0xFFC01000	0xFFC01000

Offset: 0x0

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												DE2 RW 0x0	DE1 RW 0x0	MIL RW 0x0	Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Test RW 0x0	CCE RW 0x0	DAR RW 0x0	Reserved	EIE RW 0x0	SIE RW 0x0	ILE RW 0x0	Init RW 0x1

CCTRL Fields

Bit	Name	Description	Access	Reset
19	DE2	DMA Enable for IF2 **Value** **Description** 0x0 Module DMA output port CAN_IF2DMA is always LOW. 0x1 Requesting a message object transfer from IF2 to Message RAM or vice versa with IF2CMR.DMAactive enabled the end of the transfer will be marked with setting port CAN_IF2DMA to one. The port remains one until first access to one of the IF2 registers.	RW	0x0
18	DE1	DMA Enable for IF1 **Value** **Description** 0x0 Module DMA output port CAN_IF1DMA is always LOW. 0x1 Requesting a message object transfer from IF1 to Message RAM or vice versa with IF1CMR.DMAactive enabled the end of the transfer will be marked with setting port CAN_IF1DMA to one. The port remains one until first access to one of the IF1 registers.	RW	0x0

Bit	Name	Description	Access	Reset
17	`MIL`	Message Object Interrupt Line Enable **Value** **Description** 0x0 Message Object Interrupt CAN_INT_MO is always LOW. If CCTRL.ILE is enabled all message object interrupts are routed to line CAN_INT_STATUS otherwise no message object interrupt will be visible. 0x1 Message object interrupts will set CAN_INT_MO to one, signal remains one until all pending interrupts are processed.	`RW`	`0x0`
7	`Test`	Test Mode Enable **Value** **Description** 0x0 Normal Operation. 0x1 Test Mode. Enables the write access to Test Register CTR.	`RW`	`0x0`
6	`CCE`	Configuration Change Enable **Value** **Description** 0x0 The CPU has no write access to the configuration registers. 0x1 The CPU has write access to the Bit Timing Register CBT (while CCTRL.Init = 1).	`RW`	`0x0`
5	`DAR`	Disable Automatic Retransmission **Value** **Description** 0x0 Automatic Retransmission of not successful messages enabled. 0x1 Automatic Retransmission disabled.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
3	EIE	Error Interrupt Enable **Value** — **Description** 0x0 — CSTS.PER, CSTS.BOff and CSTS.EWarn flags will still be updated, but without affecting interrupt line CAN_INT_STATUS and Interrupt register CIR 0x1 — If CSTS.PER flag is one, or CSTS.BOff or CSTS.EWarn are changed, the interrupt line CAN_INT_STATUS gets active (if ILE=1) and CIR.StatusInt is set.	RW	0x0
2	SIE	Status Interrupt Enable **Value** — **Description** 0x0 — CSTS.RxOk, CSTS.TxOk and CSTS.LEC will still be updated, but without affecting interrupt line CAN_INT_STATUS and Interrupt register CIR. 0x1 — When a message transfer is successfully completed or a CAN bus error is detected, indicated by flags CSTS.RxOk, CSTS.TxOk and CSTS.LEC, the interrupt line CAN_INT_STATUS gets active (if ILE=1) and CIR.StatusInt is set.	RW	0x0
1	ILE	Module Interrupt Line Enable **Value** — **Description** 0x0 — Module Interrupt Line CAN_INT_STATUS is always LOW. 0x1 — Error and status interrupts (if CCTRL.EIE=1 and CCTRL.SIE=1) will set line CAN_INT_STATUS to one, signal remains one until all pending interrupts are processed. If MIL is disabled, the message object interrupts will also affect this interrupt line.	RW	0x0

Bit	Name	Description	Access	Reset
0	`Init`	Initialization Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to CCTRL.Init can be read back. Therefore the programmer has to assure that the previous value written to CCTRL.Init has been accepted by reading CCTRL.Init before setting CCTRL.Init to a new value.\n Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting CCTRL.Init. If the device goes Bus_Off, it will set CCTRL.Init of its own accord, stopping all bus activities. Once CCTRL.Init has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCTRL.Init, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the us_Off recovery sequence. **Value** — **Description** 0x0 — Normal Operation. 0x1 — Initialization is started.	`RW`	`0x1`

CSTS

Status Register

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00004`
`can1`	`0xFFC01000`	`0xFFC01004`

Offset: `0x4`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							PER RO 0x0	BOff RO 0x0	EWarn RO 0x0	EPASS RO 0x0	RxOK RO 0x0	TxOK RO 0x0	LEC RO 0x7		

CSTS Fields

Bit	Name	Description	Access	Reset
8	PER	Parity Error Detected **Value** **Description** 0x0 No parity error detected since last read access. 0x1 The Parity CheckMechanism has detected a parity error in the Message RAM, this bit will be reset if Status Register is read	RO	0x0
7	BOff	Bus_Off Status **Value** **Description** 0x0 The CAN module is not Bus_Off. 0x1 The CAN module is in Bus_Off state.	RO	0x0
6	EWarn	Warning Status **Value** **Description** 0x0 Both error counters are below the error warning limit of 96. 0x1 At least one of the error counters in the EML has reached the error warning limit of 96.	RO	0x0
5	EPASS	Error Passive **Value** **Description** 0x0 The CAN Core is in the error active state. It normally takes part in bus communication and sends an active error flag when an error has been detected. 0x1 The CAN Core is in the error passive state as defined in the CAN Specification.	RO	0x0

Bit	Name	Description	Access	Reset
4	`RxOK`	Received a Message Successfully **Value** **Description** 0x0 Since this bit was read by the CPU, no message has been successfully received. This bit is never reset by CAN internal events. 0x1 Since this bit was last reset by a read access of the CPU, a message has been successfully received (independently of the result of acceptance filtering). This bit will be reset by reading the Status Register.	`RO`	`0x0`
3	`TxOK`	Transmitted a Message Successfully **Value** **Description** 0x0 Since this bit was last read by the CPU, no message has been successfully transmitted. This bit is never reset by CAN internal events. 0x1 Since this bit was last reset by a read access of the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted. This bit will be reset by reading the Status Register.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
2:0	LEC	The LEC field holds a code which indicates the type of the last error to occur on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error. **Value** — **Description** 0x0 — Set together with CSTS.RxOK or CSTS.TxOK. 0x1 — More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 0x2 — A fixed format part of a received frame has the wrong format. 0x3 — The message this CAN Core transmitted was not acknowledged by another node. 0x4 — During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant. 0x5 — During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 0x6 — The CRC checksum was incorrect in the message received, the CRC received for an incoming message does not match with the calculated CRC for the received data. 0x7 — Any read access to the Status Register re initializes the LEC to 7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the Status Register.	RO	0x7

CERC

Error Counter Register

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00008
can1	0xFFC01000	0xFFC01008

Offset: 0x8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP RO 0x0	REC RO 0x0							TEC RO 0x0							

CERC Fields

Bit	Name	Description	Access	Reset
15	RP	**Value** **Description** 0x0 The Receive Error Counter is below the error passive level. 0x1 The Receive Error Counter has reached the error passive level as defined in the CAN Specification.	RO	0x0
14:8	REC	Actual state of the Receive Error Counter. Values between 0 and 127.	RO	0x0
7:0	TEC	Actual state of the Transmit Error Counter. Values between 0 and 255.	RO	0x0

CBT

This register is only writable if bits CCTRL.CCE and CCTRL.Init are set. The CAN bit time may be programed in the range of [4 .. 25] time quanta. The CAN time quantum may be programmed in the range of [1 .. 1024] CAN_CLK periods. For details see Application Note 001 "Configuration of Bit Timing". The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. TSeg1 is the sum of Prop_Seg and Phase_Seg1. TSeg2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [TSeg1 + TSeg2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC0000C
can1	0xFFC01000	0xFFC0100C

Offset: 0xC

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												BRPE RW 0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TSeg2 RW 0x2			TSeg1 RW 0x3				SJW RW 0x0		BRP RW 0x1					

CBT Fields

Bit	Name	Description	Access	Reset
19:16	BRPE	By programming BRPE the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BRP (LSBs) is used.	RW	0x0
14:12	TSeg2	Valid values for TSeg2 are [0 .. 7].	RW	0x2
11:8	TSeg1	Valid values for TSeg1 are [1 .. 15].	RW	0x3
7:6	SJW	Valid programmed values are [0 .. 3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.	RW	0x0
5:0	BRP	The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are [0 .. 63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.	RW	0x1

CIR

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If IntID is different from 0x00 and CCTRL.MIL is set, the interrupt port CAN_INT_MO is active. The interrupt port remains active until IntID is back to value 0x00 (the cause of the interrupt is reset) or until CCTRL.MIL is reset. If CCTRL.ILE is set and CCTRL.MIL is reseted the Message Object interrupts will be routed to interrupt port CAN_INT_STATUS. The interrupt port remains active until IntID is back to value 0x00 (the cause of the interrupt is reset) or until CCTRL.MIL is set or CCTRL.ILE is reset. The Message Object's interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the Message Object's IntPnd bit.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00010
can1	0xFFC01000	0xFFC01010

Offset: 0x10

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
StatusInt RO 0x0	Reserved							IntId RO 0x0							

CIR Fields

Bit	Name	Description	Access	Reset
15	StatusInt	The Status Interrupt is cleared by reading the Status Register.	RO	0x0
7:0	IntId	0x00 No Message Object interrupt is pending. 0x01-0x80 Number of Message Object which caused the interrupt. 0x81-0xFF unused.	RO	0x0

CTR

The Test Mode is entered by setting bit CCTRL.Test to one. In Test Mode the bits EXL, Tx1, Tx0, LBack and Silent in the Test Register are writable. Bit Rx monitors the state of pin CAN_RXD and therefore is only readable. All Test Register functions are disabled when bit Test is reset to zero. Loop Back Mode and CAN_TXD Control Mode are hardware test modes, not to be used by application programs. Note: This register is only writable if bit CCTRL.Test is set.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00014
can1	0xFFC01000	0xFFC01014

Offset: 0x14

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Rx RO 0x0	Tx RW 0x0		LBack RW 0x0	Silent RW 0x0	Reserved		

CTR Fields

Bit	Name	Description	Access	Reset
7	Rx	Monitors the actual value of the CAN_RXD pin. **Value** **Description** 0x0 The CAN bus is dominant (CAN_RXD = 0). 0x1 The CAN bus is recessive (CAN_RXD = 1).	RO	0x0
6:5	Tx	Controls CAN_TXD pin. Setting to non-zero disturbs message transfer. **Value** **Description** 0x0 Reset value, CAN_TXD is controlled by the CAN_Core. 0x1 Sample Point can be monitored at CAN_TXD pin. 0x2 CAN_TXD pin drives a dominant (0) value. 0x3 CAN_TXD pin drives a recessive (1) value.	RW	0x0
4	LBack	Loop Back Mode **Value** **Description** 0x0 Loop Back Mode is disabled. 0x1 Loop Back Mode is enabled.	RW	0x0
3	Silent	Silent Mode **Value** **Description** 0x0 Normal operation. 0x1 The CAN is in Silent Mode.	RW	0x0

CFR

The Function Register controls the features RAM_Initialisation and Power_Down also by application register. The CAN module can be prepared for Power_Down by setting the port CAN_CLKSTOP_REQ to one or writing to CFR.ClkStReq a one. The power down state is left by setting port CAN_CLKSTOP_REQ to zero or writing to CFR.ClkStReq a zero, acknowledged by CAN_CLKSTOP_ACK is going to zero as well as CFR.ClkStAck. The CCTRL.Init bit is left one and has to be written by the application to re-enable CAN transfers. Note: It's recommended to use either the ports CAN_CLKSTOP_REQ and CAN_CLKSTOP_ACK or the CCTRL.ClkStReq and CFR.ClkStAck. The application CFR.ClkStReq showsalso the actual status of the portCAN_CLKSTOP_REQ.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00018`
`can1`	`0xFFC01000`	`0xFFC01018`

Offset: `0x18`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RAMinit RW 0x0	Reserved	ClkStReq RW 0x0	ClkStAck RO 0x0

CFR Fields

Bit	Name	Description	Access	Reset
3	`RAMinit`	Request for automatic RAM Initialization **Value** — **Description** 0x0 — No automatic RAM Initialization is requested, if once a ram initialization is started a write of a zero will be ignored. The Bit is cleared by hardware, after RAM Initialization is completed. 0x1 — Start automatic RAM Initialization. All message objects will be written with zeros and the parity bits will be set. The RAMInit Bit will return to zero after the RAM-Initialization process is completed. A RAM Initialization Request is only possible if CCTRL.Init is set. The duration of the automatic RAM Initialization is messagebuffer-size + 4 host_clock cycles.	RW	0x0

Bit	Name	Description	Access	Reset
1	ClkStReq	Clock Stop Request	RW	0x0
0	ClkStAck	Clock Stop Acknowledgement	RO	0x0

CRR

Core Release Register

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00020
can1	0xFFC01000	0xFFC01020

Offset: 0x20

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL RO 0x1				STEP RO 0x11								YEAR RO 0x6			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON RO 0x11								DAY RO 0x28							

CRR Fields

Bit	Name	Description	Access	Reset
31:28	REL		RO	0x1
27:20	STEP		RO	0x11
19:16	YEAR		RO	0x6
15:8	MON		RO	0x11
7:0	DAY		RO	0x28

HWS

Hardware Configuration Status Register

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00024
can1	0xFFC01000	0xFFC01024

Offset: 0x24

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													paren RO 0x0	mb_w RO 0x3	

HWS Fields

Bit	Name	Description	Access	Reset
2	paren	Parity Generation **Value** **Description** 0x0 Parity generation harware is not present. 0x1 Parity generation harware is present.	RO	0x0
1:0	mb_w	Message Buffer Count **Value** **Description** 0x0 16 Message Objects 0x1 32 Message Objects 0x2 64 Message Objects 0x3 128 Message Objects	RO	0x3

Message Handler Group Register Descriptions

These registers are related to the operation of the Message Handler. The Message Handler is a state machine that controls the data transfer between the single ported Message RAM and the CAN Core's Rx/Tx Shift Register. It also handles acceptance filtering and the interrupt setting as programmed in the Control and Configuration Registers.

Offset: 0x84

MOTRX on page 25-33
Reading this register allows the CPU to quickly detect if any of the transmission request bits in each of the MOTRA, MOTRB, MOTRC, and MOTRD Transmission Request Registers are set.

MOTRA on page 25-39
Transmission request bits for Message Objects 1 to 32. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

MOTRB on page 25-48
Transmission request bits for Message Objects 33 to 64. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

MOTRC on page 25-56
Transmission request bits for Message Objects 65 to 96. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

MOTRD on page 25-65
Transmission request bits for Message Objects 97 to 128. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

MONDX on page 25-74
Reading this register allows the CPU to quickly detect if any of the new data bits in each of the MONDA, MONDB, MONDC, and MONDD New Data Registers are set.

MONDA on page 25-79
New data bits for Message Objects 1 to 32. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

MONDB on page 25-90
New data bits for Message Objects 33 to 64. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

MONDC on page 25-101
New data bits for Message Objects 65 to 96. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

MONDD on page 25-112
New data bits for Message Objects 97 to 128. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

MOIPX on page 25-124
Reading this register allows the CPU to quickly detect if any of the interrupt pending bits in each of the MOIPA, MOIPB, MOIPC, and MOIPD Interrupt Pending Registers are set.

MOIPA on page 25-130
Interrupt pending bits for Message Objects 1 to 32. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

MOIPB on page 25-138
Interrupt pending bits for Message Objects 33 to 64. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

MOIPC on page 25-147
Interrupt pending bits for Message Objects 65 to 96. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

MOIPD on page 25-156
Interrupt pending bits for Message Objects 97 to 128. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

MOVALX on page 25-164
Reading this register allows the CPU to quickly detect if any of the message valid bits in each of the MOVALA, MOVALB, MOVALC, and MOVALD Message Valid Registers are set.

MOVALA on page 25-170
Message valid bits for Message Objects 1 to 32. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

MOVALB on page 25-179
Message valid bits for Message Objects 33 to 64. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

MOVALC on page 25-187
Message valid bits for Message Objects 65 to 96. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

MOVALD on page 25-196
Message valid bits for Message Objects 97 to 128. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

MOTRX

Reading this register allows the CPU to quickly detect if any of the transmission request bits in each of the MOTRA, MOTRB, MOTRC, and MOTRD Transmission Request Registers are set.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00084
can1	0xFFC01000	0xFFC01084

Offset: 0x84

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqstD_3 RO 0x0	TxRqstD_2 RO 0x0	TxRqstD_1 RO 0x0	TxRqstD_0 RO 0x0	TxRqstC_3 RO 0x0	TxRqstC_2 RO 0x0	TxRqstC_1 RO 0x0	TxRqstC_0 RO 0x0	TxRqstB_3 RO 0x0	TxRqstB_2 RO 0x0	TxRqstB_1 RO 0x0	TxRqstB_0 RO 0x0	TxRqstA_3 RO 0x0	TxRqstA_2 RO 0x0	TxRqstA_1 RO 0x0	TxRqstA_0 RO 0x0

MOTRX Fields

Bit	Name	Description	Access	Reset
15	TxRqstD_3	Each bit in this field is a logical OR of a byte of the MOTRD register. Array index i corresponds to byte i of the MOTRD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRD are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRD are requested and are not yet done.	RO	0x0
14	TxRqstD_2	Each bit in this field is a logical OR of a byte of the MOTRD register. Array index i corresponds to byte i of the MOTRD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRD are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRD are requested and are not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
13	`TxRqstD_1`	Each bit in this field is a logical OR of a byte of the MOTRD register. Array index i corresponds to byte i of the MOTRD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRD are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRD are requested and are not yet done.	RO	0x0
12	`TxRqstD_0`	Each bit in this field is a logical OR of a byte of the MOTRD register. Array index i corresponds to byte i of the MOTRD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRD are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRD are requested and are not yet done.	RO	0x0
11	`TxRqstC_3`	Each bit in this field is a logical OR of a byte of the MOTRC register. Array index i corresponds to byte i of the MOTRC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRC are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRC are requested and are not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
10	`TxRqstC_2`	Each bit in this field is a logical OR of a byte of the MOTRC register. Array index i corresponds to byte i of the MOTRC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRC are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRC are requested and are not yet done.	RO	0x0
9	`TxRqstC_1`	Each bit in this field is a logical OR of a byte of the MOTRC register. Array index i corresponds to byte i of the MOTRC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRC are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRC are requested and are not yet done.	RO	0x0
8	`TxRqstC_0`	Each bit in this field is a logical OR of a byte of the MOTRC register. Array index i corresponds to byte i of the MOTRC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRC are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRC are requested and are not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
7	`TxRqstB_3`	Each bit in this field is a logical OR of a byte of the MOTRB register. Array index i corresponds to byte i of the MOTRB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRB are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRB are requested and are not yet done.	RO	0x0
6	`TxRqstB_2`	Each bit in this field is a logical OR of a byte of the MOTRB register. Array index i corresponds to byte i of the MOTRB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRB are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRB are requested and are not yet done.	RO	0x0
5	`TxRqstB_1`	Each bit in this field is a logical OR of a byte of the MOTRB register. Array index i corresponds to byte i of the MOTRB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRB are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRB are requested and are not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
4	TxRqstB_0	Each bit in this field is a logical OR of a byte of the MOTRB register. Array index i corresponds to byte i of the MOTRB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRB are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRB are requested and are not yet done.	RO	0x0
3	TxRqstA_3	Each bit in this field is a logical OR of a byte of the MOTRA register. Array index i corresponds to byte i of the MOTRA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRA are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRA are requested and are not yet done.	RO	0x0
2	TxRqstA_2	Each bit in this field is a logical OR of a byte of the MOTRA register. Array index i corresponds to byte i of the MOTRA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRA are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRA are requested and are not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
1	TxRqstA_1	Each bit in this field is a logical OR of a byte of the MOTRA register. Array index i corresponds to byte i of the MOTRA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRA are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRA are requested and are not yet done.	RO	0x0
0	TxRqstA_0	Each bit in this field is a logical OR of a byte of the MOTRA register. Array index i corresponds to byte i of the MOTRA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOTRA are not waiting for transmission. 0x1 One or more of the transmission of the Message Objects in the corresponding byte of MOTRA are requested and are not yet done.	RO	0x0

MOTRA

Transmission request bits for Message Objects 1 to 32. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00088
can1	0xFFC01000	0xFFC01088

Offset: 0x88

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TxRqst_31 RO 0x0	TxRqst_30 RO 0x0	TxRqst_29 RO 0x0	TxRqst_28 RO 0x0	TxRqst_27 RO 0x0	TxRqst_26 RO 0x0	TxRqst_25 RO 0x0	TxRqst_24 RO 0x0	TxRqst_23 RO 0x0	TxRqst_22 RO 0x0	TxRqst_21 RO 0x0	TxRqst_20 RO 0x0	TxRqst_19 RO 0x0	TxRqst_18 RO 0x0	TxRqst_17 RO 0x0	TxRqst_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqst_15 RO 0x0	TxRqst_14 RO 0x0	TxRqst_13 RO 0x0	TxRqst_12 RO 0x0	TxRqst_11 RO 0x0	TxRqst_10 RO 0x0	TxRqst_9 RO 0x0	TxRqst_8 RO 0x0	TxRqst_7 RO 0x0	TxRqst_6 RO 0x0	TxRqst_5 RO 0x0	TxRqst_4 RO 0x0	TxRqst_3 RO 0x0	TxRqst_2 RO 0x0	TxRqst_1 RO 0x0	TxRqst_0 RO 0x0

MOTRA Fields

Bit	Name	Description	Access	Reset
31	`TxRqst_31`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
30	`TxRqst_30`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
29	`TxRqst_29`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
28	`TxRqst_28`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
27	`TxRqst_27`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
26	`TxRqst_26`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
25	`TxRqst_25`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
24	TxRqst_24	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
23	TxRqst_23	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
22	TxRqst_22	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
21	TxRqst_21	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
20	TxRqst_20	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
19	TxRqst_19	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
18	TxRqst_18	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
17	TxRqst_17	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
16	TxRqst_16	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
15	TxRqst_15	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
14	TxRqst_14	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
13	TxRqst_13	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
12	`TxRqst_12`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
11	`TxRqst_11`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
10	`TxRqst_10`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
9	`TxRqst_9`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
8	`TxRqst_8`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
7	`TxRqst_7`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
6	`TxRqst_6`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
5	`TxRqst_5`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
4	`TxRqst_4`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
3	`TxRqst_3`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
2	`TxRqst_2`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
1	`TxRqst_1`	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
0	TxRqst_0	Transmission request bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

MOTRB

Transmission request bits for Message Objects 33 to 64. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC0008C
can1	0xFFC01000	0xFFC0108C

Offset: 0x8C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TxRqst_31 RO 0x0	TxRqst_30 RO 0x0	TxRqst_29 RO 0x0	TxRqst_28 RO 0x0	TxRqst_27 RO 0x0	TxRqst_26 RO 0x0	TxRqst_25 RO 0x0	TxRqst_24 RO 0x0	TxRqst_23 RO 0x0	TxRqst_22 RO 0x0	TxRqst_21 RO 0x0	TxRqst_20 RO 0x0	TxRqst_19 RO 0x0	TxRqst_18 RO 0x0	TxRqst_17 RO 0x0	TxRqst_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqst_15 RO 0x0	TxRqst_14 RO 0x0	TxRqst_13 RO 0x0	TxRqst_12 RO 0x0	TxRqst_11 RO 0x0	TxRqst_10 RO 0x0	TxRqst_9 RO 0x0	TxRqst_8 RO 0x0	TxRqst_7 RO 0x0	TxRqst_6 RO 0x0	TxRqst_5 RO 0x0	TxRqst_4 RO 0x0	TxRqst_3 RO 0x0	TxRqst_2 RO 0x0	TxRqst_1 RO 0x0	TxRqst_0 RO 0x0

MOTRB Fields

Bit	Name	Description	Access	Reset
31	`TxRqst_31`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
30	`TxRqst_30`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
29	`TxRqst_29`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
28	`TxRqst_28`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
27	`TxRqst_27`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
26	`TxRqst_26`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
25	`TxRqst_25`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
24	`TxRqst_24`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
23	`TxRqst_23`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
22	`TxRqst_22`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
21	`TxRqst_21`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
20	`TxRqst_20`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
19	TxRqst_19	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
18	TxRqst_18	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
17	TxRqst_17	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
16	TxRqst_16	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
15	TxRqst_15	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
14	TxRqst_14	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
13	TxRqst_13	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
12	TxRqst_12	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
11	`TxRqst_11`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
10	`TxRqst_10`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
9	`TxRqst_9`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
8	`TxRqst_8`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
7	`TxRqst_7`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
6	`TxRqst_6`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
5	`TxRqst_5`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
4	`TxRqst_4`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
3	`TxRqst_3`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
2	`TxRqst_2`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
1	`TxRqst_1`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
0	`TxRqst_0`	Transmission request bits for Message Objects 33 to 64. Array index i corresponds to Message Object i +33. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0

MOTRC

Transmission request bits for Message Objects 65 to 96. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00090
can1	0xFFC01000	0xFFC01090

Offset: 0x90

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TxRqst_31 RO 0x0	TxRqst_30 RO 0x0	TxRqst_29 RO 0x0	TxRqst_28 RO 0x0	TxRqst_27 RO 0x0	TxRqst_26 RO 0x0	TxRqst_25 RO 0x0	TxRqst_24 RO 0x0	TxRqst_23 RO 0x0	TxRqst_22 RO 0x0	TxRqst_21 RO 0x0	TxRqst_20 RO 0x0	TxRqst_19 RO 0x0	TxRqst_18 RO 0x0	TxRqst_17 RO 0x0	TxRqst_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqst_15 RO 0x0	TxRqst_14 RO 0x0	TxRqst_13 RO 0x0	TxRqst_12 RO 0x0	TxRqst_11 RO 0x0	TxRqst_10 RO 0x0	TxRqst_9 RO 0x0	TxRqst_8 RO 0x0	TxRqst_7 RO 0x0	TxRqst_6 RO 0x0	TxRqst_5 RO 0x0	TxRqst_4 RO 0x0	TxRqst_3 RO 0x0	TxRqst_2 RO 0x0	TxRqst_1 RO 0x0	TxRqst_0 RO 0x0

MOTRC Fields

Bit	Name	Description	Access	Reset
31	TxRqst_31	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
30	`TxRqst_30`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
29	`TxRqst_29`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
28	`TxRqst_28`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
27	`TxRqst_27`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
26	`TxRqst_26`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
25	`TxRqst_25`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
24	`TxRqst_24`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
23	`TxRqst_23`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
22	TxRqst_22	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
21	TxRqst_21	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
20	TxRqst_20	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
19	TxRqst_19	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
18	TxRqst_18	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
17	TxRqst_17	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
16	TxRqst_16	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
15	TxRqst_15	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
14	`TxRqst_14`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
13	`TxRqst_13`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
12	`TxRqst_12`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
11	`TxRqst_11`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
10	`TxRqst_10`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
9	`TxRqst_9`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
8	`TxRqst_8`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
7	`TxRqst_7`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
6	TxRqst_6	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
5	TxRqst_5	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
4	TxRqst_4	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
3	TxRqst_3	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
2	`TxRqst_2`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	`RO`	`0x0`
1	`TxRqst_1`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	`RO`	`0x0`
0	`TxRqst_0`	Transmission request bits for Message Objects 65 to 96. Array index i corresponds to Message Object i +65. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	`RO`	`0x0`

MOTRD

Transmission request bits for Message Objects 97 to 128. By reading the TxRqst bits, the CPU can check for which Message Object a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Remote Frame or reset by the Message Handler after a successful transmission.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00094`
`can1`	`0xFFC01000`	`0xFFC01094`

Offset: `0x94`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TxRqst_31 RO 0x0	TxRqst_30 RO 0x0	TxRqst_29 RO 0x0	TxRqst_28 RO 0x0	TxRqst_27 RO 0x0	TxRqst_26 RO 0x0	TxRqst_25 RO 0x0	TxRqst_24 RO 0x0	TxRqst_23 RO 0x0	TxRqst_22 RO 0x0	TxRqst_21 RO 0x0	TxRqst_20 RO 0x0	TxRqst_19 RO 0x0	TxRqst_18 RO 0x0	TxRqst_17 RO 0x0	TxRqst_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxRqst_15 RO 0x0	TxRqst_14 RO 0x0	TxRqst_13 RO 0x0	TxRqst_12 RO 0x0	TxRqst_11 RO 0x0	TxRqst_10 RO 0x0	TxRqst_9 RO 0x0	TxRqst_8 RO 0x0	TxRqst_7 RO 0x0	TxRqst_6 RO 0x0	TxRqst_5 RO 0x0	TxRqst_4 RO 0x0	TxRqst_3 RO 0x0	TxRqst_2 RO 0x0	TxRqst_1 RO 0x0	TxRqst_0 RO 0x0

MOTRD Fields

Bit	Name	Description	Access	Reset
31	TxRqst_31	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0
30	TxRqst_30	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
29	`TxRqst_29`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
28	`TxRqst_28`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
27	`TxRqst_27`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
26	`TxRqst_26`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
25	TxRqst_25	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
24	TxRqst_24	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
23	TxRqst_23	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
22	TxRqst_22	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
21	`TxRqst_21`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
20	`TxRqst_20`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
19	`TxRqst_19`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
18	`TxRqst_18`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
17	TxRqst_17	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
16	TxRqst_16	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
15	TxRqst_15	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
14	TxRqst_14	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
13	`TxRqst_13`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
12	`TxRqst_12`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
11	`TxRqst_11`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
10	`TxRqst_10`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
9	TxRqst_9	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
8	TxRqst_8	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
7	TxRqst_7	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
6	TxRqst_6	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
5	`TxRqst_5`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
4	`TxRqst_4`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
3	`TxRqst_3`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
2	`TxRqst_2`	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

Bit	Name	Description	Access	Reset
1	TxRqst_1	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0
0	TxRqst_0	Transmission request bits for Message Objects 97 to 128. Array index i corresponds to Message Object i +97. **Value** / **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	RO	0x0

MONDX

Reading this register allows the CPU to quickly detect if any of the new data bits in each of the MONDA, MONDB, MONDC, and MONDD New Data Registers are set.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00098
can1	0xFFC01000	0xFFC01098

Offset: 0x98

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDatD_3 RO 0x0	NewDatD_2 RO 0x0	NewDatD_1 RO 0x0	NewDatD_0 RO 0x0	NewDatC_3 RO 0x0	NewDatC_2 RO 0x0	NewDatC_1 RO 0x0	NewDatC_0 RO 0x0	NewDatB_3 RO 0x0	NewDatB_2 RO 0x0	NewDatB_1 RO 0x0	NewDatB_0 RO 0x0	NewDatA_3 RO 0x0	NewDatA_2 RO 0x0	NewDatA_1 RO 0x0	NewDatA_0 RO 0x0

MONDX Fields

Bit	Name	Description	Access	Reset
15	NewDatD_3	Each bit in this field is a logical OR of a byte of the MONDD register. Array index i corresponds to byte i of the MONDD register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MONDD don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDD have new data available.	RO	0x0
14	NewDatD_2	Each bit in this field is a logical OR of a byte of the MONDD register. Array index i corresponds to byte i of the MONDD register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MONDD don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDD have new data available.	RO	0x0
13	NewDatD_1	Each bit in this field is a logical OR of a byte of the MONDD register. Array index i corresponds to byte i of the MONDD register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MONDD don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDD have new data available.	RO	0x0
12	NewDatD_0	Each bit in this field is a logical OR of a byte of the MONDD register. Array index i corresponds to byte i of the MONDD register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MONDD don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDD have new data available.	RO	0x0

Bit	Name	Description	Access	Reset
11	`NewDatC_3`	Each bit in this field is a logical OR of a byte of the MONDC register. Array index i corresponds to byte i of the MONDC register. **Value** — **Description** 0x0 — The Message Objects in the corresponding byte of MONDC don't have new data. 0x1 — One or more of the Message Objects in the corresponding byte of MONDC have new data available.	RO	0x0
10	`NewDatC_2`	Each bit in this field is a logical OR of a byte of the MONDC register. Array index i corresponds to byte i of the MONDC register. **Value** — **Description** 0x0 — The Message Objects in the corresponding byte of MONDC don't have new data. 0x1 — One or more of the Message Objects in the corresponding byte of MONDC have new data available.	RO	0x0
9	`NewDatC_1`	Each bit in this field is a logical OR of a byte of the MONDC register. Array index i corresponds to byte i of the MONDC register. **Value** — **Description** 0x0 — The Message Objects in the corresponding byte of MONDC don't have new data. 0x1 — One or more of the Message Objects in the corresponding byte of MONDC have new data available.	RO	0x0
8	`NewDatC_0`	Each bit in this field is a logical OR of a byte of the MONDC register. Array index i corresponds to byte i of the MONDC register. **Value** — **Description** 0x0 — The Message Objects in the corresponding byte of MONDC don't have new data. 0x1 — One or more of the Message Objects in the corresponding byte of MONDC have new data available.	RO	0x0

Bit	Name	Description	Access	Reset
7	`NewDatB_3`	Each bit in this field is a logical OR of a byte of the MONDB register. Array index i corresponds to byte i of the MONDB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDB don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDB have new data available.	`RO`	`0x0`
6	`NewDatB_2`	Each bit in this field is a logical OR of a byte of the MONDB register. Array index i corresponds to byte i of the MONDB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDB don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDB have new data available.	`RO`	`0x0`
5	`NewDatB_1`	Each bit in this field is a logical OR of a byte of the MONDB register. Array index i corresponds to byte i of the MONDB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDB don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDB have new data available.	`RO`	`0x0`
4	`NewDatB_0`	Each bit in this field is a logical OR of a byte of the MONDB register. Array index i corresponds to byte i of the MONDB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDB don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDB have new data available.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
3	`NewDatA_3`	Each bit in this field is a logical OR of a byte of the MONDA register. Array index i corresponds to byte i of the MONDA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDA don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDA have new data available.	RO	0x0
2	`NewDatA_2`	Each bit in this field is a logical OR of a byte of the MONDA register. Array index i corresponds to byte i of the MONDA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDA don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDA have new data available.	RO	0x0
1	`NewDatA_1`	Each bit in this field is a logical OR of a byte of the MONDA register. Array index i corresponds to byte i of the MONDA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDA don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDA have new data available.	RO	0x0
0	`NewDatA_0`	Each bit in this field is a logical OR of a byte of the MONDA register. Array index i corresponds to byte i of the MONDA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MONDA don't have new data. 0x1 One or more of the Message Objects in the corresponding byte of MONDA have new data available.	RO	0x0

MONDA

New data bits for Message Objects 1 to 32. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC0009C
can1	0xFFC01000	0xFFC0109C

Offset: 0x9C

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NewDat_31 RO 0x0	NewDat_30 RO 0x0	NewDat_29 RO 0x0	NewDat_28 RO 0x0	NewDat_27 RO 0x0	NewDat_26 RO 0x0	NewDat_25 RO 0x0	NewDat_24 RO 0x0	NewDat_23 RO 0x0	NewDat_22 RO 0x0	NewDat_21 RO 0x0	NewDat_20 RO 0x0	NewDat_19 RO 0x0	NewDat_18 RO 0x0	NewDat_17 RO 0x0	NewDat_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat_15 RO 0x0	NewDat_14 RO 0x0	NewDat_13 RO 0x0	NewDat_12 RO 0x0	NewDat_11 RO 0x0	NewDat_10 RO 0x0	NewDat_9 RO 0x0	NewDat_8 RO 0x0	NewDat_7 RO 0x0	NewDat_6 RO 0x0	NewDat_5 RO 0x0	NewDat_4 RO 0x0	NewDat_3 RO 0x0	NewDat_2 RO 0x0	NewDat_1 RO 0x0	NewDat_0 RO 0x0

MONDA Fields

Bit	Name	Description	Access	Reset
31	NewDat_31	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
30	`NewDat_30`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
29	`NewDat_29`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
28	`NewDat_28`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
27	`NewDat_27`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
26	`NewDat_26`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
25	`NewDat_25`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
24	NewDat_24	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
23	NewDat_23	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
22	NewDat_22	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
21	`NewDat_21`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
20	`NewDat_20`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
19	`NewDat_19`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
18	NewDat_18	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
17	NewDat_17	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
16	NewDat_16	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
15	`NewDat_15`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
14	`NewDat_14`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
13	`NewDat_13`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
12	`NewDat_12`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
11	`NewDat_11`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
10	`NewDat_10`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
9	NewDat_9	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
8	NewDat_8	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
7	NewDat_7	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
6	`NewDat_6`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
5	`NewDat_5`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
4	`NewDat_4`	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
3	NewDat_3	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
2	NewDat_2	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
1	NewDat_1	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
0	NewDat_0	New data bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

MONDB

New data bits for Message Objects 33 to 64. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000A0
can1	0xFFC01000	0xFFC010A0

Offset: 0xA0

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NewDat_31 RO 0x0	NewDat_30 RO 0x0	NewDat_29 RO 0x0	NewDat_28 RO 0x0	NewDat_27 RO 0x0	NewDat_26 RO 0x0	NewDat_25 RO 0x0	NewDat_24 RO 0x0	NewDat_23 RO 0x0	NewDat_22 RO 0x0	NewDat_21 RO 0x0	NewDat_20 RO 0x0	NewDat_19 RO 0x0	NewDat_18 RO 0x0	NewDat_17 RO 0x0	NewDat_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat_15 RO 0x0	NewDat_14 RO 0x0	NewDat_13 RO 0x0	NewDat_12 RO 0x0	NewDat_11 RO 0x0	NewDat_10 RO 0x0	NewDat_9 RO 0x0	NewDat_8 RO 0x0	NewDat_7 RO 0x0	NewDat_6 RO 0x0	NewDat_5 RO 0x0	NewDat_4 RO 0x0	NewDat_3 RO 0x0	NewDat_2 RO 0x0	NewDat_1 RO 0x0	NewDat_0 RO 0x0

MONDB Fields

Bit	Name	Description	Access	Reset
31	`NewDat_31`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0: No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1: The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
30	`NewDat_30`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0: No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1: The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
29	`NewDat_29`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0: No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1: The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
28	`NewDat_28`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
27	`NewDat_27`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
26	`NewDat_26`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
25	`NewDat_25`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
24	`NewDat_24`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
23	`NewDat_23`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
22	`NewDat_22`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
21	`NewDat_21`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
20	`NewDat_20`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
19	NewDat_19	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
18	NewDat_18	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
17	NewDat_17	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
16	NewDat_16	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
15	NewDat_15	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
14	NewDat_14	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** / **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
13	NewDat_13	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
12	NewDat_12	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
11	NewDat_11	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
10	`NewDat_10`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
9	`NewDat_9`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
8	`NewDat_8`	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
7	NewDat_7	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
6	NewDat_6	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
5	NewDat_5	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
4	NewDat_4	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
3	NewDat_3	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
2	NewDat_2	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
1	NewDat_1	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
0	NewDat_0	New data bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

MONDC

New data bits for Message Objects 65 to 96. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000A4
can1	0xFFC01000	0xFFC010A4

Offset: 0xA4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NewDat_31 RO 0x0	NewDat_30 RO 0x0	NewDat_29 RO 0x0	NewDat_28 RO 0x0	NewDat_27 RO 0x0	NewDat_26 RO 0x0	NewDat_25 RO 0x0	NewDat_24 RO 0x0	NewDat_23 RO 0x0	NewDat_22 RO 0x0	NewDat_21 RO 0x0	NewDat_20 RO 0x0	NewDat_19 RO 0x0	NewDat_18 RO 0x0	NewDat_17 RO 0x0	NewDat_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat_15 RO 0x0	NewDat_14 RO 0x0	NewDat_13 RO 0x0	NewDat_12 RO 0x0	NewDat_11 RO 0x0	NewDat_10 RO 0x0	NewDat_9 RO 0x0	NewDat_8 RO 0x0	NewDat_7 RO 0x0	NewDat_6 RO 0x0	NewDat_5 RO 0x0	NewDat_4 RO 0x0	NewDat_3 RO 0x0	NewDat_2 RO 0x0	NewDat_1 RO 0x0	NewDat_0 RO 0x0

MONDC Fields

Bit	Name	Description	Access	Reset
31	NewDat_31	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
30	NewDat_30	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
29	`NewDat_29`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
28	`NewDat_28`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
27	`NewDat_27`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
26	NewDat_26	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
25	NewDat_25	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
24	NewDat_24	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
23	`NewDat_23`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
22	`NewDat_22`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
21	`NewDat_21`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
20	NewDat_20	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
19	NewDat_19	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
18	NewDat_18	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
17	NewDat_17	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
16	NewDat_16	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
15	NewDat_15	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
14	`NewDat_14`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
13	`NewDat_13`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
12	`NewDat_12`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
11	`NewDat_11`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
10	`NewDat_10`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
9	`NewDat_9`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
8	`NewDat_8`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
7	`NewDat_7`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
6	`NewDat_6`	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
5	NewDat_5	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
4	NewDat_4	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
3	NewDat_3	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
2	NewDat_2	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
1	NewDat_1	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
0	NewDat_0	New data bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

MONDD

New data bits for Message Objects 97 to 128. By reading the NewDat bits, the CPU can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception of a Data Frame or reset by the Message Handler at start of a transmission.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000A8

Module Instance	Base Address	Register Address
can1	0xFFC01000	0xFFC010A8

Offset: 0xA8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NewDat_31 RO 0x0	NewDat_30 RO 0x0	NewDat_29 RO 0x0	NewDat_28 RO 0x0	NewDat_27 RO 0x0	NewDat_26 RO 0x0	NewDat_25 RO 0x0	NewDat_24 RO 0x0	NewDat_23 RO 0x0	NewDat_22 RO 0x0	NewDat_21 RO 0x0	NewDat_20 RO 0x0	NewDat_19 RO 0x0	NewDat_18 RO 0x0	NewDat_17 RO 0x0	NewDat_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat_15 RO 0x0	NewDat_14 RO 0x0	NewDat_13 RO 0x0	NewDat_12 RO 0x0	NewDat_11 RO 0x0	NewDat_10 RO 0x0	NewDat_9 RO 0x0	NewDat_8 RO 0x0	NewDat_7 RO 0x0	NewDat_6 RO 0x0	NewDat_5 RO 0x0	NewDat_4 RO 0x0	NewDat_3 RO 0x0	NewDat_2 RO 0x0	NewDat_1 RO 0x0	NewDat_0 RO 0x0

MONDD Fields

Bit	Name	Description	Access	Reset
31	NewDat_31	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
30	`NewDat_30`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
29	`NewDat_29`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
28	`NewDat_28`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 — The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
27	NewDat_27	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
26	NewDat_26	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
25	NewDat_25	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
24	NewDat_24	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
23	NewDat_23	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
22	NewDat_22	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
21	`NewDat_21`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
20	`NewDat_20`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
19	`NewDat_19`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
18	NewDat_18	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
17	NewDat_17	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
16	NewDat_16	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
15	`NewDat_15`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
14	`NewDat_14`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
13	`NewDat_13`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
12	`NewDat_12`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
11	`NewDat_11`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
10	`NewDat_10`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
9	`NewDat_9`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
8	`NewDat_8`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`
7	`NewDat_7`	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
6	NewDat_6	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
5	NewDat_5	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
4	NewDat_4	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
3	NewDat_3	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
2	NewDat_2	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0
1	NewDat_1	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

Bit	Name	Description	Access	Reset
0	NewDat_0	New data bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RO	0x0

MOIPX

Reading this register allows the CPU to quickly detect if any of the interrupt pending bits in each of the MOIPA, MOIPB, MOIPC, and MOIPD Interrupt Pending Registers are set.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000AC
can1	0xFFC01000	0xFFC010AC

Offset: 0xAC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPndD_3 RO 0x0	IntPndD_2 RO 0x0	IntPndD_1 RO 0x0	IntPndD_0 RO 0x0	IntPndC_3 RO 0x0	IntPndC_2 RO 0x0	IntPndC_1 RO 0x0	IntPndC_0 RO 0x0	IntPndB_3 RO 0x0	IntPndB_2 RO 0x0	IntPndB_1 RO 0x0	IntPndB_0 RO 0x0	IntPndA_3 RO 0x0	IntPndA_2 RO 0x0	IntPndA_1 RO 0x0	IntPndA_0 RO 0x0

MOIPX Fields

Bit	Name	Description	Access	Reset
15	IntPndD_3	Each bit in this field is a logical OR of a byte of the MOIPD register. Array index i corresponds to byte i of the MOIPD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPD are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPD are the source an interrupt.	RO	0x0
14	IntPndD_2	Each bit in this field is a logical OR of a byte of the MOIPD register. Array index i corresponds to byte i of the MOIPD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPD are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPD are the source an interrupt.	RO	0x0
13	IntPndD_1	Each bit in this field is a logical OR of a byte of the MOIPD register. Array index i corresponds to byte i of the MOIPD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPD are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPD are the source an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
12	`IntPndD_0`	Each bit in this field is a logical OR of a byte of the MOIPD register. Array index i corresponds to byte i of the MOIPD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPD are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPD are the source an interrupt.	RO	0x0
11	`IntPndC_3`	Each bit in this field is a logical OR of a byte of the MOIPC register. Array index i corresponds to byte i of the MOIPC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPC are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPC are the source an interrupt.	RO	0x0
10	`IntPndC_2`	Each bit in this field is a logical OR of a byte of the MOIPC register. Array index i corresponds to byte i of the MOIPC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPC are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPC are the source an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
9	`IntPndC_1`	Each bit in this field is a logical OR of a byte of the MOIPC register. Array index i corresponds to byte i of the MOIPC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPC are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPC are the source an interrupt.	RO	0x0
8	`IntPndC_0`	Each bit in this field is a logical OR of a byte of the MOIPC register. Array index i corresponds to byte i of the MOIPC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPC are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPC are the source an interrupt.	RO	0x0
7	`IntPndB_3`	Each bit in this field is a logical OR of a byte of the MOIPB register. Array index i corresponds to byte i of the MOIPB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPB are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPB are the source an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
6	IntPndB_2	Each bit in this field is a logical OR of a byte of the MOIPB register. Array index i corresponds to byte i of the MOIPB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPB are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPB are the source an interrupt.	RO	0x0
5	IntPndB_1	Each bit in this field is a logical OR of a byte of the MOIPB register. Array index i corresponds to byte i of the MOIPB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPB are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPB are the source an interrupt.	RO	0x0
4	IntPndB_0	Each bit in this field is a logical OR of a byte of the MOIPB register. Array index i corresponds to byte i of the MOIPB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPB are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPB are the source an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
3	`IntPndA_3`	Each bit in this field is a logical OR of a byte of the MOIPA register. Array index i corresponds to byte i of the MOIPA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPA are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPA are the source an interrupt.	`RO`	`0x0`
2	`IntPndA_2`	Each bit in this field is a logical OR of a byte of the MOIPA register. Array index i corresponds to byte i of the MOIPA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPA are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPA are the source an interrupt.	`RO`	`0x0`
1	`IntPndA_1`	Each bit in this field is a logical OR of a byte of the MOIPA register. Array index i corresponds to byte i of the MOIPA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOIPA are not the source of an interrupt. 0x1 One or more of the Message Objects in the corresponding byte of MOIPA are the source an interrupt.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
0	IntPndA_0	Each bit in this field is a logical OR of a byte of the MOIPA register. Array index i corresponds to byte i of the MOIPA register. **Value** — **Description** 0x0 — The Message Objects in the corresponding byte of MOIPA are not the source of an interrupt. 0x1 — One or more of the Message Objects in the corresponding byte of MOIPA are the source an interrupt.	RO	0x0

MOIPA

Interrupt pending bits for Message Objects 1 to 32. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000B0
can1	0xFFC01000	0xFFC010B0

Offset: 0xB0

Access: RO

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntPnd_31 RO 0x0	IntPnd_30 RO 0x0	IntPnd_29 RO 0x0	IntPnd_28 RO 0x0	IntPnd_27 RO 0x0	IntPnd_26 RO 0x0	IntPnd_25 RO 0x0	IntPnd_24 RO 0x0	IntPnd_23 RO 0x0	IntPnd_22 RO 0x0	IntPnd_21 RO 0x0	IntPnd_20 RO 0x0	IntPnd_19 RO 0x0	IntPnd_18 RO 0x0	IntPnd_17 RO 0x0	IntPnd_16 RO 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd_15 RO 0x0	IntPnd_14 RO 0x0	IntPnd_13 RO 0x0	IntPnd_12 RO 0x0	IntPnd_11 RO 0x0	IntPnd_10 RO 0x0	IntPnd_9 RO 0x0	IntPnd_8 RO 0x0	IntPnd_7 RO 0x0	IntPnd_6 RO 0x0	IntPnd_5 RO 0x0	IntPnd_4 RO 0x0	IntPnd_3 RO 0x0	IntPnd_2 RO 0x0	IntPnd_1 RO 0x0	IntPnd_0 RO 0x0

MOIPA Fields

Bit	Name	Description	Access	Reset
31	IntPnd_31	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
30	IntPnd_30	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
29	IntPnd_29	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
28	IntPnd_28	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** / **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
27	`IntPnd_27`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
26	`IntPnd_26`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
25	`IntPnd_25`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
24	`IntPnd_24`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
23	`IntPnd_23`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
22	`IntPnd_22`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
21	`IntPnd_21`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
20	`IntPnd_20`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
19	`IntPnd_19`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
18	`IntPnd_18`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
17	`IntPnd_17`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
16	`IntPnd_16`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
15	`IntPnd_15`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
14	`IntPnd_14`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
13	`IntPnd_13`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
12	`IntPnd_12`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
11	IntPnd_11	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
10	IntPnd_10	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
9	IntPnd_9	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
8	IntPnd_8	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
7	`IntPnd_7`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
6	`IntPnd_6`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
5	`IntPnd_5`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
4	`IntPnd_4`	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
3	IntPnd_3	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
2	IntPnd_2	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
1	IntPnd_1	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
0	IntPnd_0	Interrupt pending bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

MOIPB

Interrupt pending bits for Message Objects 33 to 64. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000B4
can1	0xFFC01000	0xFFC010B4

Offset: 0xB4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntPnd_31 RO 0x0	IntPnd_30 RO 0x0	IntPnd_29 RO 0x0	IntPnd_28 RO 0x0	IntPnd_27 RO 0x0	IntPnd_26 RO 0x0	IntPnd_25 RO 0x0	IntPnd_24 RO 0x0	IntPnd_23 RO 0x0	IntPnd_22 RO 0x0	IntPnd_21 RO 0x0	IntPnd_20 RO 0x0	IntPnd_19 RO 0x0	IntPnd_18 RO 0x0	IntPnd_17 RO 0x0	IntPnd_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPnd_15 RO 0x0	IntPnd_14 RO 0x0	IntPnd_13 RO 0x0	IntPnd_12 RO 0x0	IntPnd_11 RO 0x0	IntPnd_10 RO 0x0	IntPnd_9 RO 0x0	IntPnd_8 RO 0x0	IntPnd_7 RO 0x0	IntPnd_6 RO 0x0	IntPnd_5 RO 0x0	IntPnd_4 RO 0x0	IntPnd_3 RO 0x0	IntPnd_2 RO 0x0	IntPnd_1 RO 0x0	IntPnd_0 RO 0x0

MOIPB Fields

Bit	Name	Description	Access	Reset
31	IntPnd_31	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
30	IntPnd_30	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
29	`IntPnd_29`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
28	`IntPnd_28`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
27	`IntPnd_27`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
26	`IntPnd_26`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
25	`IntPnd_25`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
24	`IntPnd_24`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
23	`IntPnd_23`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
22	`IntPnd_22`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
21	`IntPnd_21`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
20	`IntPnd_20`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
19	`IntPnd_19`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
18	`IntPnd_18`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
17	`IntPnd_17`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
16	`IntPnd_16`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
15	`IntPnd_15`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
14	`IntPnd_14`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
13	`IntPnd_13`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is not the source of an interrupt. 0x1 — The Message Object is the source of an interrupt.	RO	0x0
12	`IntPnd_12`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is not the source of an interrupt. 0x1 — The Message Object is the source of an interrupt.	RO	0x0
11	`IntPnd_11`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is not the source of an interrupt. 0x1 — The Message Object is the source of an interrupt.	RO	0x0
10	`IntPnd_10`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is not the source of an interrupt. 0x1 — The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
9	IntPnd_9	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
8	IntPnd_8	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
7	IntPnd_7	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
6	IntPnd_6	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
5	`IntPnd_5`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
4	`IntPnd_4`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
3	`IntPnd_3`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
2	`IntPnd_2`	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
1	IntPnd_1	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
0	IntPnd_0	Interrupt pending bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

MOIPC

Interrupt pending bits for Message Objects 65 to 96. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000B8
can1	0xFFC01000	0xFFC010B8

Offset: 0xB8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntPnd_31 RO 0x0	IntPnd_30 RO 0x0	IntPnd_29 RO 0x0	IntPnd_28 RO 0x0	IntPnd_27 RO 0x0	IntPnd_26 RO 0x0	IntPnd_25 RO 0x0	IntPnd_24 RO 0x0	IntPnd_23 RO 0x0	IntPnd_22 RO 0x0	IntPnd_21 RO 0x0	IntPnd_20 RO 0x0	IntPnd_19 RO 0x0	IntPnd_18 RO 0x0	IntPnd_17 RO 0x0	IntPnd_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPnd_15 RO 0x0	IntPnd_14 RO 0x0	IntPnd_13 RO 0x0	IntPnd_12 RO 0x0	IntPnd_11 RO 0x0	IntPnd_10 RO 0x0	IntPnd_9 RO 0x0	IntPnd_8 RO 0x0	IntPnd_7 RO 0x0	IntPnd_6 RO 0x0	IntPnd_5 RO 0x0	IntPnd_4 RO 0x0	IntPnd_3 RO 0x0	IntPnd_2 RO 0x0	IntPnd_1 RO 0x0	IntPnd_0 RO 0x0

MOIPC Fields

Bit	Name	Description	Access	Reset
31	IntPnd_31	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
30	IntPnd_30	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
29	IntPnd_29	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
28	`IntPnd_28`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
27	`IntPnd_27`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
26	`IntPnd_26`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
25	`IntPnd_25`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
24	`IntPnd_24`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
23	`IntPnd_23`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
22	`IntPnd_22`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
21	`IntPnd_21`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
20	`IntPnd_20`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
19	`IntPnd_19`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
18	`IntPnd_18`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
17	`IntPnd_17`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
16	`IntPnd_16`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
15	`IntPnd_15`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
14	`IntPnd_14`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
13	`IntPnd_13`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
12	`IntPnd_12`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
11	`IntPnd_11`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
10	`IntPnd_10`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
9	`IntPnd_9`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
8	`IntPnd_8`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
7	`IntPnd_7`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
6	`IntPnd_6`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
5	`IntPnd_5`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
4	`IntPnd_4`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
3	`IntPnd_3`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
2	`IntPnd_2`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
1	`IntPnd_1`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
0	`IntPnd_0`	Interrupt pending bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

MOIPD

Interrupt pending bits for Message Objects 97 to 128. By reading the IntPnd bits, the CPU can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/ reset by the CPU via the IFx Message Interface Registers or set by the Message Handler after reception or after a successful transmission of a frame. This will also affect the valid of IntID in the Interrupt Register.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000BC
can1	0xFFC01000	0xFFC010BC

Offset: `0xBC`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IntPnd_31 RO 0x0	IntPnd_30 RO 0x0	IntPnd_29 RO 0x0	IntPnd_28 RO 0x0	IntPnd_27 RO 0x0	IntPnd_26 RO 0x0	IntPnd_25 RO 0x0	IntPnd_24 RO 0x0	IntPnd_23 RO 0x0	IntPnd_22 RO 0x0	IntPnd_21 RO 0x0	IntPnd_20 RO 0x0	IntPnd_19 RO 0x0	IntPnd_18 RO 0x0	IntPnd_17 RO 0x0	IntPnd_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IntPnd_15 RO 0x0	IntPnd_14 RO 0x0	IntPnd_13 RO 0x0	IntPnd_12 RO 0x0	IntPnd_11 RO 0x0	IntPnd_10 RO 0x0	IntPnd_9 RO 0x0	IntPnd_8 RO 0x0	IntPnd_7 RO 0x0	IntPnd_6 RO 0x0	IntPnd_5 RO 0x0	IntPnd_4 RO 0x0	IntPnd_3 RO 0x0	IntPnd_2 RO 0x0	IntPnd_1 RO 0x0	IntPnd_0 RO 0x0

MOIPD Fields

Bit	Name	Description	Access	Reset
31	IntPnd_31	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
30	IntPnd_30	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
29	IntPnd_29	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
28	IntPnd_28	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
27	IntPnd_27	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
26	IntPnd_26	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
25	IntPnd_25	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
24	IntPnd_24	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
23	`IntPnd_23`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
22	`IntPnd_22`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
21	`IntPnd_21`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
20	`IntPnd_20`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
19	`IntPnd_19`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
18	`IntPnd_18`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
17	`IntPnd_17`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`
16	`IntPnd_16`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	`RO`	`0x0`

Bit	Name	Description	Access	Reset
15	`IntPnd_15`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
14	`IntPnd_14`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
13	`IntPnd_13`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
12	`IntPnd_12`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
11	`IntPnd_11`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
10	`IntPnd_10`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
9	`IntPnd_9`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
8	`IntPnd_8`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
7	`IntPnd_7`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
6	`IntPnd_6`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
5	`IntPnd_5`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
4	`IntPnd_4`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

Bit	Name	Description	Access	Reset
3	`IntPnd_3`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
2	`IntPnd_2`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
1	`IntPnd_1`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0
0	`IntPnd_0`	Interrupt pending bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is not the source of an interrupt. 0x1 The Message Object is the source of an interrupt.	RO	0x0

MOVALX

Reading this register allows the CPU to quickly detect if any of the message valid bits in each of the MOVALA, MOVALB, MOVALC, and MOVALD Message Valid Registers are set.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC000C0`
`can1`	`0xFFC01000`	`0xFFC010C0`

Offset: `0xC0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgValD_3 RO 0x0	MsgValD_2 RO 0x0	MsgValD_1 RO 0x0	MsgValD_0 RO 0x0	MsgValC_3 RO 0x0	MsgValC_2 RO 0x0	MsgValC_1 RO 0x0	MsgValC_0 RO 0x0	MsgValB_3 RO 0x0	MsgValB_2 RO 0x0	MsgValB_1 RO 0x0	MsgValB_0 RO 0x0	MsgValA_3 RO 0x0	MsgValA_2 RO 0x0	MsgValA_1 RO 0x0	MsgValA_0 RO 0x0

MOVALX Fields

Bit	Name	Description	Access	Reset
15	MsgValD_3	Each bit in this field is a logical OR of a byte of the MOVALD register. Array index i corresponds to byte i of the MOVALD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALD are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALD are configured and should be considered by the Message Handler.	RO	0x0
14	MsgValD_2	Each bit in this field is a logical OR of a byte of the MOVALD register. Array index i corresponds to byte i of the MOVALD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALD are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALD are configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
13	`MsgValD_1`	Each bit in this field is a logical OR of a byte of the MOVALD register. Array index i corresponds to byte i of the MOVALD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALD are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALD are configured and should be considered by the Message Handler.	RO	0x0
12	`MsgValD_0`	Each bit in this field is a logical OR of a byte of the MOVALD register. Array index i corresponds to byte i of the MOVALD register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALD are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALD are configured and should be considered by the Message Handler.	RO	0x0
11	`MsgValC_3`	Each bit in this field is a logical OR of a byte of the MOVALC register. Array index i corresponds to byte i of the MOVALC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALC are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALC are configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
10	MsgValC_2	Each bit in this field is a logical OR of a byte of the MOVALC register. Array index i corresponds to byte i of the MOVALC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALC are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALC are configured and should be considered by the Message Handler.	RO	0x0
9	MsgValC_1	Each bit in this field is a logical OR of a byte of the MOVALC register. Array index i corresponds to byte i of the MOVALC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALC are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALC are configured and should be considered by the Message Handler.	RO	0x0
8	MsgValC_0	Each bit in this field is a logical OR of a byte of the MOVALC register. Array index i corresponds to byte i of the MOVALC register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALC are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALC are configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
7	`MsgValB_3`	Each bit in this field is a logical OR of a byte of the MOVALB register. Array index i corresponds to byte i of the MOVALB register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MOVALB are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALB are configured and should be considered by the Message Handler.	RO	0x0
6	`MsgValB_2`	Each bit in this field is a logical OR of a byte of the MOVALB register. Array index i corresponds to byte i of the MOVALB register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MOVALB are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALB are configured and should be considered by the Message Handler.	RO	0x0
5	`MsgValB_1`	Each bit in this field is a logical OR of a byte of the MOVALB register. Array index i corresponds to byte i of the MOVALB register. **Value** / **Description** 0x0 The Message Objects in the corresponding byte of MOVALB are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALB are configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
4	`MsgValB_0`	Each bit in this field is a logical OR of a byte of the MOVALB register. Array index i corresponds to byte i of the MOVALB register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALB are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALB are configured and should be considered by the Message Handler.	RO	0x0
3	`MsgValA_3`	Each bit in this field is a logical OR of a byte of the MOVALA register. Array index i corresponds to byte i of the MOVALA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALA are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALA are configured and should be considered by the Message Handler.	RO	0x0
2	`MsgValA_2`	Each bit in this field is a logical OR of a byte of the MOVALA register. Array index i corresponds to byte i of the MOVALA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALA are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALA are configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
1	MsgValA_1	Each bit in this field is a logical OR of a byte of the MOVALA register. Array index i corresponds to byte i of the MOVALA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALA are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALA are configured and should be considered by the Message Handler.	RO	0x0
0	MsgValA_0	Each bit in this field is a logical OR of a byte of the MOVALA register. Array index i corresponds to byte i of the MOVALA register. **Value** **Description** 0x0 The Message Objects in the corresponding byte of MOVALA are ignored by the Message Handler. 0x1 One or more of the Message Objects in the corresponding byte of MOVALA are configured and should be considered by the Message Handler.	RO	0x0

MOVALA

Message valid bits for Message Objects 1 to 32. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000C4
can1	0xFFC01000	0xFFC010C4

Offset: 0xC4

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal_31 RO 0x0	MsgVal_30 RO 0x0	MsgVal_29 RO 0x0	MsgVal_28 RO 0x0	MsgVal_27 RO 0x0	MsgVal_26 RO 0x0	MsgVal_25 RO 0x0	MsgVal_24 RO 0x0	MsgVal_23 RO 0x0	MsgVal_22 RO 0x0	MsgVal_21 RO 0x0	MsgVal_20 RO 0x0	MsgVal_19 RO 0x0	MsgVal_18 RO 0x0	MsgVal_17 RO 0x0	MsgVal_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgVal_15 RO 0x0	MsgVal_14 RO 0x0	MsgVal_13 RO 0x0	MsgVal_12 RO 0x0	MsgVal_11 RO 0x0	MsgVal_10 RO 0x0	MsgVal_9 RO 0x0	MsgVal_8 RO 0x0	MsgVal_7 RO 0x0	MsgVal_6 RO 0x0	MsgVal_5 RO 0x0	MsgVal_4 RO 0x0	MsgVal_3 RO 0x0	MsgVal_2 RO 0x0	MsgVal_1 RO 0x0	MsgVal_0 RO 0x0

MOVALA Fields

Bit	Name	Description	Access	Reset
31	MsgVal_31	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
30	MsgVal_30	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
29	MsgVal_29	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
28	`MsgVal_28`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
27	`MsgVal_27`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
26	`MsgVal_26`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
25	`MsgVal_25`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
24	`MsgVal_24`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
23	`MsgVal_23`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
22	`MsgVal_22`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
21	`MsgVal_21`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
20	`MsgVal_20`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
19	`MsgVal_19`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
18	`MsgVal_18`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
17	`MsgVal_17`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
16	`MsgVal_16`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
15	`MsgVal_15`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
14	`MsgVal_14`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
13	`MsgVal_13`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
12	`MsgVal_12`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
11	`MsgVal_11`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
10	`MsgVal_10`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
9	`MsgVal_9`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
8	`MsgVal_8`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
7	`MsgVal_7`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
6	`MsgVal_6`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
5	`MsgVal_5`	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
4	MsgVal_4	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
3	MsgVal_3	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
2	MsgVal_2	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
1	MsgVal_1	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
0	MsgVal_0	Message valid bits for Message Objects 1 to 32. Array index i corresponds to Message Object i+1. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

MOVALB

Message valid bits for Message Objects 33 to 64. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC000C8
can1	0xFFC01000	0xFFC010C8

Offset: 0xC8

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal_31 RO 0x0	MsgVal_30 RO 0x0	MsgVal_29 RO 0x0	MsgVal_28 RO 0x0	MsgVal_27 RO 0x0	MsgVal_26 RO 0x0	MsgVal_25 RO 0x0	MsgVal_24 RO 0x0	MsgVal_23 RO 0x0	MsgVal_22 RO 0x0	MsgVal_21 RO 0x0	MsgVal_20 RO 0x0	MsgVal_19 RO 0x0	MsgVal_18 RO 0x0	MsgVal_17 RO 0x0	MsgVal_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgVal_15 RO 0x0	MsgVal_14 RO 0x0	MsgVal_13 RO 0x0	MsgVal_12 RO 0x0	MsgVal_11 RO 0x0	MsgVal_10 RO 0x0	MsgVal_9 RO 0x0	MsgVal_8 RO 0x0	MsgVal_7 RO 0x0	MsgVal_6 RO 0x0	MsgVal_5 RO 0x0	MsgVal_4 RO 0x0	MsgVal_3 RO 0x0	MsgVal_2 RO 0x0	MsgVal_1 RO 0x0	MsgVal_0 RO 0x0

MOVALB Fields

Bit	Name	Description	Access	Reset
31	MsgVal_31	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
30	MsgVal_30	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
29	MsgVal_29	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
28	MsgVal_28	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
27	MsgVal_27	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
26	MsgVal_26	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
25	MsgVal_25	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
24	MsgVal_24	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
23	`MsgVal_23`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
22	`MsgVal_22`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
21	`MsgVal_21`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
20	`MsgVal_20`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
19	`MsgVal_19`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
18	`MsgVal_18`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
17	`MsgVal_17`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
16	`MsgVal_16`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
15	`MsgVal_15`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
14	`MsgVal_14`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
13	`MsgVal_13`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
12	`MsgVal_12`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
11	`MsgVal_11`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
10	`MsgVal_10`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
9	`MsgVal_9`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
8	`MsgVal_8`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
7	`MsgVal_7`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
6	`MsgVal_6`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
5	`MsgVal_5`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
4	`MsgVal_4`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
3	`MsgVal_3`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
2	`MsgVal_2`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
1	`MsgVal_1`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
0	`MsgVal_0`	Message valid bits for Message Objects 33 to 64. Array index i corresponds to Message Object i+33. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0

MOVALC

Message valid bits for Message Objects 65 to 96. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC000CC`

Module Instance	Base Address	Register Address
can1	0xFFC01000	0xFFC010CC

Offset: 0xCC

Access: RO

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal_31 RO 0x0	MsgVal_30 RO 0x0	MsgVal_29 RO 0x0	MsgVal_28 RO 0x0	MsgVal_27 RO 0x0	MsgVal_26 RO 0x0	MsgVal_25 RO 0x0	MsgVal_24 RO 0x0	MsgVal_23 RO 0x0	MsgVal_22 RO 0x0	MsgVal_21 RO 0x0	MsgVal_20 RO 0x0	MsgVal_19 RO 0x0	MsgVal_18 RO 0x0	MsgVal_17 RO 0x0	MsgVal_16 RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MsgVal_15 RO 0x0	MsgVal_14 RO 0x0	MsgVal_13 RO 0x0	MsgVal_12 RO 0x0	MsgVal_11 RO 0x0	MsgVal_10 RO 0x0	MsgVal_9 RO 0x0	MsgVal_8 RO 0x0	MsgVal_7 RO 0x0	MsgVal_6 RO 0x0	MsgVal_5 RO 0x0	MsgVal_4 RO 0x0	MsgVal_3 RO 0x0	MsgVal_2 RO 0x0	MsgVal_1 RO 0x0	MsgVal_0 RO 0x0

MOVALC Fields

Bit	Name	Description	Access	Reset
31	MsgVal_31	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
30	MsgVal_30	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
29	MsgVal_29	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
28	MsgVal_28	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
27	MsgVal_27	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
26	MsgVal_26	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
25	`MsgVal_25`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
24	`MsgVal_24`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
23	`MsgVal_23`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
22	`MsgVal_22`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
21	`MsgVal_21`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
20	`MsgVal_20`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
19	`MsgVal_19`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
18	`MsgVal_18`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
17	`MsgVal_17`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
16	`MsgVal_16`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
15	`MsgVal_15`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
14	`MsgVal_14`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
13	`MsgVal_13`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
12	`MsgVal_12`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
11	`MsgVal_11`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
10	`MsgVal_10`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
9	`MsgVal_9`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
8	`MsgVal_8`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
7	`MsgVal_7`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
6	`MsgVal_6`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
5	`MsgVal_5`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
4	`MsgVal_4`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
3	`MsgVal_3`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
2	`MsgVal_2`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
1	`MsgVal_1`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
0	`MsgVal_0`	Message valid bits for Message Objects 65 to 96. Array index i corresponds to Message Object i+65. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

MOVALD

Message valid bits for Message Objects 97 to 128. By reading the MsgVal bits, the CPU can check for which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC000D0`
`can1`	`0xFFC01000`	`0xFFC010D0`

Offset: `0xD0`

Access: `RO`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
`MsgVal_31` RO 0x0	`MsgVal_30` RO 0x0	`MsgVal_29` RO 0x0	`MsgVal_28` RO 0x0	`MsgVal_27` RO 0x0	`MsgVal_26` RO 0x0	`MsgVal_25` RO 0x0	`MsgVal_24` RO 0x0	`MsgVal_23` RO 0x0	`MsgVal_22` RO 0x0	`MsgVal_21` RO 0x0	`MsgVal_20` RO 0x0	`MsgVal_19` RO 0x0	`MsgVal_18` RO 0x0	`MsgVal_17` RO 0x0	`MsgVal_16` RO 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`MsgVal_15` RO 0x0	`MsgVal_14` RO 0x0	`MsgVal_13` RO 0x0	`MsgVal_12` RO 0x0	`MsgVal_11` RO 0x0	`MsgVal_10` RO 0x0	`MsgVal_9` RO 0x0	`MsgVal_8` RO 0x0	`MsgVal_7` RO 0x0	`MsgVal_6` RO 0x0	`MsgVal_5` RO 0x0	`MsgVal_4` RO 0x0	`MsgVal_3` RO 0x0	`MsgVal_2` RO 0x0	`MsgVal_1` RO 0x0	`MsgVal_0` RO 0x0

MOVALD Fields

Bit	Name	Description	Access	Reset
31	`MsgVal_31`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
30	`MsgVal_30`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
29	`MsgVal_29`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
28	`MsgVal_28`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
27	`MsgVal_27`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
26	`MsgVal_26`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
25	`MsgVal_25`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
24	`MsgVal_24`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
23	`MsgVal_23`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
22	`MsgVal_22`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
21	`MsgVal_21`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
20	`MsgVal_20`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
19	MsgVal_19	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
18	MsgVal_18	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
17	MsgVal_17	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
16	MsgVal_16	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
15	`MsgVal_15`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
14	`MsgVal_14`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
13	`MsgVal_13`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
12	`MsgVal_12`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
11	`MsgVal_11`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
10	`MsgVal_10`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
9	`MsgVal_9`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
8	`MsgVal_8`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** / **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
7	`MsgVal_7`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
6	`MsgVal_6`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
5	`MsgVal_5`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0
4	`MsgVal_4`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Bit	Name	Description	Access	Reset
3	`MsgVal_3`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
2	`MsgVal_2`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
1	`MsgVal_1`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0
0	`MsgVal_0`	Message valid bits for Message Objects 97 to 128. Array index i corresponds to Message Object i+97. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RO	0x0

Message Interface Group Register Descriptions

These registers provide indirect read and write access for the host CPU to the Message RAM. Buffers the data to be transferred to and from the RAM, avoiding conflicts between CPU accesses and CAN frame reception/transmission. The function of the two Interface Register sets is identical. The second interface register set is provided to serve application programming. Two groups of software drivers may defined, each group is restricted to the use of one of the Interface Register sets. The software drivers of one group may interrupt software drivers of the other group, but not of the same group. In a simple example, there is one Read_Message task that uses IF1 to get received messages from the Message RAM and there is one

Write_Message task that uses IF2 to write messages to be transmitted into the Message RAM. Both tasks may interrupt each other. Each set of Interface Registers consists controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Offset: `0x100`

IF1CMR on page 25-207
The control bits of the IF1/2 Command Register specify the transfer direction and select which portions of the Message Object should be transferred. A message transfer is started as soon as the CPU has written the message number to the low byte of the Command Request Register and IFxCMR.AutoInc is zero. With this write operation, the IFxCMR.Busy bit is automatically set to 1 to notify the CPU that a transfer is in progress. After a wait time of 2 to 8 HOST_CLK periods, the transfer between theInterface Register and the Message RAM has been completed and the IFxCMR.Busy bit is cleared to 0. The upper limit of the wait time occurs when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both Command Registers consecutively (requests a second transfer while another transfer is already in progress), the second transfer starts when the first one is completed. Note: While Busy bit of IF1/2 Command Register is one, IF1/2 Register Set is write protected.

IF1MSK on page 25-211
The Message Object Mask Bits together with the arbitration bits are used for acceptance filtering of incoming messages. Note: While IFxCMR.Busy bit is one, the IF1/2 Register Set is write protected.

IF1ARB on page 25-213
The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

IF1MCTR on page 25-214
The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

IF1DA on page 25-217
The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

IF1DB on page 25-218
The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

IF2CMR on page 25-219
The control bits of the IF1/2 Command Register specify the transfer direction and select which portions of the Message Object should be transferred. A message transfer is started as soon as the CPU has written the message number to the low byte of the Command Request Register and IFxCMR.AutoInc is zero. With this write operation, the IFxCMR.Busy bit is automatically set to 1 to notify the CPU that a transfer is in progress. After a wait time of 2 to 8 HOST_CLK periods, the transfer between theInterface Register and the Message RAM has been completed and the IFxCMR.Busy bit is cleared to 0. The upper limit of the wait time occurs when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both Command Registers consecutively (requests a second transfer while another transfer is already in progress), the second transfer starts when the first one is completed. Note: While Busy bit of IF1/2 Command Register is one, IF1/2 Register Set is write protected.

IF2MSK on page 25-223
The Message Object Mask Bits together with the arbitration bits are used for acceptance filtering of incoming messages. Note: While IFxCMR.Busy bit is one, the IF1/2 Register Set is write protected.

IF2ARB on page 25-225
The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

IF2MCTR on page 25-226
The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

IF2DA on page 25-229
The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

IF2DB on page 25-230
The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

IF1CMR

The control bits of the IF1/2 Command Register specify the transfer direction and select which portions of the Message Object should be transferred. A message transfer is started as soon as the CPU has written the message number to the low byte of the Command Request Register and IFxCMR.AutoInc is zero. With this write operation, the IFxCMR.Busy bit is automatically set to 1 to notify the CPU that a transfer is in progress. After a wait time of 2 to 8 HOST_CLK periods, the transfer between theInterface Register and the Message RAM has been completed and the IFxCMR.Busy bit is cleared to 0. The upper limit of the wait time occurs when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both Command Registers consecutively (requests a second transfer while another transfer is already in progress), the second transfer starts when the first one is completed. Note: While Busy bit of IF1/2 Command Register is one, IF1/2 Register Set is write protected.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00100
can1	0xFFC01000	0xFFC01100

Offset: 0x100

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		ClrAutoInc RW 0x0	Reserved					WR1RD0 RW 0x0	Mask RW 0x0	Arb RW 0x0	Control RW 0x0	ClrIntPnd RW 0x0	TxRqstNewDat RW 0x0	DataA RW 0x0	DataB RW 0x0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Busy RO 0x0	DMAactive RW 0x0	AutoInc RW 0x0	Reserved					MONum RW 0x1							

IF1CMR Fields

Bit	Name	Description	Access	Reset
29	ClrAutoInc	Clear the AutoInc bit without starting a transfer **Value** / **Description** 0x0 Has no effect to the other Bits of this Register. 0x1 Clear the AutoInc bit without starting a transfer, all other bits will be ignored.	RW	0x0

Bit	Name	Description	Access	Reset
23	WR1RD0	Write / Read Transfer **Value** **Description** 0x0 Transfer data from the Message Object addressed by IFxCMR.MONum into the selected IFx Message Buffer Registers. 0x1 Transfer data from the selected IFx Message Buffer Registers to the Message Object addressed by IFxCMR.MONum.	RW	0x0
22	Mask	Write Direction: 0= Mask bits unchanged. 1= transfer Identifier Mask + MDir + MXtd to Message Object. Read Direction: 0= Mask bits unchanged. 1= transfer Identifier Mask + MDir + MXtd to IFxMSK Register.	RW	0x0
21	Arb	Write Direction: 0= Arbitration bits unchanged. 1= transfer Identifier + Dir + Xtd + MsgVal to Message Object. Read Direction: 0= Arbitration bits unchanged. 1= transfer Identifier + Dir + Xtd + MsgVal to IFxARB Register.	RW	0x0
20	Control	Write Direction: 0= Control Bits unchanged. 1= transfer Control Bits to Message Object. Note: If IFxCMR.TxRqst/NewDat bit is set, bits IFxMCTR.TxRqst and IFxMCTR.NewDat will be ignored. Read Direction: 0= Control Bits unchanged. 1= transfer Control Bits to IFxMCTR Register.	RW	0x0
19	ClrIntPnd	Write Direction: Has no influence to Message Object at write transfer. Note: When writing to a Message Object, this bit is ignored and copying of IntPnd flag from IFx Control Register to Message RAM could only be controlled by IFxMTR.IntPnd bit. Read Direction: 0= IntPnd bit remains unchanged. 1= clear IntPnd bit in the Message Object.	RW	0x0

Bit	Name	Description	Access	Reset
18	TxRqstNewDat	Write Direction: 0= TxRqst and NewDat bit will be handled according IFxMCTR.NewDat bit and IFxMCTR.TxRqst bit. 1= set TxRqst and NewDat in Message Object to one Note: If a CAN transmission is requested by setting IFxCMR.TxRqst/NewDat, the TxRqst and NewDat bits in the Message Object will be set to one independently of the values in IFxMCTR. Read Direction: 0= NewDat bit remains unchanged. 1= clear NewDat bit in the Message Object. Note: A read access to a Message Object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IFxMCTR always reflect the status before resetting them.	RW	0x0
17	DataA	Write Direction: 0= Data Bytes 0-3 unchanged. 1= transfer Data Bytes 0-3 to Message Object. Read Direction: 0= Data Bytes 0-3 unchanged. 1= transfer Data Bytes 0-3 to IFxDA.	RW	0x0
16	DataB	Write Direction: 0= Data Bytes 4-7 unchanged. 1= transfer Data Bytes 4-7 to Message Object. Read Direction: 0= Data Bytes 4-7 unchanged. 1= transfer Data Bytes 4-7 to IFxDB. Note: The speed of the message transfer does not depend on how many bytes are transferred.	RW	0x0
15	Busy	Busy Flag **Value** **Description** 0x0 Set to zero when read/write action has finished. 0x1 Set to one when writing to the IFxCMR.MONum. While bit is one, IFx Register Set is write protected.	RO	0x0

Bit	Name	Description	Access	Reset
14	`DMAactive`	Activation of DMA feature for subsequent internal IFx Register Set **Value** / **Description** 0x0 — DMA line leaves passive, independent of IFx activities. 0x1 — By writing to the Command Request Register, an internal transfer of Message Object Data between RAM and IFx will be initiated. When this transfer is complete and DMAactive bit was set, the CAN_IFxDMA line gets active. The DMAactive bit and port CAN_IFxDMA are staying active until first read or write access to one of the IFx registers. If AutoInc is set DMAactive will be left active, otherwise the bit is reset. Note: Due to auto reset feature of DMAactive bit if AutoInc is inactive, this bit has to be set for each subsequent DMA cycle separately. DMA line has to be enabled in CAN Control Register.	RW	0x0

Bit	Name	Description	Access	Reset
13	`AutoInc`	Automatic Increment of Message Object Number The behavior of the Message Object Number increment depends on the Transfer Direction, IFxCMR.WR1RD0. * Read: The first transfer will be initiated (Busy Bit will set) at write of IFxCMR.MONum. The Message Object Number will be incremented and the next Message Object will be transferred from Message Object RAM to Interface Registers after a read access of Data-Byte 7. * Write: The first as well as each other transfer will be started after write access to Data- Byte7. The Message Object Number will be incremented after successful transfer from the Interface Registers to the Message Object RAM. Always after successful transfer the Busy Bit will be reset. In combination with DMAactive the port CAN_IFxDMA is set, too. Note: If the direction is configured as Read a write access to Data-Byte 7 will not start any transfer, as well as if the direction is configured as Write a read access to Data-Byte 7 will not start any transfer. At transfer direction Read each read of Data-Byte 7 will start a transfer until IFxCMR.AutoInc is reset. To aware of resetting a NewDat bit of the following message object, the application has to reset IFxCMR.AutoInc before reading the Data-Byte 7 of the last message object which will be read. **Value** — **Description** 0x0 — AutoIncrement of Message Object Number disabled. 0x1 — AutoIncrement of Message Object Number enabled.	RW	0x0
7:0	`MONum`	0x01-0x80 Valid Message Number, the Message Object in the Message RAM is selected for data transfer (up to 128 MsgObj). 0x00 Not a valid Message Number, interpreted as 0x80. 0x81-0xFF Not a valid Message Number, interpreted as 0x01-0x7F. Note: When an invalid Message Number is written to IFxCMR.MONum which is higher than the last Message Object number, a modulo addressing will occur.When e.g. accessing Message Object 33 in a CAN module with 32 Message Objects only, the Message Object 1 will be accessed instead.	RW	0x1

IF1MSK

The Message Object Mask Bits together with the arbitration bits are used for acceptance filtering of incoming messages. Note: While IFxCMR.Busy bit is one, the IF1/2 Register Set is write protected.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00104
can1	0xFFC01000	0xFFC01104

Offset: 0x104

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd RW 0x1	MDir RW 0x1	Reserved	Msk RW 0x1FFFFFFF												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk RW 0x1FFFFFFF															

IF1MSK Fields

Bit	Name	Description	Access	Reset
31	MXtd	When 11-bit (standard) Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. **Value** **Description** 0x0 The extended identifier bit (IDE) has no effect on the acceptance filtering. 0x1 The extended identifier bit (IDE) is used for acceptance filtering.	RW	0x1
30	MDir	**Value** **Description** 0x0 The message direction bit (Dir) has no effect on the acceptance filtering. Handle with care setting IFxMSK.MDir to zero. 0x1 The message direction bit (Dir) is used for acceptance filtering.	RW	0x1
28:0	Msk	0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.	RW	0x1FFFFFFF

IF1ARB

The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00108`
`can1`	`0xFFC01000`	`0xFFC01108`

Offset: `0x108`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal RW 0x0	Xtd RW 0x0	Dir RW 0x0	ID RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID RW 0x0															

IF1ARB Fields

Bit	Name	Description	Access	Reset
31	`MsgVal`	The CPU must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the Messages Object is no longer used in operation. For reconfiguration of Message Objects during normal operation. **Value** — **Description** 0x0 — The Message Object is ignored by the Message Handler. 0x1 — The Message Object is configured and should be considered by the Message Handler.	RW	0x0

Bit	Name	Description	Access	Reset
30	`Xtd`	Extended Identifier **Value** **Description** 0x0 The 11-bit (standard) Identifier will be used for this Message Object. 0x1 The 29-bit (extended) Identifier will be used for this Message Object.	RW	0x0
29	`Dir`	Message Direction **Value** **Description** 0x0 On TxRqst, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object. 0x1 On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this Message Object is set (if RmtEn = one).	RW	0x0
28:0	`ID`	ID28 - ID0 29-bit Identifier (Extended Frame). ID28 - ID18 11-bit Identifier (Standard Frame).	RW	0x0

IF1MCTR

The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC0010C
can1	0xFFC01000	0xFFC0110C

Offset: `0x10C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat RW 0x0	MsgLst RW 0x0	IntPnd RW 0x0	UMask RW 0x0	TxIE RW 0x0	RxIE RW 0x0	RmtEn RW 0x0	TxRqst RW 0x0	EoB RW 0x0	Reserved			DLC RW 0x0			

IF1MCTR Fields

Bit	Name	Description	Access	Reset
15	NewDat	New Data **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RW	0x0
14	MsgLst	Message Lost **Value** **Description** 0x0 No message lost since last time this bit was reset by the CPU. 0x1 The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.	RW	0x0
13	IntPnd	Interrupt Pending **Value** **Description** 0x0 This message object is not the source of an interrupt. 0x1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.	RW	0x0

Bit	Name	Description	Access	Reset
12	`UMask`	Use Acceptance Mask **Value** — **Description** 0x0 — Acceptance formula1: (RTRRx == ~DIR) && (IDERx == IDE) && (IDRx == ID) 0x1 — (Msk28-0, MXtd, and MDir) for acceptance filtering, formula: ((RTRRx & MDIR) == (~DIR & MDIR)) && ((IDERx & MXtd) == (IDE & MXtd)) && ((IDRx & Msk) == (ID & Msk)) Note: If the UMask bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MsgVal is set to one.	`RW`	`0x0`
11	`TxIE`	Transmit Interrupt Enable **Value** — **Description** 0x0 — IntPnd will be left unchanged after the successful transmission of a frame. 0x1 — IntPnd will be set after a successful transmission of a frame.	`RW`	`0x0`
10	`RxIE`	Receive Interrupt Enable **Value** — **Description** 0x0 — IntPnd will be left unchanged after the successful reception of a frame. 0x1 — IntPnd will be set after a successful reception of a frame.	`RW`	`0x0`
9	`RmtEn`	Remote Enable **Value** — **Description** 0x0 — At the reception of a Remote Frame, TxRqst is left unchanged. 0x1 — At the reception of a Remote Frame, TxRqst is set.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
8	`TxRqst`	Transmit Request **Value** **Description** 0x0 This Message Object is not waiting for transmission. 0x1 The transmission of this Message Object is requested and is not yet done.	`RW`	`0x0`
7	`EoB`	Note: This bit is used to concatenate two or more Message Objects (up to 128) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer) this bit must always be set to one. **Value** **Description** 0x0 Message Object belongs to a FIFO Buffer Block and is not the last Message Object of that FIFO Buffer Block. 0x1 Single Message Object or last Message Object of a FIFO Buffer Block.	`RW`	`0x0`
3:0	`DLC`	0-8 Data Frame has 0-8 data bytes. 9-15 Data Frame has 8 data bytes. Note: The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.	`RW`	`0x0`

IF1DA

The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00110`
`can1`	`0xFFC01000`	`0xFFC01110`

Offset: `0x110`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data3 RW 0x0								Data2 RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data1 RW 0x0								Data0 RW 0x0							

IF1DA Fields

Bit	Name	Description	Access	Reset
31:24	Data3	4th data byte of a CAN Data Frame	RW	0x0
23:16	Data2	3rd data byte of a CAN Data Frame	RW	0x0
15:8	Data1	2nd data byte of a CAN Data Frame	RW	0x0
7:0	Data0	1st data byte of a CAN Data Frame	RW	0x0

IF1DB

The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00114
can1	0xFFC01000	0xFFC01114

Offset: 0x114

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data7 RW 0x0								Data6 RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data5 RW 0x0								Data4 RW 0x0							

IF1DB Fields

Bit	Name	Description	Access	Reset
31:24	Data7	8th data byte of a CAN Data Frame	RW	0x0
23:16	Data6	7th data byte of a CAN Data Frame	RW	0x0

Bit	Name	Description	Access	Reset
15:8	Data5	6th data byte of a CAN Data Frame	RW	0x0
7:0	Data4	5th data byte of a CAN Data Frame	RW	0x0

IF2CMR

The control bits of the IF1/2 Command Register specify the transfer direction and select which portions of the Message Object should be transferred. A message transfer is started as soon as the CPU has written the message number to the low byte of the Command Request Register and IFxCMR.AutoInc is zero. With this write operation, the IFxCMR.Busy bit is automatically set to 1 to notify the CPU that a transfer is in progress. After a wait time of 2 to 8 HOST_CLK periods, the transfer between theInterface Register and the Message RAM has been completed and the IFxCMR.Busy bit is cleared to 0. The upper limit of the wait time occurs when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage. If the CPU writes to both Command Registers consecutively (requests a second transfer while another transfer is already in progress), the second transfer starts when the first one is completed. Note: While Busy bit of IF1/2 Command Register is one, IF1/2 Register Set is write protected.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00120
can1	0xFFC01000	0xFFC01120

Offset: 0x120

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		ClrAutoInc RW 0x0	Reserved					WR1RD0 RW 0x0	Mask RW 0x0	Arb RW 0x0	Control RW 0x0	ClrIntPnd RW 0x0	TxRqstNewDat RW 0x0	DataA RW 0x0	DataB RW 0x0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Busy RO 0x0	DMAactive RW 0x0	AutoInc RW 0x0	Reserved					MONum RW 0x1							

IF2CMR Fields

Bit	Name	Description	Access	Reset
29	ClrAutoInc	Clear the AutoInc bit without starting a transfer **Value** / **Description** 0x0 — Has no effect to the other Bits of this Register. 0x1 — Clear the AutoInc bit without starting a transfer, all other bits will be ignored.	RW	0x0
23	WR1RD0	Write / Read Transfer **Value** / **Description** 0x0 — Transfer data from the Message Object addressed by IFxCMR.MONum into the selected IFx Message Buffer Registers. 0x1 — Transfer data from the selected IFx Message Buffer Registers to the Message Object addressed by IFxCMR.MONum.	RW	0x0
22	Mask	Write Direction: 0= Mask bits unchanged. 1= transfer Identifier Mask + MDir + MXtd to Message Object. Read Direction: 0= Mask bits unchanged. 1= transfer Identifier Mask + MDir + MXtd to IFxMSK Register.	RW	0x0
21	Arb	Write Direction: 0= Arbitration bits unchanged. 1= transfer Identifier + Dir + Xtd + MsgVal to Message Object. Read Direction: 0= Arbitration bits unchanged. 1= transfer Identifier + Dir + Xtd + MsgVal to IFxARB Register.	RW	0x0
20	Control	Write Direction: 0= Control Bits unchanged. 1= transfer Control Bits to Message Object. Note: If IFxCMR.TxRqst/NewDat bit is set, bits IFxMCTR.TxRqst and IFxMCTR.NewDat will be ignored. Read Direction: 0= Control Bits unchanged. 1= transfer Control Bits to IFxMCTR Register.	RW	0x0
19	ClrIntPnd	Write Direction: Has no influence to Message Object at write transfer. Note: When writing to a Message Object, this bit is ignored and copying of IntPnd flag from IFx Control Register to Message RAM could only be controlled by IFxMTR.IntPnd bit. Read Direction: 0= IntPnd bit remains unchanged. 1= clear IntPnd bit in the Message Object.	RW	0x0

Bit	Name	Description	Access	Reset
18	TxRqstNewDat	Write Direction: 0= TxRqst and NewDat bit will be handled according IFxMCTR.NewDat bit and IFxMCTR.TxRqst bit. 1= set TxRqst and NewDat in Message Object to one Note: If a CAN transmission is requested by setting IFxCMR.TxRqst/NewDat, the TxRqst and NewDat bits in the Message Object will be set to one independently of the values in IFxMCTR. Read Direction: 0= NewDat bit remains unchanged. 1= clear NewDat bit in the Message Object. Note: A read access to a Message Object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IFxMCTR always reflect the status before resetting them.	RW	0x0
17	DataA	Write Direction: 0= Data Bytes 0-3 unchanged. 1= transfer Data Bytes 0-3 to Message Object. Read Direction: 0= Data Bytes 0-3 unchanged. 1= transfer Data Bytes 0-3 to IFxDA.	RW	0x0
16	DataB	Write Direction: 0= Data Bytes 4-7 unchanged. 1= transfer Data Bytes 4-7 to Message Object. Read Direction: 0= Data Bytes 4-7 unchanged. 1= transfer Data Bytes 4-7 to IFxDB. Note: The speed of the message transfer does not depend on how many bytes are transferred.	RW	0x0
15	Busy	Busy Flag **Value** **Description** 0x0 Set to zero when read/write action has finished. 0x1 Set to one when writing to the IFxCMR.MONum. While bit is one, IFx Register Set is write protected.	RO	0x0

Bit	Name	Description	Access	Reset
14	`DMAactive`	Activation of DMA feature for subsequent internal IFx Register Set **Value** — **Description** 0x0 — DMA line leaves passive, independent of IFx activities. 0x1 — By writing to the Command Request Register, an internal transfer of Message Object Data between RAM and IFx will be initiated. When this transfer is complete and DMAactive bit was set, the CAN_IFxDMA line gets active. The DMAactive bit and port CAN_IFxDMA are staying active until first read or write access to one of the IFx registers. If AutoInc is set DMAactive will be left active, otherwise the bit is reset. Note: Due to auto reset feature of DMAactive bit if AutoInc is inactive, this bit has to be set for each subsequent DMA cycle separately. DMA line has to be enabled in CAN Control Register.	`RW`	`0x0`

Bit	Name	Description	Access	Reset
13	AutoInc	Automatic Increment of Message Object Number The behavior of the Message Object Number increment depends on the Transfer Direction, IFxCMR.WR1RD0. * Read: The first transfer will be initiated (Busy Bit will set) at write of IFxCMR.MONum. The Message Object Number will be incremented and the next Message Object will be transferred from Message Object RAM to Interface Registers after a read access of Data-Byte 7. * Write: The first as well as each other transfer will be started after write access to Data- Byte7. The Message Object Number will be incremented after successful transfer from the Interface Registers to the Message Object RAM. Always after successful transfer the Busy Bit will be reset. In combination with DMAactive the port CAN_IFxDMA is set, too. Note: If the direction is configured as Read a write access to Data-Byte 7 will not start any transfer, as well as if the direction is configured as Write a read access to Data-Byte 7 will not start any transfer. At transfer direction Read each read of Data-Byte 7 will start a transfer until IFxCMR.AutoInc is reset. To aware of resetting a NewDat bit of the following message object, the application has to reset IFxCMR.AutoInc before reading the Data-Byte 7 of the last message object which will be read. **Value** — **Description** 0x0 — AutoIncrement of Message Object Number disabled. 0x1 — AutoIncrement of Message Object Number enabled.	RW	0x0
7:0	MONum	0x01-0x80 Valid Message Number, the Message Object in the Message RAM is selected for data transfer (up to 128 MsgObj). 0x00 Not a valid Message Number, interpreted as 0x80. 0x81-0xFF Not a valid Message Number, interpreted as 0x01-0x7F. Note: When an invalid Message Number is written to IFxCMR.MONum which is higher than the last Message Object number, a modulo addressing will occur.When e.g. accessing Message Object 33 in a CAN module with 32 Message Objects only, the Message Object 1 will be accessed instead.	RW	0x1

IF2MSK

The Message Object Mask Bits together with the arbitration bits are used for acceptance filtering of incoming messages. Note: While IFxCMR.Busy bit is one, the IF1/2 Register Set is write protected.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00124
can1	0xFFC01000	0xFFC01124

Offset: 0x124

Access: RW

Bit Fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MXtd RW 0x1	MDir RW 0x1	Reserved	Msk RW 0x1FFFFFFF												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Msk RW 0x1FFFFFFF															

IF2MSK Fields

Bit	Name	Description	Access	Reset
31	MXtd	When 11-bit (standard) Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 are considered. **Value** **Description** 0x0 The extended identifier bit (IDE) has no effect on the acceptance filtering. 0x1 The extended identifier bit (IDE) is used for acceptance filtering.	RW	0x1
30	MDir	**Value** **Description** 0x0 The message direction bit (Dir) has no effect on the acceptance filtering. Handle with care setting IFxMSK.MDir to zero. 0x1 The message direction bit (Dir) is used for acceptance filtering.	RW	0x1
28:0	Msk	0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.	RW	0x1FFFFFFF

IF2ARB

The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00128
can1	0xFFC01000	0xFFC01128

Offset: 0x128

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MsgVal RW 0x0	Xtd RW 0x0	Dir RW 0x0	ID RW 0x0												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID RW 0x0															

IF2ARB Fields

Bit	Name	Description	Access	Reset
31	MsgVal	The CPU must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init in the CAN Control Register. MsgVal must also be reset if the Messages Object is no longer used in operation. For reconfiguration of Message Objects during normal operation. **Value** **Description** 0x0 The Message Object is ignored by the Message Handler. 0x1 The Message Object is configured and should be considered by the Message Handler.	RW	0x0

Bit	Name	Description	Access	Reset
30	`Xtd`	Extended Identifier **Value** **Description** 0x0 The 11-bit (standard) Identifier will be used for this Message Object. 0x1 The 29-bit (extended) Identifier will be used for this Message Object.	`RW`	`0x0`
29	`Dir`	Message Direction **Value** **Description** 0x0 On TxRqst, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object. 0x1 On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit of this Message Object is set (if RmtEn = one).	`RW`	`0x0`
28:0	`ID`	ID28 - ID0 29-bit Identifier (Extended Frame). ID28 - ID18 11-bit Identifier (Standard Frame).	`RW`	`0x0`

IF2MCTR

The Arbitration Registers ID28-0, Xtd, and Dir are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0, MXtd, and MDir) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction=receive (Data Frame) or Direction=transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC0012C`
`can1`	`0xFFC01000`	`0xFFC0112C`

Offset: `0x12C`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NewDat RW 0x0	MsgLst RW 0x0	IntPnd RW 0x0	UMask RW 0x0	TxIE RW 0x0	RxIE RW 0x0	RmtEn RW 0x0	TxRqst RW 0x0	EoB RW 0x0	Reserved			DLC RW 0x0			

IF2MCTR Fields

Bit	Name	Description	Access	Reset
15	NewDat	New Data **Value** **Description** 0x0 No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. 0x1 The Message Handler or the CPU has written new data into the data portion of this Message Object.	RW	0x0
14	MsgLst	Message Lost **Value** **Description** 0x0 No message lost since last time this bit was reset by the CPU. 0x1 The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.	RW	0x0
13	IntPnd	Interrupt Pending **Value** **Description** 0x0 This message object is not the source of an interrupt. 0x1 This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.	RW	0x0

Bit	Name	Description	Access	Reset
12	UMask	Use Acceptance Mask **Value** — **Description** 0x0 — Acceptance formula1: (RTRRx == ~DIR) && (IDERx == IDE) && (IDRx == ID) 0x1 — (Msk28-0, MXtd, and MDir) for acceptance filtering, formula: ((RTRRx & MDIR) == (~DIR & MDIR)) && ((IDERx & MXtd) == (IDE & MXtd)) && ((IDRx & Msk) == (ID & Msk)) Note: If the UMask bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MsgVal is set to one.	RW	0x0
11	TxIE	Transmit Interrupt Enable **Value** — **Description** 0x0 — IntPnd will be left unchanged after the successful transmission of a frame. 0x1 — IntPnd will be set after a successful transmission of a frame.	RW	0x0
10	RxIE	Receive Interrupt Enable **Value** — **Description** 0x0 — IntPnd will be left unchanged after the successful reception of a frame. 0x1 — IntPnd will be set after a successful reception of a frame.	RW	0x0
9	RmtEn	Remote Enable **Value** — **Description** 0x0 — At the reception of a Remote Frame, TxRqst is left unchanged. 0x1 — At the reception of a Remote Frame, TxRqst is set.	RW	0x0

Bit	Name	Description	Access	Reset
8	`TxRqst`	Transmit Request **Value** — **Description** 0x0 — This Message Object is not waiting for transmission. 0x1 — The transmission of this Message Object is requested and is not yet done.	`RW`	`0x0`
7	`EoB`	Note: This bit is used to concatenate two or more Message Objects (up to 128) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer) this bit must always be set to one. **Value** — **Description** 0x0 — Message Object belongs to a FIFO Buffer Block and is not the last Message Object of that FIFO Buffer Block. 0x1 — Single Message Object or last Message Object of a FIFO Buffer Block.	`RW`	`0x0`
3:0	`DLC`	0-8 Data Frame has 0-8 data bytes. 9-15 Data Frame has 8 data bytes. Note: The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.	`RW`	`0x0`

IF2DA

The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Module Instance	Base Address	Register Address
`can0`	`0xFFC00000`	`0xFFC00130`
`can1`	`0xFFC01000`	`0xFFC01130`

Offset: `0x130`

Access: `RW`

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data3 RW 0x0								Data2 RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data1 RW 0x0								Data0 RW 0x0							

IF2DA Fields

Bit	Name	Description	Access	Reset
31:24	Data3	4th data byte of a CAN Data Frame	RW	0x0
23:16	Data2	3rd data byte of a CAN Data Frame	RW	0x0
15:8	Data1	2nd data byte of a CAN Data Frame	RW	0x0
7:0	Data0	1st data byte of a CAN Data Frame	RW	0x0

IF2DB

The data bytes of CAN messages are stored in the IF1/2 registers in the following order. In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Module Instance	Base Address	Register Address
can0	0xFFC00000	0xFFC00134
can1	0xFFC01000	0xFFC01134

Offset: 0x134

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Data7 RW 0x0								Data6 RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data5 RW 0x0								Data4 RW 0x0							

IF2DB Fields

Bit	Name	Description	Access	Reset
31:24	Data7	8th data byte of a CAN Data Frame	RW	0x0
23:16	Data6	7th data byte of a CAN Data Frame	RW	0x0

Bit	Name	Description	Access	Reset
15:8	`Data5`	6th data byte of a CAN Data Frame	`RW`	`0x0`
7:0	`Data4`	5th data byte of a CAN Data Frame	`RW`	`0x0`

Document Revision History

Table 25-4: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Add address map and register definitions
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Minor formatting updates
November 2012	1.2	• Minor updates. • Expanded reset section. • Expanded interrupts seciton.
May 2012	1.1	Added block diagram and system integration, functional description, programming model, and address map and register definitions sections.
January 2012	1.0	Initial release.

Introduction to the HPS Component 26

2014.06.30

cv_54026 Subscribe Send Feedback

The hard processor system (HPS) component is a soft component that you can instantiate in the FPGA fabric of the Cyclone® V SoC. It enables other soft components to interface with the HPS hard logic. The HPS component itself has a small footprint in the FPGA fabric, because its only purpose is to enable soft logic to connect to the extensive hard logic in the HPS.

For a description of the HPS and its integration into the system on a chip (SoC), refer to the *Cyclone V Device Datasheet*. For a description of HPS system architecture and features, refer to the *Introduction to the Hard Processor* chapter in volume 3 of the *Cyclone V Device Handbook* and the *CoreSight Debug and Trace* chapter in volume 3 of the *Cyclone V Device Handbook*.

The HPS supports the following peripheral architectures and features. The chapters that describe these features can be found on the Cyclone V documentation page.

- Clock manager
- Reset manager
- Interconnect
- HPS-FPGA AXI Bridge
- Cortex-A9 Microprocessor Unit Subsystem
- CoreSight Debug and Trace
- SDRAM Controller Subsystem
- On-Chip RAM and boot block ROM
- NAND Flash Controller
- SD/MMC Controller
- Quad SPI Flash Controller
- FPGA Manager Block Diagram and System Integration
- System Manager
- Scan Manager
- DMA Controller
- Ethernet Media Access Controller
- USB OTG Controller
- SPI Controller
- I^{2}C Controller
- UART Controller

- General-Purpose I/O Interface
- Timer
- Watchdog Timer

Related Information

- **Introduction to Cyclone V Hard Processor System** on page 1-1
- **CoreSight Debug and Trace** on page 10-1
- **http://www.altera.com/literature/hb/cyclone-v/cv_51002.pdf**
  For a description of the HPS and its integration into the system on a chip (SoC).

Document Revision History

Table 26-1: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Maintenance release
February 2014	2014.02.28	Maintenance release
December 2013	2013.12.30	Maintenance release
June 2012	1.0	Maintenance release.
May 2012	0.1	Preliminary draft.

Instantiating the HPS Component 27

2014.06.30

cv_54027 Subscribe Send Feedback

You instantiate the hard processor system (HPS) component in Qsys. The HPS is available in the Qsys IP catalog under **Processor and Peripherals** > **Hard Processor Systems**. This chapter describes the parameters available in the HPS component parameter editor, which opens when you add or edit an HPS component.

Related Information

- **http://www.altera.com/literature/hb/cyclone-v/cv_51002.pdf**
  The HPS requires specific device targets. For a detailed list of supported devices, refer to the *Cyclone V Device Datasheet*.
- **http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf**
  For general information about using Qsys, refer to the *Creating a System with Qsys* chapter in the *Quartus® II* Handbook.

FPGA Interfaces

The **FPGA Interfaces** tab is one of four tabs on the HPS Component. This tab contains several groups with the following parameters:

General Interfaces on page 27-1

AXI Bridges on page 27-3

FPGA-to-HPS SDRAM Interface on page 27-4

Reset Interfaces on page 27-5

DMA Peripheral Request on page 27-6

Interrupts on page 27-6

Related Information

Introduction to the HPS Component on page 26-1
For general information about interfaces, refer to *HPS Component Interfaces*.

General Interfaces

When enabled, the interfaces described in the following table become visible in the HPS component.

Table 27-1: General Parameters

Parameter Name	Parameter Description	Interface Name
Enable MPU standby and event signals	Enables interfaces that perform the following functions: • Notify the FPGA fabric that the microprocessor unit (MPU) is in standby mode. • Wake up an MPCore processor from a wait for event (WFE) state.	`h2f_mpu_events`
Enable general purpose signals	Enables a pair of 32-bit unidirectional general-purpose interfaces between the FPGA fabric and the FPGA manager in the HPS portion of the SoC device.	`h2f_gp`

Enable Debug APB interface	Enables debug interface to the FPGA, allowing access to debug components in the HPS. For more information, refer to the *CoreSight Debug and Trace* chapter.	`h2f_debug_apb`
		`h2f_debug_apb_sideband`
		`h2f_debug_apb_clock`
		`h2f_debug_apb_reset`

Enable System Trace Macrocell hardware events	Enables system trace macrocell (STM) hardware events, allowing logic inside the FPGA to insert messages into the trace stream. For more information, refer to the *CoreSight Debug and Trace* chapter.	`f2h_stm_hw_events`
Enable FPGA Cross Trigger interface	Enables the cross trigger interface (CTI), which allows trigger sources and sinks to interface with the embedded cross trigger (ECT). For more information, refer to the *CoreSight Debug and Trace* chapter.	`h2f_cti` `h2f_cti_clock`

Enable FPGA Trace Port Interface Unit	Enables an interface between the trace port interface unit (TPIU) and logic in the FPGA. The TPIU is a bridge between on-chip trace sources and a trace port. For more information, refer to the *CoreSight Debug and Trace* chapter.	`h2f_tpiu`
		`h2f_tpiu_clock_in`
		`h2f_tpiu_clock`

Parameter Name	Parameter Description	Interface Name
Enable boot from FPGA signals	Enable an input to the HPS indicating whether a preloader is available in on-chip memory of FPGA. This option also enables a separate input to the HPS indicating a fallback preloader is available in the FPGA memory. A fallback preloader is used when there is no valid preloader image found in flash memory. For more information, refer to *Appendix A: Booting and Configuration*.	`f2h_boot_from_fpga`
Enable HLGPI Interface	Enable a general purpose interface that is connected to the General Purpose I/O (GPIO) peripheral of HPS. This is an input-only interface with 14-bit width. This interface shares the I/O pins with the HPS DDR SDRAM controller.	`hps_io (hps_io_gpio_inst_HLGPI[0..13])`

Related Information

- **CoreSight Debug and Trace** on page 10-1
  Trace port interface unit (TPIU). Enabling the TPIU exposes trace signals to the device pins. Refer to the *CoreSight Debug and Trace* for more information.
- **Booting and Configuration Introduction** on page 30-1
  For detailed information about the HPS boot sequence, refer to the *Booting and Configuration*.

AXI Bridges

Table 27-2: Bridge Parameters

Parameter Name	Parameter Description	Interface Name
FPGA-to-HPS interface width	Enable or disable the FPGA-to-HPS interface; if enabled, set the data width to 32, 64, or 128 bits.	`f2h_axi_slave` `f2h_axi_clock`
HPS-to-FPGA interface width	Enable or disable the HPS-to-FPGA interface; if enabled, set the data width to 32, 64, or 128 bits.	`h2f_axi_master` `h2f_axi_clock`
Lightweight HPS-to-FPGA interface width	Enable or disable the lightweight HPS-to-FPGA interface. When enabled, the data width is 32 bits.	`h2f_lw_axi_master` `h2f_lw_axi_clock`

Note: To facilitate accessing these slaves from a memory-mapped master with a smaller address width, you can use the Altera® Address Span Extender.

Related Information

- Using the Address Span Extender Component on page 27-15
  Address span extender details
-

FPGA-to-HPS SDRAM Interface

In the **FPGA-to-HPS SDRAM Interface** table, use + or – to add or remove FPGA-to-HPS SDRAM interfaces.

You can add one or more SDRAM ports that make the HPS SDRAM subsystem accessible to the FPGA fabric.

Table 27-3: FPGA-to-HPS SDRAM Port and Interface Names

Port Name	Interface Name
f2h_sdram0	`f2h_sdram0_data` `f2h_sdram0_clock`
f2h_sdram1	`f2h_sdram1_data` `f2h_sdram1_clock`
f2h_sdram2	`f2h_sdram2_data` `f2h_sdram2_clock`
f2h_sdram3	`f2h_sdram3_data` `f2h_sdram3_clock`
f2h_sdram4	`f2h_sdram4_data` `f2h_sdram4_clock`
f2h_sdram5	`f2h_sdram5_data` `f2h_sdram5_clock`

The following table shows the parameters available for each SDRAM interface where the **Name** parameter denotes the interface name.

Table 27-4: FPGA-to-HPS SDRAM Interface Parameters

Parameter Name	Parameter Description
Name	Port name (auto assigned as shown in Table 27-3 table below)

Parameter Name	Parameter Description
Type	Interface type: • AXI-3 • Avalon-MM Bidirectional • Avalon-MM Write-only • Avalon-MM Read-only
Width	32, 64, 128, or 256

Note: You can configure the slave interface to a data width of 32, 64, 128, or 256 bits. To facilitate accessing this slave from a memory-mapped master with a smaller address width, you can use the *Altera Address Span Extender*.

Related Information

- **Using the Address Span Extender Component** on page 27-15
  Address span extender details
- **Functional Description—HPS Memory Controller** on page 11-1

Reset Interfaces

You can enable most resets on an individual basis.

Table 27-5: Reset Parameters

Parameter Name	Parameter Description	Interface Name
Enable HPS-to-FPGA cold reset output	Enable interface for HPS-to-FPGA cold reset output	`h2f_cold_reset`
Enable HPS warm reset handshake signals	Enable an additional pair of reset handshake signals allowing soft logic to notify the HPS when it is safe to initiate a warm reset in the FPGA fabric.	`h2f_warm_reset_handshake`
Enable FPGA-to-HPS debug reset request	Enable interface for FPGA-to-HPS debug reset request	`f2h_debug_reset_req`
Enable FPGA-to-HPS warm reset request	Enable interface for FPGA-to-HPS warm reset request	`f2h_warm_reset_req`
Enable FPGA-to-HPS cold reset request	Enable interface for FPGA-to-HPS cold reset request	`f2h_cold_reset_req`

Related Information

Reset Manager on page 3-1
For more information about the reset interfaces, refer to *Functional Description of the Reset Manager* in the *Reset Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

DMA Peripheral Request

You can enable each direct memory access (DMA) controller peripheral request ID individually. Each request ID enables an interface for FPGA soft logic to request one of eight logical DMA channels to the FPGA.

Related Information

DMA Controller on page 16-1

For more information, refer to the *DMA Controller* chapter in the *Cyclone V Device Handbook, Volume 3*.

Interrupts

Table 27-6: FPGA-to-HPS Interrupts Interface

Parameter Name	Parameter Description	Interface Name
Enable FPGA-to-HPS Interrupts	Enables interface for FPGA interrupt signals to the MPU (in the HPS).	f2h_irq0 f2h_irq1

You can enable the interfaces for each individual HPS peripheral interrupt to the FPGA. Table 27-7 shows the available HPS to FPGA interrupt interfaces and the corresponding parameters to enable them.

Table 27-7: HPS-to-FPGA Interrupts Interface

Parameter Name	Parameter Description	Interface Name
Enable CAN interrupts	Enables interface for HPS CAN controllers interrupt to the FPGA	`h2f_can0_interrupt` `h2f_can1_interrupt`
Enable clock peripheral interrupts	Enables interface for HPS clock manager and MPU wake-up interrupt signals to the FPGA	`h2f_clkmgr_interrupt` `h2f_mpuwakeup_interrupt`
Enable CTI interrupts	Enables interface for HPS cross-trigger interrupt signals to the FPGA	`h2f_cti_interrupt0` `h2f_cti_interrupt1`

Parameter Name	Parameter Description	Interface Name

Enable DMA interrupts	Enables interface for HPS DMA channels interrupt and DMA abort interrupt to the FPGA	`h2f_dma_interrupt0`
		`h2f_dma_interrupt1`
		`h2f_dma_interrupt2`
		`h2f_dma_interrupt3`
		`h2f_dma_interrupt4`
		`h2f_dma_interrupt5`
		`h2f_dma_interrupt6`
		`h2f_dma_interrupt7`
		`h2f_dma_abort_interrupt`

Enable EMAC interrupts	Enables interface for HPS Ethernet MAC controller interrupt to the FPGA	`h2f_emac0_interrupt` `h2f_emac1_interrupt`
Enable FPGA manager interrupts	Enables interface for the HPS FPGA manager interrupt to the FPGA	`h2f_fpga_man_interrupt`

Enable GPIO interrupts	Enables interface for the HPS general purpose IO (GPIO) interrupt to the FPGA	`h2f_gpio0_interrupt`
		`h2f_gpio1_interrupt`
		`h2f_gpio2_interrupt`

Enable I^{2}C-EMAC interrupts (for I2C2 and I2C3)	Enables interface for the HPS I^{2}C interrupt to the FPGA (I^{2}C used for EMAC media interface control)	`h2f_i2c_emac0_interrupt` `h2f_i2c_emac1_interrupt`

Enable I^{2}C peripheral interrupts (for I2C0 and I2C1)	Enables interface for the HPS I^{2}C interrupt to the FPGA (I^{2}C used for general purpose)	`h2f_i2c0 _interrupt`
		`h2f_i2c1_interrupt`

Enable L4 timer interrupts	Enables interface for the HPS SP timer interrupt to the FPGA. For more information, refer to the Timer Clock Characteristics table in the *Timer* chapter in the *Cyclone V Device Handbook, Volume 3.*	`h2f_l4sp0_interrupt` `h2f_l4sp1_interrupt`

Parameter Name	Parameter Description	Interface Name
Enable NAND interrupts	Enables interface for the HPS NAND controller interrupt to the FPGA	`h2f_nand_interrupt`
Enable OSC timer interrupts	Enables interface for the HPS OSC timer interrupt to the FPGA. For more information, refer to the Timer Clock Characteristics table in the *Timer* chapter in the *Cyclone V Device Handbook, Volume 3*.	`h2f_osc0_interrupt` `h2f_osc1_interrupt`
Enable Quad SPI interrupts	Enables interface for the HPS QSPI controller interrupt to the FPGA	`h2f_qspi_interrupt`
Enable SD/MMC interrupts	Enables interface for the HPS SD/MMC controller interrupt to the FPGA	`h2f_sdmmc_interrupt`
Enable SPI master interrupts	Enables interface for the HPS SPI master controller interrupt to the FPGA	`h2f_spi0_interrupt` `h2f_spi1_interrupt`
Enable SPI slave interrupts	Enables interface for the HPS SPI slave controller interrupt to the FPGA	`h2f_spi2_interrupt` `h2f_spi3_interrupt`
Enable UART interrupts	Enables interface for the HPS UART controller interrupt to the FPGA	`h2f_uart0_interrupt` `h2f_uart1_interrupt`
Enable USB interrupts	Enables interface for the HPS USB controller interrupt to the FPGA	`h2f_usb0_interrupt` `h2f_usb1_interrupt`
Enable watchdog interrupts	Enables interface for the HPS watchdog interrupt to the FPGA	`h2f_wdog0_interrupt` `h2f_wdog1_interrupt`

Related Information

Timer on page 23-1

Configuring Peripheral Pin Multiplexing

The **Peripheral Pin Multiplexing** tab is one of four tabs on the HPS Component. This tab contains several groups with the following parameters:

Configuring Peripherals on page 27-9

Connecting Unassigned Pins to GPIO on page 27-10

Using Unassigned IO as LoanIO on page 27-10
You can utilize unused HPS IOs as LoanIO, which is directly driven by the FPGA and can be used as input, output, or bi-directional.

Resolving Pin Multiplexing Conflicts on page 27-10
Use the **Peripherals MUX Table** to view pins with invalid multiple assignments.

Route Peripheral Signals to FPGA on page 27-11
You can route the peripheral signals to the FPGA fabric and assign them to the FPGA I/O pins.

Configuring Peripherals

The **Peripheral Pin Multiplexing** tab contains a group of parameters for each available type of peripheral. You can enable one or more instances of each peripheral type by selecting an HPS I/O pin set for each instance. When enabled, some peripherals also have a mode settings specific to their functions.

When you assign a peripheral to an HPS I/O pin set, the corresponding peripheral is highlighted in the **Peripherals MUX Table** at the end of **Peripheral Pin Multiplexing** tab. The other unused peripheral pin set remains un-highlighted in the table.

Each list in the **Peripheral Pin Multiplexing** tab has a hint describing in detail the options available in the list. The hint for each mode list shows the signals used by each available mode. View each hint by hovering over the corresponding mode list.

You can enable the following types of peripherals. For details of peripheral-specific settings, refer to the chapter for each peripheral:

Related Information

- CoreSight Debug and Trace on page 10-1
  Trace port interface unit (TPIU). Enabling the TPIU exposes trace signals to the device pins. Refer to the *CoreSight Debug and Trace* for more information.
- NAND Flash Controller on page 13-1
- SD/MMC Controller on page 14-1
  Secure Digital / MultiMediaCard (SD/MMC) Controller chapter in the *Cyclone V Device Handbook, Volume 3*.
- Quad SPI Flash Controller on page 15-1
  Quad serial peripheral interface (SPI) Flash Controller chapter in the *Cyclone V Device Handbook, Volume 3*.
- Ethernet Media Access Controller on page 17-1
  Ethernet Media Access Controller chapter in the *Cyclone V Device Handbook, Volume 3*.
- USB 2.0 OTG Controller on page 18-1
- SPI Controller on page 19-1
- I^{2}C Controller on page 20-1

- UART Controller on page 21-1
- CAN Controller Introduction on page 25-1

Connecting Unassigned Pins to GPIO

For pins that are not assigned to any peripherals, you can assign them to the GPIO peripheral by clicking on the corresponding GPIO push button in the table. Click on the selected push button to remove GPIO pin assignment. The table also shows the GPIO bit number that correlates to the I/O pins.

Using Unassigned IO as LoanIO

You can utilize unused HPS IOs as LoanIO, which is directly driven by the FPGA and can be used as input, output, or bi-directional.

Each LOANIO port has an input, output, and output enable, which directly controls the HPS IO functions. The LoanIO only operates when the HPS registers have been set up in the pre-loader to allow their operation. The LoanIO are asynchronous, thus no clocking is required.

Use the following steps to enable the LoanIO signals:

1. Select **Peripheral Pins Multiplexing** tap in the HPS Megawizard.
2. Choose the corresponding LoanIO pins from the **Peripherals Mux Table**, click the push button to select/unselect it.
3. Export the peripheral signals out of the Qsys system.
4. In the Quartus II software, connect the user logic to the LoanIO interface to drive the HPS IOs.

Table 27-8: Generated Conduit Signal Interface

Conduit Name	Direction	Declarations
._hps_io_gpio_inst_LOANIOXX	Bi-direction	User must declare as a top-level pin; pin assignment is hardcoded following the HPS IO location.
._h2f_loan_io_in[48]	Out	HPS IO data input signal, output to FPGA user logic.
._h2f_loan_io_out[48]	In	HPS IO data output signal, input from FPGA user logic.
._h2f_loan_io_oe[48]	In	HPS IO data output enable signal, input from FPGA user logic.

Resolving Pin Multiplexing Conflicts

Use the **Peripherals MUX Table** to view pins with invalid multiple assignments.

Pins that have multiple peripherals assigned to them are highlighted in red color with the conflicting peripherals in boldface font style. Solve the multiplexing conflicts by de-selecting peripherals that are assigned to the pins, leaving only one peripheral, which is needed.

[48] Qsys will generate a full signal array for `h2f_loan_io_in, h2f_loan_io_out`, and `h2f_loan_io_oe`. You must assign user logic to the specific signal array. For example, you have triggered LoanIO 40, so its respective signal array is `h2f_loan_io_in[40], h2f_loan_io_out[40]`, and `h2f_loan_io_oe[40]`.

Route Peripheral Signals to FPGA

You can route the peripheral signals to the FPGA fabric and assign them to the FPGA I/O pins.

The following steps show you how to enable the peripheral signals:

1. Select FPGA in the peripheral pin multiplexing selection drop-down box.
2. Export the peripheral signals out of Qsys system.
3. In the Quartus II software, connect the signals to the FPGA I/O pins.

Configuring HPS Clocks

The **HPS Clocks** tab is one of four tabs on the HPS Component. This tab contains several groups with the following parameters:

User Clocks on page 27-11

PLL Reference Clocks on page 27-12

Peripheral FPGA Clocks on page 27-12

User Clocks

When you enable a user clock, you must manually enter its maximum frequency for timing analysis. The TimeQuest Timing Analyzer has no other information about how software running on the HPS configures the phase-locked loop (PLL) outputs. Each possible clock, including clocks that are available from peripherals, has its own parameter for describing the clock frequency.

Related Information

Selecting PLL Output Frequency and Phase on page 27-14

User Clock Parameters

The frequencies that you provide are the maximum expected frequencies. The actual clock frequencies can be modified through the register interface, for example by software running on the microprocessor unit (MPU). For further details, refer to *Selecting PLL Output Frequency and Phase*.

Table 27-9: User Clock Parameters

Parameter Name	Parameter Description	Clock Interface Name
Enable HPS-to-FPGA user 0 clock	Enable main PLL from HPS-to-FPGA	`h2f_user0_clock`
User 0 clock frequency	Specify the maximum expected frequency for the main PLL	
Enable HPS-to-FPGA user 1 clock	Enable peripheral PLL from HPS-to-FPGA	`h2f_user1_clock`
User 1 clock frequency	Specify the maximum expected frequency for the peripheral PLL	

Parameter Name	Parameter Description	Clock Interface Name
Enable HPS-to-FPGA user 2 clock	Enable SDRAM PLL from HPS-to-FPGA	`h2f_user2_clock`
User 2 clock frequency	Specify the maximum expected frequency for the SDRAM PLL	

Related Information

Selecting PLL Output Frequency and Phase on page 27-14

Clock Frequency Usage

The clock frequencies you provide are reported in a Synopsys Design Constraints File (**.sdc**) generated by Qsys.

Related Information

- Selecting PLL Output Frequency and Phase on page 27-14
- Clock Manager on page 2-1
  For general information about clock signals, refer to the *Clock Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

PLL Reference Clocks

Table 27-10: PLL Reference Clock Parameters

Parameter Name	Parameter Description	Clock Interface Name
Enable FPGA-to-HPS peripheral PLL reference clock	Enable the interface for FPGA fabric to supply reference clock to HPS peripheral PLL	`f2h_periph_ref_clock`
Enable FPGA-to-HPS SDRAM PLL reference clock	Enable the interface for FPGA fabric to supply reference clock to HPS SDRAM PLL	`f2h_sdram_ref_clock`

Related Information

Clock Manager on page 2-1
For general information about clock signals, refer to the *Clock Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

Peripheral FPGA Clocks

Table 27-11: Peripheral FPGA Clock Parameters

Parameter Name	Parameter Description
EMAC 0 (emac0_md_clk clock frequency)	If EMAC 0 peripheral is routed to FPGA, use this field to specify EMAC 0 MDIO clock frequency

Parameter Name	Parameter Description
EMAC 0 (emac0_gtx_clk clock frequency)	If EMAC 0 peripheral is routed to FPGA, use this field to specify EMAC 0 transmit clock frequency
EMAC 1 (emac1_md_clk clock frequency)	If EMAC 1 peripheral is routed to FPGA, use this field to specify EMAC 1 MDIO clock frequency
EMAC 1 (emac1_gtx_clk clock frequency)	If EMAC 1 peripheral is routed to FPGA, use this field to specify EMAC 1 transmit clock frequency
QSPI (qspi_sclk_out clock frequency)	If QSPI peripheral is routed to FPGA, use this field to specify QSPI serial clock frequency
SPIM 0 (spim0_sclk_out clock frequency)	If SPI master 0 peripheral is routed to FPGA, use this field to specify SPI master 0 output clock frequency
SPIM 1 (spim1_sclk_out clock frequency)	If SPI master 1 peripheral is routed to FPGA, use this field to specify SPI master 1 output clock frequency
I^{2}C0 (i2c0_clk clock frequency)	If I^{2}C 0 peripheral is routed to FPGA, use this field to specify I^{2}C 0 output clock frequency
I^{2}C1 (i2c1_clk clock frequency)	If I^{2}C 1 peripheral is routed to FPGA, use this field to specify I^{2}C 1 output clock frequency
I^{2}C2 (i2c2_clk clock frequency)	If I^{2}C 2 peripheral is routed to FPGA, use this field to specify I^{2}C 2 output clock frequency
I^{2}C3 (i2c3_clk clock frequency)	If I^{2}C 3 peripheral is routed to FPGA, use this field to specify I^{2}C 3 output clock frequency

Configuring the External Memory Interface

The **SDRAM** tab is one of four tabs on the HPS component. This tab contains the PLL output frequency and phase group.

The HPS supports one memory interface implementing double data rate 2 (DDR2), double data rate 3 (DDR3), and low-power double data rate 2 (LPDDR2) protocols. The interface can be up to 40 bits wide with optional error correction code (ECC).

Configuring the HPS SDRAM controller is similar to configuring any other Altera SDRAM controller. There are several important differences:

- The HPS parameter editor supports all SDRAM protocols with one tab. When you parameterize the SDRAM controller, you must specify the memory protocol: DDR2, DDR3, or LPDDR2.

To select the memory protocol, select DDR2, DDR3, or LPDDR2 from the **SDRAM Protocol** list in the **SDRAM** tab. After you select the protocol, settings not applicable to that protocol are disabled.

- Many HPS SDRAM controller settings are the same as for Altera’s dedicated DDR2, DDR3, and LPDDR2 controllers. This section only describes SDRAM parameters that are specific to the HPS component.
- Because the HPS memory controller is not configurable through the Quartus II software, the Controller and Diagnostic tabs are not present in the HPS parameter editor.
- Some settings, such as the controller settings, are not included because they can only be configured through the register interface, for example by software running on the MPU.
- Unlike the memory interface clocks in the FPGA, the memory interface clocks for the HPS are initialized by the boot-up code using values provided by the configuration process. You can accept the values provided by UniPHY, or you can use your own PLL settings, as described in *Selecting PLL Output Frequency and Phase*.

Note: The HPS does not support external memory interface (EMIF) synthesis generation, compilation, or timing analysis.

The HPS memory controller cannot be bonded with a memory controller on the FPGA portion of the device.

For detailed information about SDRAM controller parameters, refer to the following chapters:

Related Information

- **Selecting PLL Output Frequency and Phase** on page 27-14
- **http://www.altera.com/literature/hb/external-memory/emi_parameters.pdf**
  The Implementing and Parameterizing Memory IP chapter in the *External Memory Interface* Handbook.
- **http://www.altera.com/literature/hb/external-memory/emi_fd_hard_memory.pdf**
  The Functional Description--Hard Memory Interface chapter in the *External Memory Interface* Handbook. "EMI-Related HPS Features in SoC Devices" describes features specific to the HPS SDRAM controller.

Selecting PLL Output Frequency and Phase

You select PLL output frequency and phase with controls in the **PHY Settings** tab. In the HPS, PLL frequencies and phases are set by software at system startup. A PLL might not be able to produce the exact frequency that you specify in **Memory clock frequency**. Normally, the Quartus II software sets **Achieved memory clock frequency** to the closest achievable frequency, using an algorithm that tries to balance frequency accuracy against clock jitter. This clock frequency is used for timing analysis by the TimeQuest analyzer.

It is possible to use a different software algorithm for configuring the PLLs. You can force the **Achieved memory clock frequency** box to take on the same value as **Memory clock frequency**, by turning on **Use specified frequency instead of calculated frequency** in the **PHY Settings** tab, under **Clocks**.

Note: If you turn on **Use specified frequency instead of calculated frequency**, the Quartus II software assumes that the value in the **Achieved memory clock frequency** box is correct. If it is not, timing analysis results are incorrect.

Related Information

- **http://www.altera.com/literature/hb/external-memory/emi_parameters.pdf**
  The Implementing and Parameterizing Memory IP chapter in the *External Memory Interface* Handbook.

- **http://www.altera.com/literature/hb/external-memory/emi_fd_hard_memory.pdf**
  The Functional Description--Hard Memory Interface chapter in the *External Memory Interface* Handbook. "EMI-Related HPS Features in SoC Devices" describes features specific to the HPS SDRAM controller.

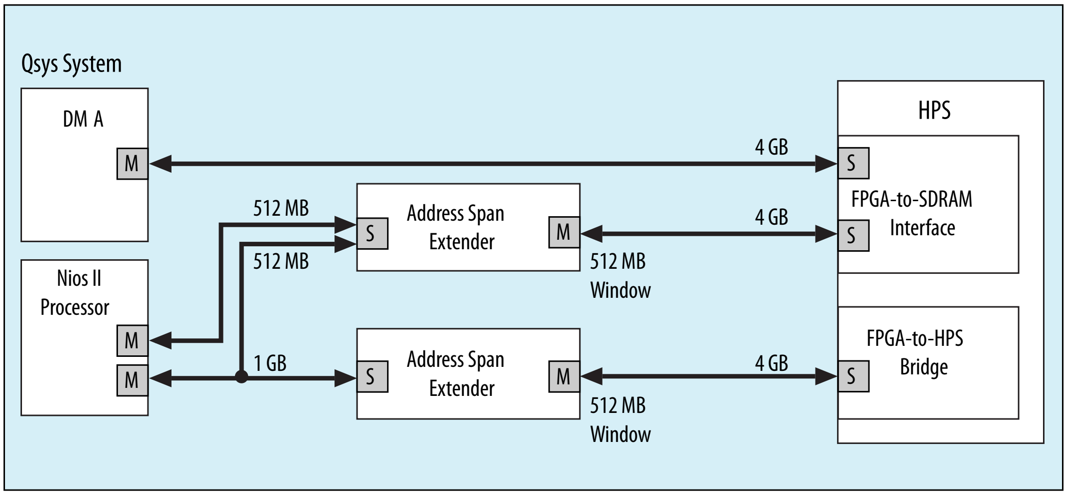
Using the Address Span Extender Component

The FPGA-to-HPS bridge and FPGA-to-HPS SDRAM memory-mapped interfaces expose their entire 4 GB address spaces to the FPGA fabric. The Address Span Extender component provides a memory-mapped window into the address space that it masters. Using the address span extender, you can expose portions of the HPS memory space without needing to expose the entire 4 GB address space.

You can use the address span extender between a soft logic master and an FPGA-to-HPS bridge or FPGA-to-HPS SDRAM interface. This component reduces the number of address bits required for a master to address a memory-mapped slave interface located in the HPS.

Figure 27-1: Address Span Extender Components

Two address span extender components used in a system with the HPS.



You can also use the address span extender in the HPS-to-FPGA direction, for slave interfaces in the FPGA. In this case, the HPS-to-FPGA bridge exposes a limited, variable address space in the FPGA, which can be paged in using the address span extender.

For example, suppose that the HPS-to-FPGA bridge has a 1 GB span, and the HPS needs to access three independent 1 GB memories in the FPGA portion of the device. To achieve this, the HPS programs the address span extender to access one SDRAM (1 GB) in the FPGA at a time. This technique is commonly called paging or windowing.

Related Information

Qsys interconnect and System Design Components
For more information about the address span extender, refer to *Bridges* in the *Qsys* Interconnect and System Design Components chapter in the *Quartus® II* Handbook.

Generating and Compiling the HPS Component

The process of generating and compiling an HPS design is very similar to the process for any other Qsys project. Perform the following steps:

1. Generate the design with Qsys. The generated files include an **.sdc** file containing clock timing constraints. If simulation is enabled, simulation files are also generated.
2. Add *<qsys_system_name>.qip* to the Quartus II project. *<qsys_system_name>.qip* is the Quartus II IP File for the HPS component, generated by Qsys.
3. Perform analysis and synthesis with the Quartus II software.
4. Assign constraints to the SDRAM component. When Qsys generates the HPS component (step 1), it generates the pin assignment Tcl Script File (**.tcl**) to perform memory assignments. The script file name is **<qsys_system_name>_pin_assignments.tcl**, where <qsys_system_name> is the name of your Qsys system. Run this script to assign constraints to the SDRAM component.

   Note: For information about running the pin assignment script, refer to “MegaWizard Plug-In Manager Flow” in the *Implementing and* Parameterizing Memory IP chapter in the *External Memory Interface Handbook.*

   You do not need to specify pin assignments other than memory assignments. When you configure pin multiplexing as described in *Configuring Peripheral Pin Multiplexing*, you implicitly make pin assignments for all HPS peripherals. Each peripheral is routed exclusively to the pins you specify. HPS I/O signals are exported to the top level of the Qsys design, with information enabling the Quartus II software to make pin assignments automatically.

   You can view and modify the assignments in the **Peripheral Pin Multiplexing** tab. You can also view the assignments in the Quartus fitter report.
5. Compile the design with the Quartus II software.
6. Optionally back-annotate the SDRAM pin assignments, to eliminate pin assignment warnings the next time you compile the design.

Related Information

- **Configuring Peripheral Pin Multiplexing**
- **http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf**
  For general information about using Qsys, refer to the *Creating a System with Qsys* chapter in the *Quartus® II* Handbook.
- **http://www.altera.com/literature/hb/external-memory/emi_parameters.pdf**
  The Implementing and Parameterizing Memory IP chapter in the *External Memory Interface* Handbook.
- **Specifying HPS Simulation Model in Qsys** on page 29-11
  For a description of the simulation files generated, refer to "Simulation Flow" in the Simulating the HPS Component chapter of the *Cyclone V Device Handbook, Volume 3.*
- **http://www.altera.com/literature/hb/cyclone-v/cv_54030.pdf**
  For information about back-annotating pin assignments, refer to *About Back-Annotating Assignments* in Quartus II Help.

Document Revision History

Table 27-12: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Updated the Instantiating the HPS Component section. • Added the Using Unassigned IO as LoanIO section.
February 2014	2014.02.28	• Add interfaces to tables • Add parameters to General Parameters table
December 2013	1.2	Maintenance release.
November 2012	1.1	• Added debug interfaces • Added boot options • Corrected slave address width • Corrected SDRAM interface widths • Added TPIU peripheral • Added .sdc file generation • Added .tcl script for memory assignments
June 2012	1.0	Initial release.
May 2012	0.1	Preliminary draft.

HPS Component Interfaces 28

2014.06.30

cv_54028 Subscribe Send Feedback

This chapter describes the interfaces, including clocks and resets, implemented by the hard processor system (HPS) component.

The majority of the resets can be enabled on an individual basis. The exception is the `h2f_reset` interface, which is always enabled.

You must declare the clock frequency of each HPS-to-FPGA clock for timing purposes. Each possible clock, including ones that are available from peripherals, has its own parameter for describing the clock frequency. Declaring the clock frequency for HPS-to-FPGA clocks specifies how you plan to configure the PLLs and peripherals, to enable TimeQuest to accurately estimate system timing. It has no effect on PLL settings.

Related Information

- **Avalon Interface Specifications**
  For Avalon protocol timing, refer to *Avalon Interface Specifications*.
- **HPS Component Interfaces** on page 28-1
  For information about instantiating the HPS component, refer to the *Instantiating the HPS Component* chapter.
- **Info Center**
  For Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) protocol timing, refer to the AMBA AXI Protocol Specification v1.0, which you can download from the ARM info center website.

Memory-Mapped Interfaces

FPGA-to-HPS Bridge

Table 28-1: FPGA-to-HPS Bridges and Clocks

Interface Name	Description	Associated Clock Interface
`f2h_axi_slave`	FPGA-to-HPS AXI slave interface	`f2h_axi_clock`

The FPGA-to-HPS interface is a configurable data width AXI slave allowing FPGA masters to issue transactions to the HPS. This interface allows the FPGA fabric to access the majority of the HPS slaves. This interface also provides a coherent memory interface.

The FPGA-to-HPS interface is an AXI-3 compliant interface with the following features:

- Configurable data width: 32, 64, or 128 bits
- Accelerator Coherency Port (ACP) sideband signals
- HPS-side AXI bridge to manage clock crossing, buffering, and data width conversion

Other interface standards in the FPGA fabric, such as connecting to Avalon® Memory-Mapped (Avalon-MM) interfaces, can be supported through the use of soft logic adapters. The Qsys system integration tool automatically generates adapter logic to connect AXI to Avalon-MM interfaces.

This interface has an address width of 32 bits. To access existing Avalon-MM/AXI masters, you can use the Altera® Address Span Extender.

Related Information

- Clocks on page 28-4
- Features of the HPS-FPGA Bridges on page 8-1
  For more information, refer to the *HPS FPGA AXI Bridges Component* chapter.
- HPS Component Interfaces on page 28-1
  For information about the address span extender, refer to "Using the Address Span Extender Component" in the *Instantiating the HPS Component* chapter.

ACP Sideband Signals

For communication with the ACP on the microprocessor unit (MPU) subsystem, AXI sideband signals are used to describe the inner cacheable attributes for the transaction.

Related Information

Cortex-A9 MPCore on page 9-4
For more information about the ACP sideband signals, refer to the *Cortex-A9 MPU Subsystem* chapter.

HPStoFPGA and Lightweight HPS-to-FPGA Bridges

Table 28-2: HPStoFPGA and Lightweight HPS-to-FPGA Bridges and Clocks

Interface Name	Description	Associated Clock Interface
h2f_axi_master	HPS-to-FPGA AXI master interface	h2f_axi_clock
h2f_lw_axi_master	HPS-to-FPGA lightweight AXI master interface	h2f_lw_axi_clock

The HPS-to-FPGA interface is a configurable data width AXI master (32-, 64-, or 128-bit) that allows HPS masters to issue transactions to the FPGA fabric.

The lightweight HPS-to-FPGA interface is a 32-bit AXI master that allows HPS masters to issue transactions to the FPGA fabric.

Both HPS-to-FPGA interfaces are AXI-3 compliant. The HPS-side AXI bridges manage clock crossing, buffering, and data width conversion where necessary.

Other interface standards in the FPGA fabric, such as connecting to Avalon-MM interfaces, can be supported through the use of soft logic adaptors. The Qsys system integration tool automatically generates adaptor logic to connect AXI to Avalon-MM interfaces.

Each AXI bridge accepts a clock input from the FPGA fabric and performs clock domain crossing internally. The exposed AXI interface operates on the same clock domain as the clock supplied by the FPGA fabric.

Related Information

- **Clocks** on page 28-4
- **Features of the HPS-FPGA Bridges** on page 8-1
  For more information, refer to the *HPS FPGA AXI Bridges Component* chapter.

FPGA-to-HPS SDRAM Interface

The FPGA-to-HPS SDRAM interface is a direct connection between the FPGA fabric and the HPS SDRAM controller. This interface is highly configurable, allowing a mix between number of ports and port width. The interface supports both AXI-3 and Avalon-MM protocols.

Table 28-3: FPGA-to-HPS SDRAM Interfaces and Clocks

Interface Name	Description	Associated Clock Interface
`f2h_sdram0_data`	SDRAM AXI or Avalon-MM port 0	`f2h_sdram0_clock`
`f2h_sdram1_data`	SDRAM AXI or Avalon-MM port 1	`f2h_sdram1_clock`
`f2h_sdram2_data`	SDRAM AXI or Avalon-MM port 2	`f2h_sdram2_clock`
`f2h_sdram3_data`	SDRAM AXI or Avalon-MM port 3	`f2h_sdram3_clock`
`f2h_sdram4_data`	SDRAM AXI or Avalon-MM port 4	`f2h_sdram4_clock`
`f2h_sdram5_data`	SDRAM AXI or Avalon-MM port 5	`f2h_sdram5_clock`

The FPGA-to-HPS SDRAM interface is a configurable interface to the multi-port SDRAM controller.

The total data width of all interfaces is limited to a maximum of 256 bits in the read direction and 256 bits in the write direction. The interface is implemented as four 64-bit read ports and four 64-bit write ports. As a result, the minimum data width used by the interface is 64 bits, regardless of the number or type of interfaces.

You can configure this interface the following ways:

- AXI-3 or Avalon-MM protocol
- Number of interfaces
- Data width of interfaces

The FPGA-to-HPS SDRAM interface supports six command ports, allowing up to six Avalon-MM interfaces or three bidirectional AXI interfaces.

Each command port is available either to implement a read or write command port for AXI, or to form part of an Avalon-MM interface.

You can use a mix of Avalon-MM and AXI interfaces, limited by the number of command/data ports available. Some AXI features are not present in Avalon-MM interfaces.

This interface has an address width of 32 bits. To access existing Avalon-MM/AXI masters, you can use the Altera Address Span Extender.

Related Information

- **Clocks** on page 28-4
- **Features of the SD/MMC Controller** on page 14-1
  For more information about available combinations of interfaces and ports, refer to the *SDRAM Controller Subsystem* chapter.
- **HPS Component Interfaces** on page 28-1
  For information about the address span extender, refer to “Using the Address Span Extender Component” in the *Instantiating the HPS Component* chapter.

Clocks

The HPS-to-FPGA clock interface supplies physical clocks and resets to the FPGA. These clocks and resets are generated in the HPS.

Alternative Clock Inputs to HPS PLLs

This section lists alternative clock inputs to HPS PLLs.

- `f2h_periph_ref_clock`—FPGA-to-HPS peripheral PLL reference clock. You can connect this clock input to a clock in your design that is driven by the clock network on the FPGA side.
- `f2h_sdram_ref_clock`—FPGA-to-HPS SDRAM PLL reference clock. You can connect this clock to a clock in your design that is driven by the clock network on the FPGA side.

User Clocks

A user clock is a PLL output that is connected to the FPGA fabric rather than the HPS. You can connect a user clock to logic that you instantiate in the FPGA fabric.

- `h2f_user0_clock`—HPS-to-FPGA user clock, driven from main PLL
- `h 2f_user1_clock`—HPS-to-FPGA user clock, driven from peripheral PLL
- `h2f_user2_clock`—HPS-to-FPGA user clock, driven from SDRAM PLL

AXI Bridge FPGA Interface Clocks

The AXI interface has an asynchronous clock crossing in the FPGA-to-HPS bridge. The FPGA-to-HPS and HPS-to-FPGA interfaces are synchronized to clocksgenerated in the FPGA fabric. These interfaces can be asynchronous to one another. The SDRAM controller’s multiport front end (MPFE) transfers the data between the FPGA and HPS clock domains.

- `f2h_axi_clock`—AXI slave clock for FPGA-to-HPS bridge, generated in FPGA fabric
- `h2f_axi_clock`—AXI master clock for HPS-to-FPGA bridge, generated in FPGA fabric
- `h2f_lw_axi_clock`—AXI master clock for lightweight HPS-to-FPGA bridge, generated in FPGA fabric

SDRAM Clocks

You can configure the HPS component with up to six FPGA-to-HPS SDRAM clocks.

Each command channel to the SDRAM controller has an individual clock source from the FPGA fabric. The interface clock is always supplied by the FPGA fabric, with clock crossing occurring on the HPS side of the boundary.

The FPGA-to-HPS SDRAM clocks are driven by soft logic in the FPGA fabric.

- `f2h_sdram0_clock`—SDRAM clock for port 0
- `f2h_sdram1_clock`—SDRAM clock for port 1
- `f2h_sdram2_clock`—SDRAM clock for port 2
- `f2h_sdram3_clock`—SDRAM clock for port 3
- `f2h_sdram4_clock`—SDRAM clock for port 4
- `f2h_sdram5_clock`—SDRAM clock for port 5

Peripheral FPGA Clocks

The HPS peripheral clocks are exposed when the peripheral signals are routed to the FPGA.

Table 28-4: Peripheral FPGA Clocks

Clock Name	Description
`emac_md_clk`	Ethernet PHY management interface clock
`emac_gtx_clk`	Ethernet transmit clock that is used by the PHY in GMII mode
`emac_rx_clk_in`	Ethernet MAC reference clock from the PHY
`emac_tx_clk_in`	Ethernet MAC uses this clock input for TX reference
`emac_ptp_ref_clock`	Ethernet timestamp precision time protocol (PTP) reference clock
`qspi_sclk_out`	QSPI master clock output
`spim_sclk_out`	SPI master serial clock output
`spis_sclk_in`	SPI slave serial clock input
`i2c_clk`	I^{2}C outgoing clock (part of the SCL bidirectional pin signals)
`i2c_scl_in`	I^{2}C incoming clock (part of the SCL bidirectional pin signals)

Resets

This section describes the reset interfaces to the HPS component.

Related Information

Reset Manager on page 3-1

For details about the HPS reset sequences, refer to the *Functional Description of the Reset Manager* in the *Reset Manager* chapter .

HPS-to-FPGA Reset Interfaces

The following interfaces allow the HPS to reset soft logic in the FPGA fabric:

- `h2f_reset`—HPS-to-FPGA cold and warm reset
- `h2f_cold_reset`—HPS-to-FPGA cold reset
- `h2f_warm_reset_handshake`—Warm reset request and acknowledge interface between HPS and FPGA

HPS External Reset Sources

The following interfaces allow soft logic in the FPGA fabric to reset the HPS:

- `f2h_cold_reset_req`—FPGA-to-HPS cold reset request
- `f2h_warm_reset_req`—FPGA-to-HPS warm reset request
- `f2h_dbg_reset_req`—FPGA-to-HPS debug reset request

Peripheral Reset Interfaces

The following are Ethernet reset interfaces, that can be used when the Ethernet is routed to the FPGA:

- `emac_tx_reset`— Ethernet transmit clock reset output used to reset external PHY TX clock domain logic
- `emac_rx_reset`— Ethernet receive clock reset output used to reset external PHY RX clock domain logic

Debug and Trace Interfaces

Trace Port Interface Unit

The TPIU is a bridge between on-chip trace sources and a trace port.

- `h2f_tpiu`
- `h2f_tpiu_clock_in`
- `h2f_tpiu_clock`

FPGA System Trace Macrocell Events Interface

The system trace macrocell (STM) hardware events allow logic in the FPGA to insert messages into the trace stream.

- `f2h_stm_hw_events`

FPGA Cross Trigger Interface

The cross trigger interface (CTI) allows trigger sources and sinks to interface with the embedded cross trigger (ECT).

- `h2f_cti`
- `h2f_cti_clock`

Debug APB Interface

The debug Advanced Peripheral Bus (APB™) interface allows debug components in the FPGA fabric to debug components on the CoreSight™ debug APB.

- `h2f_debug_apb`
- `h2f_debug_apb_sideband`
- `h2f_debug_apb_reset`
- `h2f_debug_apb_clock`

Peripheral Signal Interfaces

The DMA controller interface allows soft IP in the FPGA fabric to communicate with the DMA controller in the HPS. You can configure up to eight separate interface channels.

- `f2h_dma_req0`—FPGA DMA controller peripheral request interface 0
- `f2h_dma_re q1`—FPGA DMA controller peripheral request interface 1
- `f2h_dma_req2`—FPGA DMA controller peripheral request interface 2
- `f2h_dma_req3`—FPGA DMA controller peripheral request interface 3
- `f2h_dma_req4`—FPGA DMA controller peripheral request interface 4
- `f2h_dma_req5`—FPGA DMA controller peripheral request interface 5
- `f2h_dma_req6`—FPGA DMA controller peripheral request interface 6
- `f2h_dma_req7`—FPGA DMA controller peripheral request interface 7

Each of the DMA peripheral request interface contains the following three signals:

- `f2h_dma_req`—This signal is used to request burst transfer using the DMA
- `f2h_dma_single`—This signal is used to request single word transfer using the DMA
- `f2h_dma_ack`—This signal indicates the DMA acknowledgment upon requests from the FPGA

Related Information

DMA Controller on page 16-1
For details about the DMA Controller, refer to *DMA controller* chapter.

Other Interfaces

Related Information

Cortex-A9 MPCore on page 9-4
For more information, refer to *Cortex-A9 MPU Subsystem*.

MPU Standby and Event Interfaces

MPU standby and event signals are notification signals to the FPGA fabric that the MPU is in standby. Event signals are used to wake up the Cortex-A9 processors from a wait for event (WFE) state. The standby and event signals are included in the following interfaces:

- `h2f_mpu_events`—MPU standby and event interface, including the following signals.
- `h2f_mpu_eventi`—Sends an event from logic in the FPGA fabric to the MPU. This FPGA-to-HPS signal is used to wake up a processor that is in a Wait For Event state. Asserting this signal has the same effect as executing the `SEV` instruction in the Cortex-A9. This signal must be de-asserted until the FPGA fabric is powered-up and configured.
- `h2f_mpu_evento`—Sends an event from the MPU to logic in the FPGA fabric. This HPS-to-FPGA signal is asserted when an SEV instruction is executed by one of the Cortex-A9 processors.
- `h2f_mpu_standbywfe[1:0]`—Indicates whether each Cortex-A9 processor is in the WFE state
- `h2f_mpu_standbywfi[1:0]`—Indicates whether each Cortex-A9 processor is in the wait for interrupt (WFI) state
- `h2f_mpu_gp`—General purpose interface

The MPU provides signals to indicate when it is in a standby state. These signals are available to custom hardware designs in the FPGA fabric.

Related Information

Cortex-A9 MPCore on page 9-4
For more information, refer to *Cortex-A9 MPU Subsystem*.

FPGA-to-HPS Interrupts

You can configure the HPS component to provide 64 general-purpose FPGA-to-HPS interrupts, allowing soft IP in the FPGA fabric to trigger interrupts to the MPU's generic interrupt controller (GIC). The interrupts are implemented through the following 32-bit interfaces:

- `f2h_irq0`—FPGA-to-HPS interrupts 0 through 31
- `f2h_irq1`—FPGA-to-HPS interrupts 32 through 63

The FPGA-to-HPS interrupts are asynchronous on the FPGA interface. Inside the HPS, the interrupts are synchronized to the MPU's internal peripheral clock (`periphclk`).

Related Information

HPS Component Interfaces on page 28-1
There are various HPS peripherals to FPGA interrupt interfaces that you can configure. To learn the complete list of interfaces, refer to the *Instantiating the HPS Component* chapter.

Boot from FPGA Interface

You can enable the boot from FPGA interface to indicate the availability of preloader software in the FPGA memory. This interface is used to indicate the availability of a fallback preloader software in FPGA memory as well. The fallback preloader is used when there is no valid preloader image found in HPS flash memories.

Input-only General Purpose Interface

You can enable a general purpose interface that is connected to the general purpose I/O (GPIO) peripheral of the HPS. This is an input-only interface with 14-bit wide width. The interface share the same I/O pins with the HPS DDR SDRAM controller.

Document Revision History

Table 28-5: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Added address map and register descriptions
February 2014	2014.02.28	Added in new sections • Peripheral FPGA Clocks • Peripheral Reset Interfaces • Boot from FPGA Interface • Input-only General Purpose Interfaces Removed section • General Purpose Interfaces Updated sections • Trace Port Interface Unit • Peripheral Signal Interfaces • FPGA-to-HPS Interrupts
December 2013	2013.12.30	Minor formatting issues
November 2012	1.1	• Added debug interfaces. • Updated HPS-to-FPGA reset interface names. • Updated HPS external reset source interface names. • Removed DMA peripheral interface clocks. • Referred to Altera Address Span Extender.
June 2012	1.0	Initial release.
May 2012	0.1	Preliminary draft.

HPS Simulation Support 29

2014.06.30

cv_54030 Subscribe Send Feedback

This section describes the simulation support for the hard processor system (HPS) component. The HPS simulation models support interfaces between the HPS and FPGA fabric, including:

- Bus functional models (BFMs) for most interfaces between HPS and FPGA fabric
- A simulation model for the HPS SDRAM memory

The HPS simulation support does not include modules implemented in the HPS, such as the ARM Cortex-A9 MPCore processor.

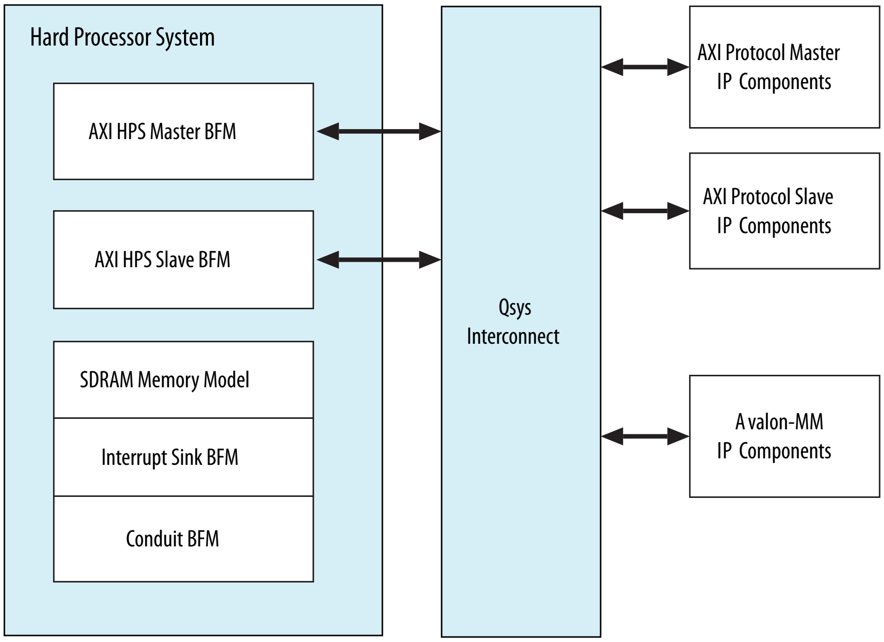
You specify simulation support files when you instantiate the HPS component in the Qsys system integration tool. When you enable a particular HPS-FPGA interface, Qsys provides the corresponding model during the generation process.

The HPS simulation support enables you to develop and verify your own FPGA soft logic or intellectual property (IP) that interfaces to the HPS component.

The simulation model supports the following interfaces:

- Clock and reset interfaces
- FPGA-to-HPS Advanced Microcontroller Bus Architecture (AMBA) Advanced eXtensible Interface (AXI) slave interface
- HPS-to-FPGA AXI master interface
- Lightweight HPS-to-FPGA AXI master interface
- FPGA-to-HPS SDRAM interface
- Microprocessor unit (MPU) general-purpose I/O interface
- MPU standby and event interface
- Interrupts interface
- Direct memory access (DMA) controller peripheral request interface
- Debug Advanced Peripheral Bus (APB) interface
- System Trace Macrocell (STM) hardware event
- FPGA cross trigger interface
- FPGA trace port interface
- Boot from FPGA interface
- Input only general purpose interface

Figure 29-1: HPS BFM Block Diagram



The HPS BFMs use standard function calls from the Altera BFM application programming interface (API), as detailed in the remainder of this section.

HPS simulation supports only Verilog HDL or SystemVerilog simulation environments.

Related Information

- **Simulation Flows** on page 29-10
- **Instantiating the HPS Component** on page 27-1
- **Avalon Verification IP Suite User Guide**
  For information about the BFM API.
- **Mentor Verification IP Altera Edition User Guide**
  For information about the BFM API.

Clock and Reset Interfaces

Related Information

Memory-Mapped Interfaces on page 28-1

Clock Interface

Qsys generates the clock source BFM for each clock output interface from the HPS component. For HPS-to-FPGA user clocks, specify the BFM clock rate in the **User clock frequency field** in the **HPS Clocks** page when instantiating the HPS component in Qsys.

The HPS-to-FPGA trace port interface unit generates a clock output to the FPGA, named `h2f_tpiu_clock`. In simulation, the clock source BFM also represents this clock output's behavior.

Table 29-1: HPS Clock Output Interface Simulation Model

The Altera clock source BFM application programming interface (API) applies to all the BFMs listed in this table. Your Verilog interfaces use the same API, passing in different instance names.

Interface Name	BFM Instance Name
h2f_user0_clock	h2f_user0_clock_inst
h2f_user1_clock	h2f_user1_clock_inst
h2f_user2_clock	h2f_user2_clock_inst
h2f_tpiu_clock	h2f_tpiu_clock_inst

Qsys does not generate BFMs for FPGA-to-HPS clock input interfaces.

Reset Interface

The HPS reset request and handshake interfaces are connected to Altera conduit BFMs for simulation.

Table 29-2: HPS Reset Input Interface Simulation Model

You can monitor the reset request interface state changes or set the interface by using the API listed.

Interface Name	BFM Instance Name	API Function Names
f2h_cold_reset_req	f2h_cold_reset_req_inst	get_f2h_cold_rst_req_n()
f2h_debug_reset_req	f2h_debug_reset_req_inst	get_f2h_dbg_rst_req_n()
f2h_warm_reset_req	f2h_warm_reset_req_inst	get_f2h_warm_rst_req_n()

h2f_warm_reset_handshake	h2f_warm_reset_handshake_inst	set_h2f_pending_rst_req_n()
		get_f2h_pending_rst_ack_n()

Table 29-3: HPS Reset Output Interface Simulation Model

The Altera reset source BFM application programming interface applies to all the BFMs listed.

Interface Name	BFM Instance Name
h2f_reset	h2f_reset_inst
h2f_cold_reset	h2f_cold_reset_inst
h2f_debug_apb_reset	h2f_debug_apb_reset_inst

Table 29-4: Configuration of Reset Source BFM for HPS Reset Output Interface

The HPS reset output interface is connected to a reset source BFM. Qsys configures the BFM as shown. The parameter value of the instantiated BFM as configured for HPS simulation.

Parameter	BFM Value	Meaning
Assert reset high	Off	This parameter is off, specifying an active-low reset signal from the BFM.
Cycles of initial reset	0	This parameter is 0, specifying that the BFM does not assert the reset signal automatically.

FPGA-to-HPS AXI Slave Interface

The FPGA-to-HPS AXI slave interface, `f2h_axi_slave`, is connected to a Mentor Graphics AXI slave BFM for simulation with an instance name of `f2h_axi_slave_inst`. Qsys configures the BFM as shown in the following table. The BFM clock input is connected to `f2h_axi_clock` clock.

Table 29-5: Configuration of FPGA-to-HPS AXI Slave BFM

Parameter	Value
AXI Address Width	32
AXI Read Data Width	32, 64, 128
AXI Write Data Width	32, 64, 128
AXI ID Width	8

You control and monitor the AXI slave BFM by using the BFM API.

Related Information

- **Memory-Mapped Interfaces** on page 28-1
- **Mentor Verification IP Altera Edition User Guide**

HPS-to-FPGA AXI Master Interface

The HPS-to-FPGA AXI master interface, `h2f_axi_master`, is connected to a Mentor Graphics AXI master BFM for simulation with an instance name of `h2f_axi_master_inst`. Qsys configures the BFM as shown in the following table. The BFM clock input is connected to `h2f_axi_clock` clock.

Table 29-6: Configuration of HPS-to-FPGA AXI Master BFM

Parameter	Value
AXI Address Width	30

Parameter	Value
AXI Read and Write Data Width	32, 64, 128
AXI ID Width	12

You control and monitor the AXI master BFM by using the BFM API.

Related Information

- **Memory-Mapped Interfaces** on page 28-1
- **Mentor Verification IP Altera Edition User Guide**

Lightweight HPS-to-FPGA AXI Master Interface

The lightweight HPS-to-FPGA AXI master interface, `h2f_lw_axi_master`, is connected to a Mentor Graphics AXI master BFM for simulation with an instance name of `h2f_lw_axi_master_inst`. Qsys configures the BFM as shown in the following table. The BFM clock input is connected to `h2f_lw_axi_clock` clock.

Table 29-7: Configuration of Lightweight HPS-to-FPGA AXI Master BFM

Parameter	Value
AXI Address Width	21
AXI Read and Write Data Width	32
AXI ID Width	12

You control and monitor the AXI master BFM by using the BFM API.

Related Information

- **Memory-Mapped Interfaces** on page 28-1
- **Mentor Verification IP Altera Edition User Guide**

FPGA-to-HPS SDRAM Interface

The HPS component contains a memory interface simulation model to which all of the FPGA-to-HPS SDRAM interfaces are connected. The model is based on the HPS implementation and provides cycle-level accuracy, reflecting the true bandwidth and latency of the interface. However, the model does not have the detailed configuration provided by the HPS software, and hence does not reflect any inter-port scheduling that might occur under contention on the real hardware when different priorities or weights are used.

Related Information

Functional Description—Hard Memory Interface

For more information, refer to "EMI-Related HPS Features in SoC Devices" in the *Functional Description —Hard Memory Interface* chapter in volume 3 of the *External Memory Interface Handbook*.

HPS-to-FPGA MPU General Purpose I/O Interface

The HPS-to-FPGA MPU general-purpose I/O interface is connected to an Altera conduit BFM for simulation. The following table lists the name of each interface, along with API function names for each type of simulation. You can monitor the interface state changes or set the interface by using the API listed.

Table 29-8: HPS-to-FPGA MPU General Purpose I/O Interface Simulation Model

Interface Name	BFM Instance Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

h2f_mpu_gp	h2f_mpu_gp_inst	set_h2f_mpu_gp_out()
		get_h2f_mpu_gp_in()
			set_gp_out()
			get_gp_in()

HPS-to-FPGA MPU Event Interface

The HPS-to-FPGA MPU event interface is connected to an Altera conduit BFM for simulation. The following table lists the name of each interface, along with API function names for each type of simulation. You can monitor the interface state changes or set the interface by using the API listed.

Table 29-9: HPS-to-FPGA MPU Event Interface Simulation Model

The usage of conduit `get_*()` and `set_*()` API functions is the same as with the general Avalon conduit BFM.

Interface Name	BFM Instance Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

h2f_mpu_events	h2f_mpu_events_inst	get_h2f_mpu_eventi()
		set_h2f_mpu_evento()
		set_h2f_mpu_standbywfe()
		set_h2f_mpu_standbywfi()
			get_eventi()
			set_evento()
			set_standbywfe()
			set_standbywfi()

FPGA-to-HPS Interrupts Interface

The FPGA-to-HPS interrupts interface is connected to an Altera Avalon interrupt sink BFM for simulation.

Table 29-10: FPGA-to-HPS Interrupts Interface Simulation Model

Interface Name	BFM Instance Name
f2h_irq0	f2h_irq0_inst
f2h_irq1	f2h_irq1_inst

Note: The simulation support for HPS peripherals to FPGA interfaces are not supported at the moment.

Related Information

Reset Interface on page 29-3
The Altera Avalon interrupt sink BFM API applies to all the BFMs listed in the HPS Reset Output Interface Simulation Model.

HPS-to-FPGA Debug APB Interface

The HPS-to-FPGA debug APB interface simulation is not supported at the moment. Updates on this subject will be provided in upcoming versions of this handbook.

Related Information

Product Support
For more information, refer to the KDB solution on the Altera product support solution page.

FPGA-to-HPS System Trace Macrocell (STM) Hardware Event Interface

The FPGA-to-HPS STM hardware event interface is connected to an Altera conduit BFM for simulation. The following table lists the name of each interface, along with API function name for each type of simulation. You can monitor the interface state changes or set the interface by using the API functions listed.

Table 29-12: FPGA-to-HPS STM Hardware Event Interface Simulation Model

Interface Name	BFM Name	RTL Simulation API Function Name	Post-Fit Simulation API Function Name
f2h_stm_hw_events	f2h_stm_hw_events_inst	get_f2h_stm_hwevents()	get_stm_events()

HPS-to-FPGA Cross-Trigger Interface

The HPS-to-FPGA cross-trigger interface is connected to an Altera conduit BFM for simulation. The following table lists the name of each interface, along with API function names for each type of simulation. You can monitor the interface state changes or set the interface by using the API functions listed.

Table 29-13: HPS-to-FPGA Cross-Trigger Interface Simulation Model

Interface Name	BFM Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

h2f_cti	h2f_cti_inst	get_h2f_cti_trig_in()
		set_h2f_cti_trig_in_ack()
		set_h2f_cti_trig_out()
		get_h2f_cti_trig_out_ack()
		set_h2f_cti_asicctl()
		get_h2f_cti_fpga_clk_en()
			get_trig_in()
			set_trig_inack()
			set_trig_out()
			get_trig_outack()
			set_asicctl()
			get_clk_en()

HPS-to-FPGA Trace Port Interface

The HPS-to-FPGA trace port interface is connected to an Altera conduit BFM for simulation. The following table lists the name of each interface, along with API function names for each type of simulation. You can monitor the interface state changes or set the interface by using the API functions listed.

Table 29-14: HPS-to-FPGA Trace Port Interface Simulation Model

Interface Name	BFM Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

h2f_tpiu	h2f_tpiu_inst	get_h2f_tpiu_clk_ctl()
		set_h2f_tpiu_data()
			get_traceclk_ctl()
			set_trace_data()

FPGA-to-HPS DMA Handshake Interface

The FPGA-to-HPS DMA handshake interface is connected to an Altera conduit BFM for simulation. The following table lists the name for each interface, along with API function names for each type of simulation. You can monitor the interface state changes or set the interface by using the API listed.

Table 29-15: FPGA-to-HPS DMA Handshake Interface Simulation Model

The usage of conduit `get_*()` and `set_*()` API functions is the same as with the general Avalon conduit BFM.

Interface Name	BFM Instance Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

f2h_dma_req0	f2h_dma_req0_inst	get_f2h_dma_req0_req()
		get_f2h_dma_req0_single()
		set_f2h_dma_req0_ack()
			get_channel0_req()
			get_channel0_single()
			set_channel0_xx_ack()

Interface Name	BFM Instance Name	RTL Simulation API Function Names	Post-Fit Simulation API Function Names

f2h_dma_req1	f2h_dma_req1_inst	get_f2h_dma_req1_req()
		get_f2h_dma_req1_single()
		set_f2h_dma_req1_ack()
			get_channel1_req()
			get_channel1_single()
			set_channel1_xx_ack()

f2h_dma_req2	f2h_dma_req2_inst	get_f2h_dma_req2_req()
		get_f2h_dma_req2_single()
		set_f2h_dma_req2_ack()
			get_channel2_req()
			get_channel2_single()
			set_channel2_xx_ack()

f2h_dma_req3	f2h_dma_req3_inst	get_f2h_dma_req3_req()
		get_f2h_dma_req3_single()
		set_f2h_dma_req3_ack()
			get_channel3_req()
			get_channel3_single()
			set_channel3_xx_ack()

f2h_dma_req4	f2h_dma_req4_inst	get_f2h_dma_req4_req()
		get_f2h_dma_req4_single()
		set_f2h_dma_req4_ack()
			get_channel4_req()
			get_channel4_single()
			set_channel4_xx_ack()

f2h_dma_req5	f2h_dma_req5_inst	get_f2h_dma_req5_req()
		get_f2h_dma_req5_single()
		set_f2h_dma_req5_ack()
			get_channel5_req()
			get_channel5_single()
			set_channel5_xx_ack()

f2h_dma_req6	f2h_dma_req6_inst	get_f2h_dma_req6_req()
		get_f2h_dma_req6_single()
		set_f2h_dma_req6_ack()
			get_channel6_req()
			get_channel6_single()
			set_channel6_xx_ack()

f2h_dma_req7	f2h_dma_req7_inst	get_f2h_dma_req7_req()
		get_f2h_dma_req7_single()
		set_f2h_dma_req7_ack()
			get_channel7_req()
			get_channel7_single()
			set_channel7_xx_ack()

Boot from FPGA Interface

The boot from FPGA interface is connected to an Altera conduit BFM for simulation. You can monitor the interface state changes or set the interface by using the API functions in the table below.

Table 29-16: Boot from FPGA Interface Simulation Model

Interface Name	BFM Name	RTL simulation API function names	Post-fit simulation API function names

f2h_boot_from_fpga	f2h_boot_from_fpga_inst	get_f2h_boot_from_fpga_ready()
		get_f2h_boot_from_fpga_on_failure()
			get_boot_fpga_ready()
			get_boot_from_fpga_on_failure()

General Purpose Input (GPI) Interface

The general purpose input interface is connected to an Altera conduit BFM for simulation. You can monitor the interface state changes or set the interface by using the API functions in the table below.

Table 29-17: General Purpose Input Interface Simulation Model

Interface Name	BFM Name	RTL simulation API function names

hps_io	hps_io_inst	get_hps_io_gpio_inst_HLGPI0()
		get_hps_io_gpio_inst_HLGPI1()
		get_hps_io_gpio_inst_HLGPI2()
		get_hps_io_gpio_inst_HLGPI3()
		get_hps_io_gpio_inst_HLGPI4()
		get_hps_io_gpio_inst_HLGPI5()
		get_hps_io_gpio_inst_HLGPI6()
		get_hps_io_gpio_inst_HLGPI7()
		get_hps_io_gpio_inst_HLGPI8()
		get_hps_io_gpio_inst_HLGPI9()
		get_hps_io_gpio_inst_HLGPI10()
		get_hps_io_gpio_inst_HLGPI11()
		get_hps_io_gpio_inst_HLGPI12()
		get_hps_io_gpio_inst_HLGPI13()

Simulation Flows

This section describes the simulation flows for an HPS-based design.

Altera provides both functional register transfer level (RTL) simulation and post-fitter gate-level simulation flows. The simulation flows involve the following major steps, which are based on Altera Complete Design Suite (ACDS 14.0):

1. **Specifying HPS Simulation Model in Qsys** on page 29-11
2. **Generating HPS Simulation Model in Qsys** on page 29-13
3. **Running the Simulation** on page 29-13

Related Information

Simulating Altera Designs

For general information about simulation, refer to the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Specifying HPS Simulation Model in Qsys

The following steps outline how to set up the HPS component for simulation.

1. Add the HPS component from the Qsys Component Library.
2. Configure the component based on your application needs by selecting or deselecting the HPS-FPGA interfaces.
3. Connect the appropriate HPS interfaces to other components in the system. For example, connect the FPGA-to-HPS AXI slave interface to an AXI master interface in another component in the system.

When you create your component, make sure the conduit interfaces have the correct role names, widths and are opposite in direction to what is shown in the HPS conduit Interfaces table.

HPS Conduit Interfaces on page 29-11

Related Information

Instantiating the HPS Component on page 27-1

HPS Conduit Interfaces

Table 29-18: HPS Conduit Interfaces

Role Name	Direction	Width
h2f_warm_reset_handshake		
`h2f_pending_rst_req_n`	Output	1
`f2h_pending_rst_ack_n`	Input	1
h2f_gp		
`gp_in`	Input	32
`gp_out`	Output	32
h2f_mpu_events		

Role Name	Direction	Width
eventi	Input	1
evento	Output	1
standbywfe	Output	2
standbywfi	Output	2
f2h_dma_req0 to f2h_dma_req7		
dma_req	Input	1
dma_single	Input	1
dma_ack	Output	1
h2f_debug_apb_sideband		
pclken	Input	1
dbg_apb_disable	Input	1
f2h_stm_hw_events		
stm_hwevents	Input	28
h2f_cti		
trig_in	Input	8
trig_in_ack	Output	8
trig_out	Output	8
trig_out_ack	Input	8
asicctl	Output	8
fpga_clk_en	Input	1
h2f_tpiu		
clk_ctl	Input	1
data	Output	32
f2h_boot_from_fpga		

Role Name	Direction	Width
`boot_from_fpga_ready`	Input	1
`boot_from_fpga_on_failure`	Input	1

Generating HPS Simulation Model in Qsys

The following steps outline how to generate the simulation model.

1. In Qsys, click **Generate HDL** under the Generate menu.
2. For RTL simulation, perform the following steps:

   - Set **Create simulation model** to **Verilog**.
   - Click **Generate**.

   Note: HPS simulation does not support the VHDL simulation environment.

   For post-fit simulation, perform the following steps:

   - Turn on the **Create HDL design files for synthesis** option.
   - Turn on the **Create block symbol file (.bsf)** option.
   - Click **Generate**.

Related Information

- **Instantiating the HPS Component** on page 27-1
- **Creating a System with Qsys**
  For more information about Qsys simulation, refer to “Simulating a Qsys System” in the *Creating a System with Qsys* chapter in volume 1 of the *Quartus II Handbook*.

Running the Simulation

Refer to one of the following.

Running HPS RTL Simulation on page 29-13

Running HPS Post-Fit Simulation on page 29-14

Running HPS RTL Simulation

Qsys generates scripts for various simulators that you use to complete the simulation process.

Table 29-19: Qsys-Generated Scripts for Various Simulators

Simulator	Script Name	Directory
Mentor Graphics Modelsim® Altera Edition	**msim_setup.tcl**	*<project directory>/<Qsys design name>*/**simulation/mentor**
Cadence NC-Sim	**ncsim_setup.sh**	*<project directory>/<Qsys design name>*/**simulation/cadence**

Simulator	Script Name	Directory
Synopsys VCS	**vcs_setup.sh**	*<project directory>/<Qsys design name>*/**simulation/synopsys/vcs**
Synopsys VCS-MX	**vcsmx_setup.sh**	*<project directory>/<Qsys design name>*/**simulation/synopsys/vcsmx**
Aldec RivieraPro	**rivierapro_setup.tcl**	*<project directory>/<Qsys design name>*/**simulation/aldec**

Related Information

- **Avalon Verification IP Suite User Guide**
  For information about the BFM API.
- **Mentor Verification IP Altera Edition User Guide**
  For information about the BFM API.

Running HPS Post-Fit Simulation

The section describes how you run HPS post-fit simulation. After successful Qsys generation, perform the following steps:

1. Add the Qsys-generated synthesis file set to your Quartus II project by performing the following steps:
   a. In the Quartus II software, click **Settings** in the Assignments menu.
   b. In the **Settings** *<your Qsys system name>* dialog box, on the **Files** tab, browse to *<your project directory>/<your Qsys system name>*/**synthesis/** and select *<your Qsys system name>*.**qip**.
   c. Click **OK**.
2. You can instantiate the Qsys system that contains HPS component as your Quartus II project top-level entity, if necessary.
3. Compile the design by clicking **Start Compilation** in the Processing menu.
4. Change the EDA Netlist Writer settings, if necessary, by performing the following steps:
   a. Click **Settings** in the Assignment menu.
   b. On the **Simulation** tab, under the **EDA Tool Settings** tab, you can specify the following EDA Netlist Writer settings:
      - **Tool name**—The name of the simulation tool
      - **Format for output netlist**
      - **Output directory**
   c. Click **OK**.
5. To create the post-fitter simulation model with Quartus II EDA Netlist Writer, in the Start menu, point to **Processing** and click **Start EDA Netlist Writer**.
6. Compile the necessary simulation files in your simulation tool.
7. Start simulation.

Related Information

Simulation Page (Settings Dialog Box)

Post-Fit Simulation Files

Table 29-20: Post-Fit Simulation Files

This table uses the following symbols:

- *<ACDS install>* = Altera Complete Design Suite installation path
- *<Avalon Verification IP>* = *<ACDS install>***/ip/altera/sopc_builder_ip/verification**
- *<AXI Verification IP>* = *<ACDS install>***/ip/altera/mentor_vip_ae**
- *<HPS Post-fit Sim>* = *<ACDS install>***/ip/altera/hps/postfitter_simulation**
- *<Device Sim Lib>* = *<ACDS install>***/quartus/eda/sim_lib**

Library	Directory	File
Altera Verification IP Library	*<Avalon Verification IP>***/lib/**	**verbosity_pkg.sv** **avalon_mm_pkg.sv** **avalon_utilities_pkg.sv**
Avalon Clock Source BFM	*<Avalon Verification IP>*/ **altera_avalon_clock_source/**	**altera_avalon_clock_source.sv**
Avalon Reset Source BFM	*<Avalon Verification IP>*/ **altera_avalon_reset_source/**	**altera_avalon_reset_source.sv**
Avalon MM Slave BFM	*<Avalon Verification IP>*/ **altera_avalon_mm_slave_bfm/**	**altera_avalon_mm_slave_bfm.sv**
Avalon Interrupt Sink BFM	*<Avalon Verification IP>*/ **altera_avalon_interrupt_sink/**	**altera_avalon_interrupt_sink.sv**
Mentor AXI Verification IP Library	*<AXI Verification IP>***/common/**	**questa_mvc_svapi.svh**
Mentor AXI3 BFM	*<AXI Verification IP>***/axi3/axi3/bfm/**	**mgc_common_axi.sv** **mgc_axi_master.sv** **mgc_axi_slave.sv**
HPS Post-Fit Simulation Library	*<HPS Post-fit Sim>/*	**All the files in the directory**

Library	Directory	File
Device Simulation Library (The device simulation library is not needed with Modelsim-Altera.)	*<Device Sim Lib>/*	**altera_primitives.v** **220model.v** **sgate.v** **altera_mf.v** **altera_lnsim.sv** **cyclonev_atoms.v** **arriav_atoms.v** **mentor/cyclonev_atoms_ncrypt.v** **mentor/arriav_atoms_ncrypt.v**
EDA Netlist Writer Generated Post-Fit Simulation Model	*<User project directory>/*	***.vo** ***.vho** (Mixed-language simulator is needed for Verilog HDL and VHDL mixed design)
User testbench files	*<User project directory>/*	***.v** ***.sv** ***.vhd** (Mixed-language simulator is needed for Verilog HDL and VHDL mixed design)

BFM API hierarchy Format

For post-fit simulation, you must call the BFM API in your test program with a specific hierarchy. The hierarchy format is:

```
<DUT>.\<HPS>|fpga_interfaces|<interface><space>.<BFM>.<API function>
```

Where:

- *<DUT>* is the instance name of the design under test that you instantiated in your test bench that consists of the HPS component.
- *<HPS>* is the HPS component instance name that you use in your Qsys system.
- *<interface>* is the instance name for a specific FPGA-to-HPS or HPS-to-FPGA interface. This name can be found in the **fpga_interfaces.sv** file located in *<project directory>*/*<Qsys design name>*/**synthesis/submodules**.
- *<space>*—You must insert one space character after the interface instance name.
- *<BFM>* is the BFM instance name. In *<ACDS install>*/**ip/altera/hps/postfitter_simulation**, identify the SystemVerilog file corresponding to the interface type that you are using. The SystemVerilog file contains the BFM instance name.

For example, a path for the Lightweight HPS-to-FPGA master interface hierarchy could be formed as follows:

```
top.dut.\my_hps_component|fpga_interface|hps2fpga_light_weight .h2f_lw_axi_master
```

Notice the space after "hps2fpga_light_weight". Omitting this space would cause simulation failure because the instance name "hps2fpga_light_weight ", including the space, is the name used in the post-fit simulation model generated by the Quartus® II software.

Document Revision History

Table 29-21: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	• Updated the Simulation Flows section. • Updated the Generating HPS Simulation Model in Qsys section.
February 2014	2014.02.28	Added new sections: • Boot from FPGA Interface • General Purpose Input (GPI) Interface Updated content in sections: • Specifying HPS Simulation Model in Qsys • Running HPS RTL Simulation
December 2013	2013.12.30	Maintenance release
November 2012	1.1	• Added debug APB, STM hardware event, FPGA cross trigger, FPGA trace port interfaces. • Added support for post-fit simulation. • Updated some API function names. • Removed DMA peripheral clock.
June 2012	1.0	Initial release.
May 2012	0.1	Preliminary draft.

Booting and Configuration Introduction A

2014.06.30

cv_5400a Subscribe Send Feedback

This topic describes the booting of the hard processor system (HPS) and the configuration of the FPGA portion of the Altera system-on-a-chip (SoC) device.

The HPS boot starts when a processor is released from reset (for example, on power up) and executes code in the internal boot ROM at the reset exception address. The boot process ends when the code in the boot ROM jumps to the next stage of the boot software. This next stage of boot software is referred to as the preloader. The preloader can be customized and is typically stored external to the HPS in a nonvolatile flash-based memory.

The processor can boot from the following sources:

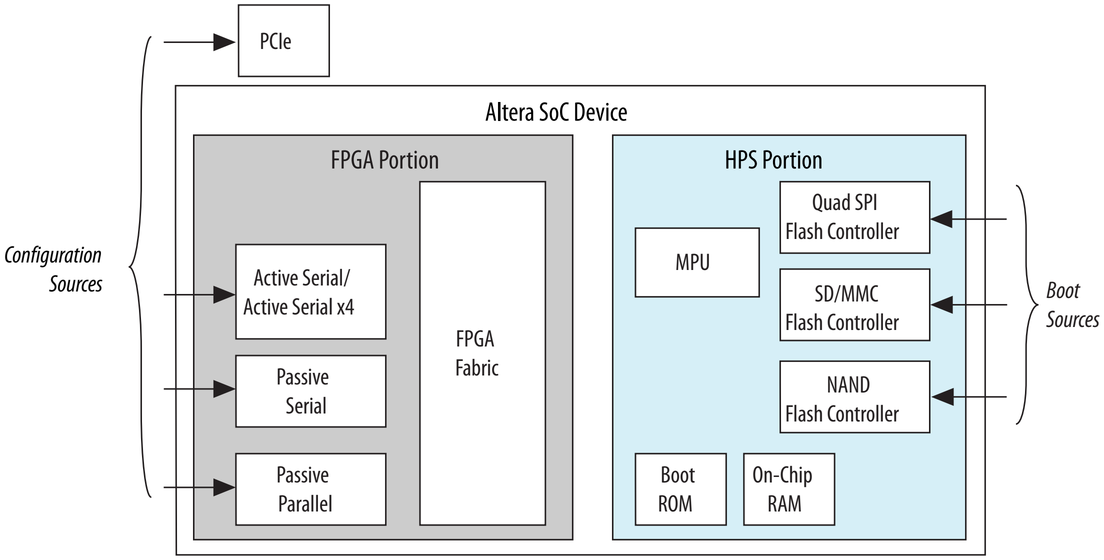
- NAND flash memory through the NAND flash controller
- Secure Digital/MultiMediaCard (SD/MMC) flash memory through the SD/MMC flash controller
- Serial peripheral interface (SPI) and quad SPI flash memory through the quad SPI flash controller using Slave Select 0
- FPGA fabric

The HPS boot supports indirect or direct execution of the preloader depending on the boot device. With indirect execution, the boot ROM code copies the preloader from the boot device into the on-chip RAM and jumps to it. Indirect execution is used for flash memory boot sources. With direct execution, the boot ROM code jumps to the preloader located in the FPGA fabric.

Configuration of the FPGA portion of the device starts when the FPGA portion is released from the reset state (for example, on power-on). The control block (CB) in the FPGA portion of the device is responsible for obtaining an FPGA configuration image and configuring the FPGA. The FPGA configuration ends when the configuration image has been fully loaded and the FPGA enters user mode. The FPGA configuration image is provided by users and is typically stored in non-volatile flash-based memory. The FPGA CB can obtain a configuration image from the HPS through the FPGA manager or from any of the sources supported by the Cyclone V FPGAs family.

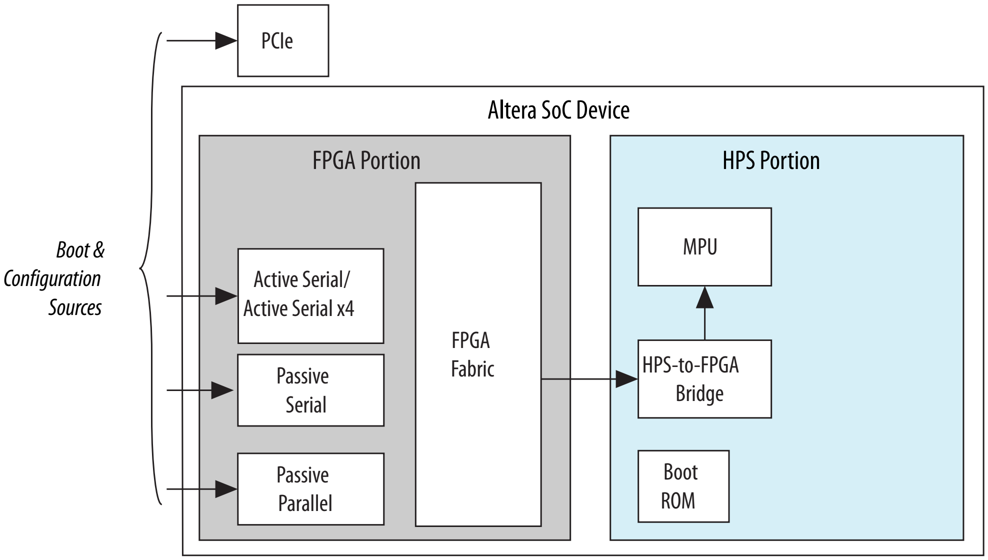
The following three figures illustrate the possible HPS boot and FPGA configuration schemes. The arrows in the figures denote the data flow direction. The following figure shows that the FPGA configuration and HPS boot occur independently. The FPGA configuration obtains its configuration image from a non-HPS source, while the HPS boot obtains its preloader from a non-FPGA fabric source.

Figure A-1: Independent FPGA Configuration and HPS Booting



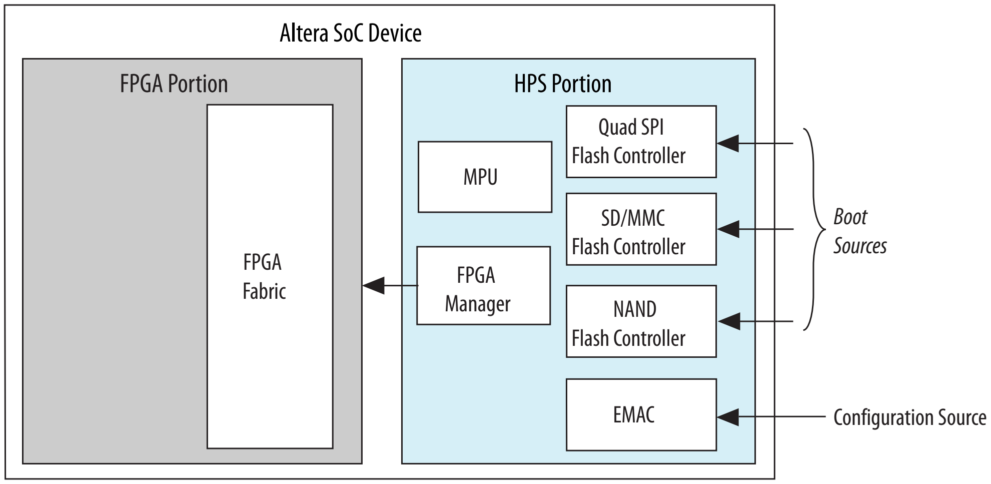
The following figure shows that the FPGA is first configured through one of its non-HPS configuration sources and then the HPS boots from the FPGA fabric. The HPS boot waits for the FPGA fabric to be powered on and in user mode before executing. The HPS boot ROM code executes the preloader from the FPGA fabric over the HPS-to-FPGA bridge. The preloader can be obtained from the FPGA RAM or by accessing an external interface, depending on your design and implementation.

Figure A-2: FPGA Configures First



The following figure shows that the HPS boots first through one of its non-FPGA fabric boot sources and then software running on the HPS configures the FPGA fabric through the HPS FPGA manager. The software on the HPS obtains the FPGA configuration image from any of its flash memory devices or communication interfaces, for example, the Ethernet media access controller (EMAC). The software is provided by users and the boot ROM is not involved in configuring the FPGA fabric.

Figure A-3: HPS Boots First



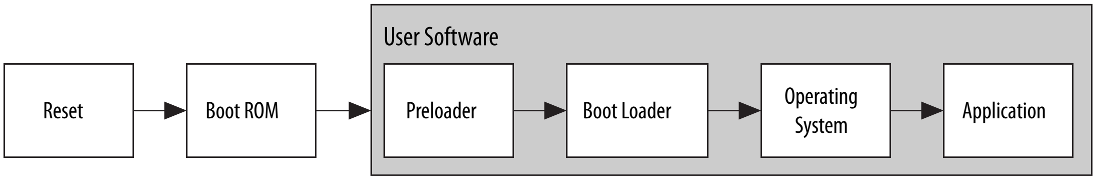
HPS Boot

Booting software on the HPS is a multi-stage process. Each stage is responsible for loading the next stage. The first software stage is the boot ROM. The boot ROM code locates and executes the second software stage, called the preloader. The preloader locates, and if present, executes the next software stage. The preloader and subsequent software stages (if present) are collectively referred to as user software.

Only the boot ROM code is located in the HPS. Subsequent software is located external to the HPS and is provided by users. The boot ROM code is only aware of the preloader and not aware of any potential subsequent software stages.

The figure below illustrates the typical boot flow. However, there may be more or less software stages in the user software than shown and the roles of the software stages may vary.

Figure A-4: Typical Boot Flow



Boot Process Overview

Reset

The boot process begins when a CPU in the MPU exits from the reset state. When a CPU exits from reset, it starts running code at the reset exception address. In normal operation, the boot ROM is mapped at the reset exception address so code starts running in the boot ROM.

It is possible to map the on-chip RAM or SDRAM at the reset exception address and run code other than the boot ROM code. However, this chapter assumes that the boot ROM maps to the reset exception address.

Boot ROM

The boot ROM contains software code that executes after a reset exits. If running on CPU0, the boot ROM code reads the boot select (BOOTSEL) and clock select (CLKSEL) values from the bsel and csel fields of the boot information register (boot info) in the system manager. This is done to determine the boot source and to set up the clock manager. The bsel and csel field values come from the BOOTSEL and CLKSEL pins which are sampled by the system manager coming out of reset.

The user software in CPU0 is responsible to release CPU1 from reset. If CPU1 is released from reset with the boot ROM mapped to the reset exception address, the boot ROM code jumps to the value in the CPU1 start address (`cpu1startaddr`) in the boot ROM code register group (`romcodegrp`) in the system manager.

BOOTSEL Field Values and Flash Device Selection

Table A-1: BOOTSEL Field Values and Flash Device Selection

BOOTSEL Field Value	Flash Device
0x0	Reserved
0x1	FPGA (HPS-to-FPGA bridge)
0x2	1.8 V NAND flash memory
0x3	3.3 V NAND flash memory
0x4	1.8 V SD/MMC flash memory with external transceiver
0x5	3.3 V SD/MMC flash memory with internal transceiver
0x6	1.8 V SPI or quad SPI flash memory
0x7	3.3 V SPI or quad SPI flash memory

Note:

If the BOOTSEL value is set 0x1 to Boot from FPGA, then the CLKSEL values are ignored. PLLs are bypassed so that OSC1 drives all the clocks.

For indirect execution from flash memory boot sources, the boot ROM code loads the preloader image from the flash device to the on-chip RAM and passes software control to the preloader in the on-chip RAM. For direct execution from the FPGA fabric boot source, the boot ROM code waits until the FPGA portion of the device is in user mode, and is ready to execute code and then passes software control to the preloader in the FPGA RAM.

Preloader

The function of the preloader is user-defined. However, typical functions include initializing the SDRAM interface and configuring the HPS I/O pins. Initializing the SDRAM allows the preloader to load the next stage of the boot software (that might not fit in the 60 kilobytes (KB) available in the on-chip RAM). A typical next software stage is the open source boot loader, U-boot.

The preloader is allowed to load the next boot software stage from any device available to the HPS. Typical sources include the same flash device that contains the preloader, a different flash device, or a communication interface such as an EMAC.

Boot Loader

The boot loader loads the operating system and passes software control to the operating system.

Boot ROM

The function of the boot ROM code is to determine the boot source, initialize the HPS after a reset, and jump to the preloader. In the case of indirect execution, the boot ROM code loads the preloader image from the flash memory to on-chip RAM. The boot ROM performs the following actions to initialize the HPS:

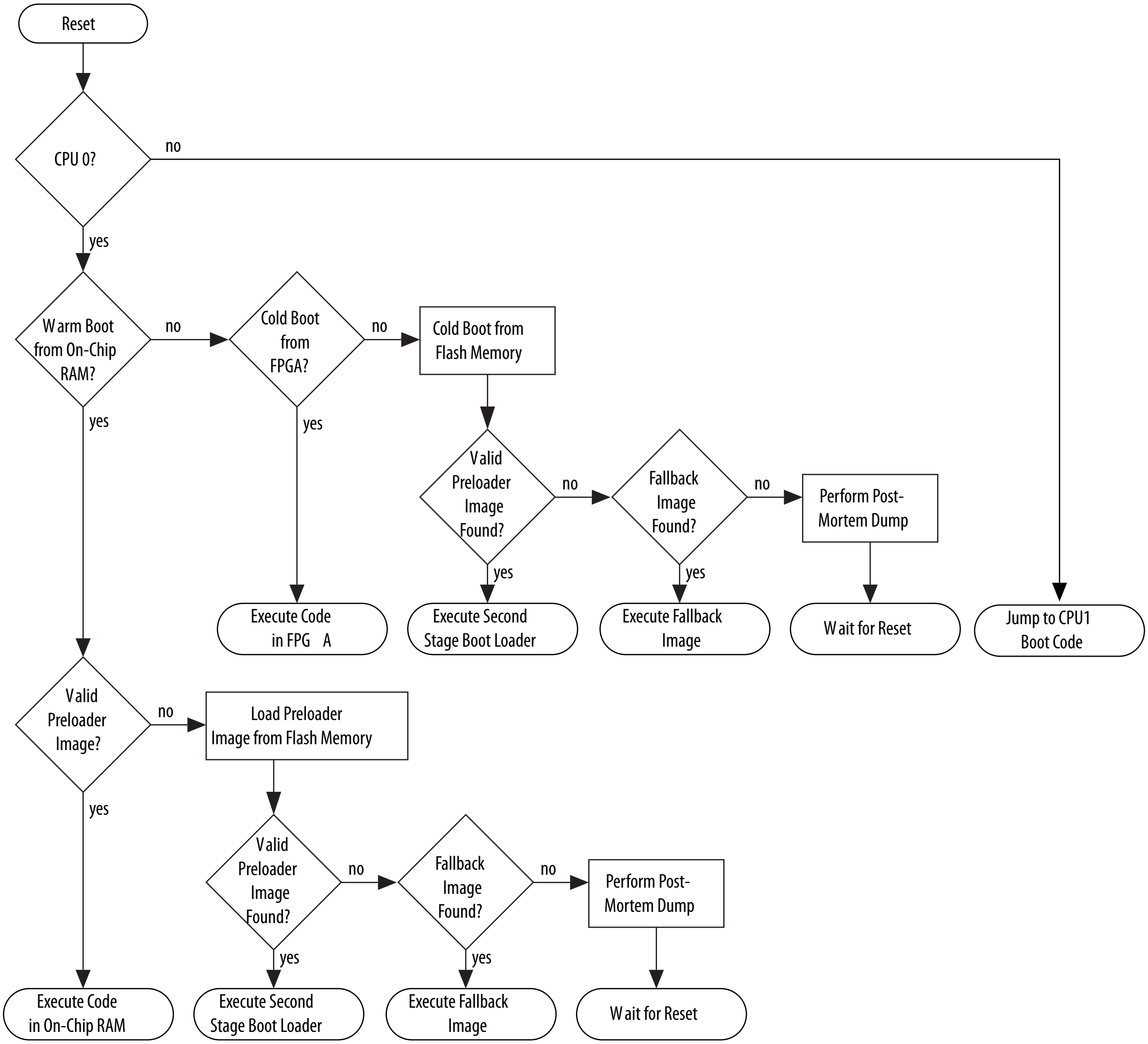
- Enable instruction cache, branch predictor, floating point unit, NEON vector unit
- Sets up the level 4 (l4) watchdog 0 timer
- Configures the main PLL and peripheral PLL based on the CLKSEL value
- Configures I/O elements and pin multiplexing based on the BOOTSEL value
- Initializes the flash controller to default settings

When booting from flash memory, the boot ROM code uses the top 4 KB of the on-chip RAM as data workspace. This area is reserved for the boot ROM code after a reset until the boot ROM code passes software control to preloader. This limits the maximum size of the preloader for indirect execution to 60 KB. For a warm boot or cold boot from FPGA, the boot ROM code does not reserve the top 4KB of the on-chip RAM, and the user may place user data in this area without being overwritten by boot ROM.

Boot ROM Flow

This section describes the software flow from reset until the boot ROM code passes software control to the preloader. The following figure illustrates that the boot ROM code can perform a warm boot from on-chip RAM, a cold boot from the FPGA portion of the device, or a cold boot from flash memory.

Figure A-5: BOOT ROM FLOW



During a cold boot from the FPGA portion of the device, the boot ROM code waits until the FPGA is ready and then attempts direct execution at address 0xC0000000 across the HPS-to-FPGA bridge. For example, the boot software could be provided by initialized on-chip RAM in the FPGA portion of the device at address 0xC0000000 (offset 0x0 from HPS-to-FPGA bridge). During a cold boot from flash memory, the boot ROM code attempts to load the first preloader image from flash memory to on-chip RAM and pass control to the preloader. If the image is invalid, the boot ROM code attempts to load up to three subsequent images from flash memory. If there is still no valid image found after the subsequent loads, the boot ROM code checks the FPGA portion of the device for a fallback image.

Warm boot from on-chip RAM has the highest priority to execute if the `warmramgrp` registers in the `romcodegrp` group in the system manager has been configured to support booting from on-chip RAM on a warm reset. When the enable in `warmramgrp` is enabled and length is set to 0x0, boot ROM will not perform CRC on the preloader image and immediately jump to the address in execution. If the length is not 0x0, boot ROM will ensure the preloader image passes the CRC check before the preloader image is executed.

If a valid preloader image cannot be found in the on-chip RAM, or the preloader in the on-chip RAM fails the CRC check, boot ROM code attempts to load the last valid preloader image loaded from the flash memory, identified by the `index` field of the initial software last image loaded register (`initswlastld`) in the `romcodegrp` group in the system manager. If the image is invalid, boot ROM code attempts to load up to three subsequent images from flash memory. If a valid preloader image cannot be found in the on-chip RAM or flash memory, the boot ROM code checks the FPGA portion of the device for a fallback image.

Loading the Preloader

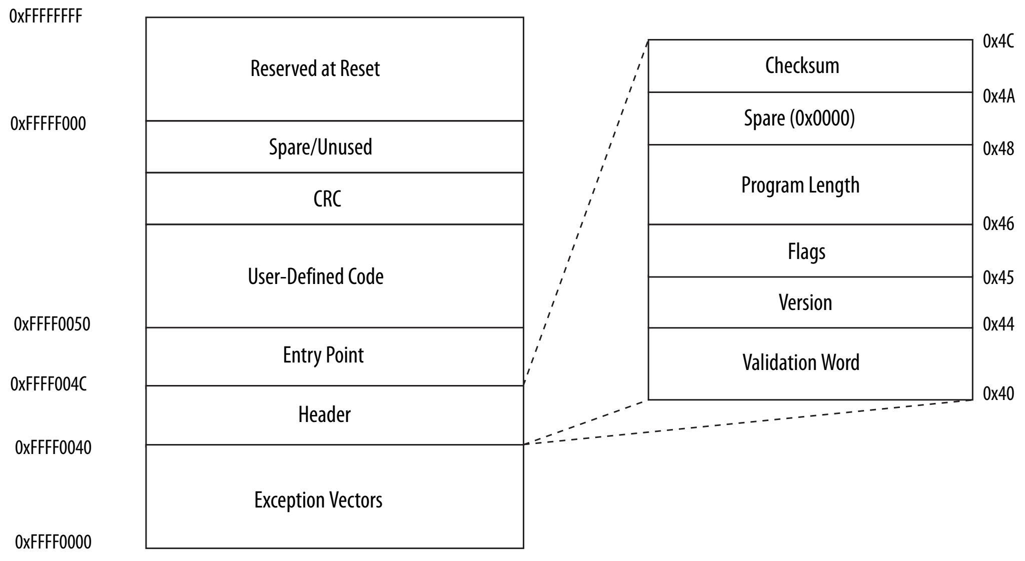
The boot ROM code loads the preloader image from flash memory into the on-chip RAM and passes control to the preloader. The boot ROM code checks for a valid image by verifying the header and cyclic redundancy check (CRC) in the preloader image.

The boot ROM code checks the header for the following information:

- Validation word—validates the preloader image. The validation word has a fixed value of 0x31305341.
- Version—indicates the header version.
- Program length—the total length of the image (in 32-bit words) from offset 0x0 to the end of code area, including exception vectors and CRC.
- Checksum—a checksum of all the bytes in the header, from offset 0x40 to 0x49.

The preloader image has a maximum size of 60 KB. This size is limited by the on-chip RAM size of 64 KB, where 4 KB is reserved as a workspace for the boot ROM data and stack. The preloader can use this 4 KB region (for its stack and data, for example) after the boot ROM code passes control to the preloader. This 4 KB region is overwritten by the boot ROM code on a subsequent reset. The following figure shows the preloader image layout in the on-chip RAM after being loaded from the boot ROM.

Figure A-6: Preloader Image Layout



Exception vectors—Exception vectors are located at the start of the on-chip RAM. Typically, the preloader remaps the lowest region of the memory map to the on-chip RAM (from the boot ROM) to create easier access to the exception vectors.

Header—contains information such as validation word, version, flags, program length, and checksum for the boot ROM code to validate the preloader image before passing control to the preloader.

Entry point—contains the preloader image address. After the boot ROM code validates the header, the boot ROM code jumps to this address.

User-defined code—typically contains the program code of the preloader.

CRC—contains a CRC of data from address 0xFFFF0000 to 0xFFFF0000+(Program Length*4)–0x0004. The polynomial used to validate the preloader image is *x 32 + x 26 + x 23 + x 22 + x 16 + x 12 + x 11 + x 10 + x 8 + x 7 + x 5 + x 4 + x 2 + x + 1*. There is no reflection of the bits. The initial value of the remainder is 0xFFFFFFFF and the final value is XORed with 0xFFFFFFFF.

Reserved at reset—the top 4 KB is reserved for the boot ROM code after a reset. The boot ROM code uses this area for internal structures, workspace, and post-mortem dump. This area includes the shared memory where the boot ROM code passes information to the preloader.

Shared Memory

The shared memory contains information that the boot ROM code passes to the preloader. The boot ROM code passes the location of shared memory to the preloader in register `r0`, as described in **HPS State on Entry to the Preloader** on page 30-11.

The shared memory contains the following information:

- Common—contains non-flash-specific settings used by the boot ROM code.
- Saved hardware register—contains hardware register values that the boot ROM code saves before the registers are modified by the boot ROM code.
- Flash device-specific—contains flash device-specific settings used by the boot ROM code that the preloader may use to continue using the flash device without reinitialization.

Table A-2: Shared Memory Block

Information Type	Content	Description
Common	Flash image	Indicates which preloader image (0-3) the boot ROM code loaded from the flash device. A nonzero value indicates that there was an error loading image 0 and the boot ROM code loaded another image.
	CLKSEL value used	Indicates the CLKSEL value used by the boot ROM code. Typically, this value is the value read from the `clksel` field of the `bootinfo` register in the system manager. However, if the PLL fails to lock, the boot ROM code ignores the `clksel` field and uses zero to indicate PLL bypass mode.
	BOOTSEL value used	Indicates the BOOTSEL value used by the boot ROM code. Typically, this value is the value read from the `bootsel` field of the `bootinfo` register in the system manager.
	Last page	Indicates the number of the last page read from the flash device.
	Page size	Indicates the page size (in bytes) used by the flash device. • For NAND flash memory, the boot ROM code reads this value from the NAND flash controller. • For SPI and quad SPI flash memory, the boot ROM code configures the page size through the quad SPI flash controller. • For SD/MMC flash memory, the boot ROM code configures the page size through the SD/MMC flash controller.
	Flash device type	Indicates the flash device used by the boot ROM code.
	Step complete	Track the completion state of up to 64 individual major steps during the boot process. The 64 bit value has one bit set for each major step completed in the boot

Information Type	Content	Description
Saved hardware registers	Status register (`stat`) of the reset manager	Contains reset source and event timeout information.
	Control (`ctrl`) register in the `romcodegrp` group in the system manager	Contains information used to control the boot ROM code.
	`initswstate` register in the `romcodegrp` group in the system manager	Contains the magic value 0x49535756 when the preloader has reached a valid state.
Flash device- specific (SD/MMC)	is_sd_card	Indicates the card type. 1 indicates SD card; 0 indicates MMC.
	is_sector_mode	Indicates the addressing mode of card. 1 indicates sector addressing mode; 0 indicates byte addressing mode.
	rca	Contains the relative card address, which is used for host-card communication during card identification
	partition_start_sector	Indicates the partition start offset in unit sectors. 0 indicates raw mode.
	partition_size	Size of the partition in unit sectors. 0 indicates raw mode.

Related Information

HPS State on Entry to the Preloader on page 30-11

L4 Watchdog 0 Timer

The L4 watchdog 0 timer is reserved for boot ROM use. While booting, if a watchdog reset happens before software control passes to the preloader, boot ROM code attempts to load the last valid preloader image, identified by the `initswlastld` register in the `romcodegrp` group in the system manager.

If the watchdog reset happens after the preloader has started executing but before the preloader writes a valid value to `initswstate` register, the boot ROM increments `initswlastld` and attempts to load that image. If the watchdog reset happens after the preloader writes a valid value to `initswstate` register, boot ROM code attempts to load the image indicated by `initswlastld` register.

HPS State on Entry to the Preloader

When the boot ROM code is ready to pass control to the preloader, the processor (CPU0) is in the following state:

- Instruction cache is enabled
- Branch predictor is enabled
- Data cache is disabled
- MMU is disabled
- Floating point unit is enabled
- NEON vector unit is enabled
- Processor is in ARM secure supervisor mode

The boot ROM code sets the ARM® Cortex™-A9 MPCore™ registers to the following values:

- r0—contains the pointer to the shared memory block, which is used to pass information from the boot ROM code to the preloader. The shared memory block is located in the top 4 KB of on-chip RAM.
- r1—contains the length of the shared memory.
- r2—unused and set to 0x0.
- r3—reserved.

All other MPCore registers are undefined.

Note: When booting CPU0 using the FPGA boot, or when booting CPU1 using any boot source, all MPCore registers, caches, the MMU, the floating point unit, and the NEON vector unit are undefined. HPS subsystems and the PLLs are undefined.

When the boot ROM code passes control to the preloader, the following conditions also exist:

- The boot ROM is still mapped to address 0x0.
- The L4 watchdog 0 timer is active and has been toggled.

Preloader

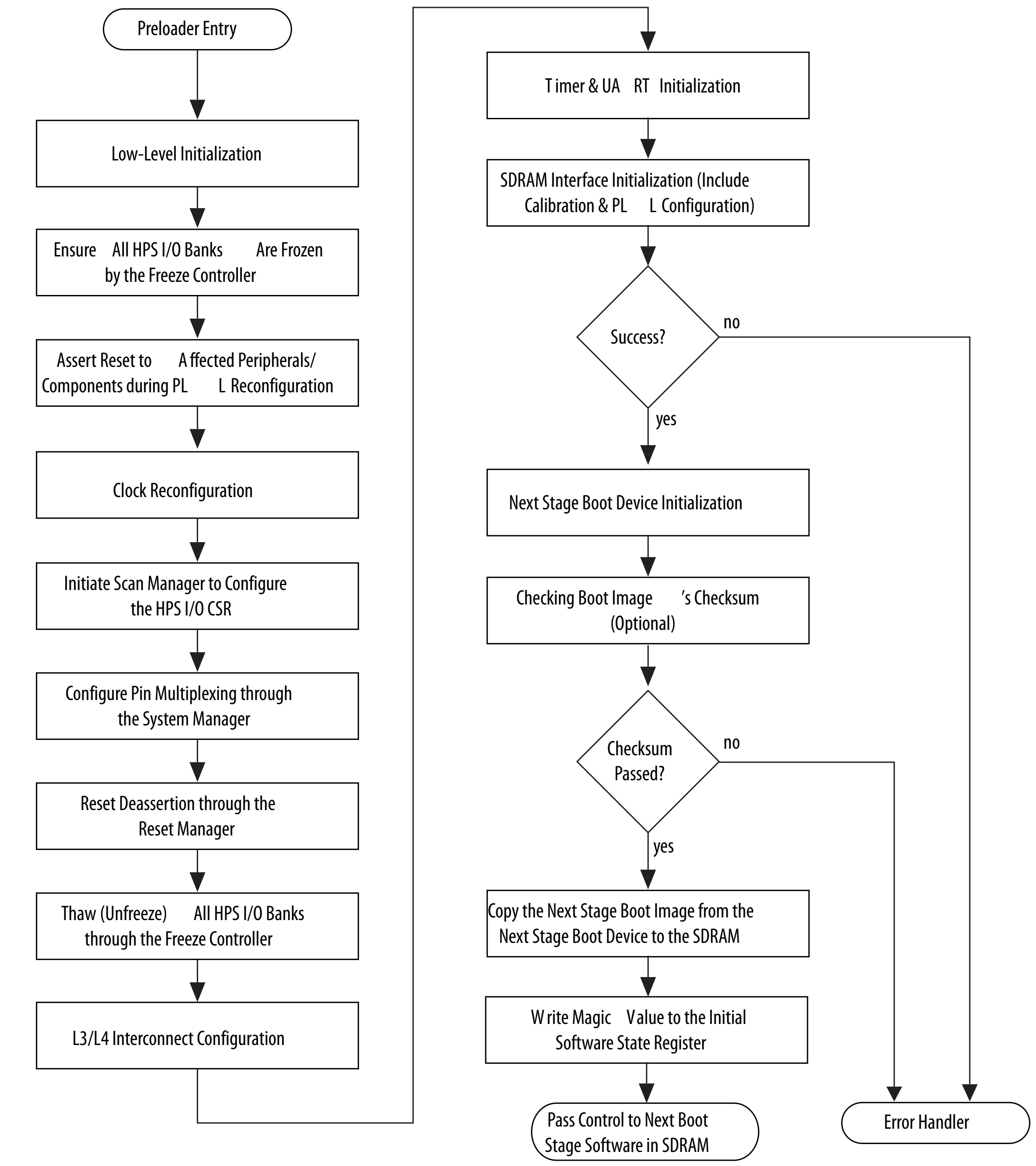
The preloader typically performs the following actions:

- Initialize the SDRAM interface.
- Configure the `remap` register to map the on-chip RAM to address 0x0 so that exceptions are handled by the preloader.
- The on-chip RAM is also accessible with the alias 0x0.
- Configure the HPS I/O through the scan manager.
- Configure pin multiplexing through the system manager.
- Configure HPS clocks through the clock manager.
- Initialize the flash controller (NAND, SD/MMC, or quad SPI) that contains the next stage boot software.
- Load the next stage boot software into the SDRAM and pass control to it.

Typical Preloader Boot Flow

This section describes a typical software flow from the preloader entry point until the software passes control to the next stage boot software.

Figure A-7: Typical Preloader Bootflow


Preloader Entry
Low-Level Initialization
Ensure All HPS I/O Banks Are Frozen by the Freeze Controller
Assert Reset to Affected Peripherals/ Components during PLL Reconfiguration
Clock Reconfiguration
Initiate Scan Manager to Configure the HPS I/O CSR
Configure Pin Multiplexing through the System Manager
Reset Deassertion through the Reset Manager
Thaw (Unfreeze) All HPS I/O Banks through the Freeze Controller
L3/L4 Interconnect Configuration
Timer & UART Initialization
SDRAM Interface Initialization (Include Calibration & PLL Configuration)
Success?
no
yes
Next Stage Boot Device Initialization
Checking Boot Image's Checksum (Optional)
Checksum Passed?
no
yes
Copy the Next Stage Boot Image from the Next Stage Boot Device to the SDRAM
Write Magic Value to the Initial Software State Register
Pass Control to Next Boot Stage Software in SDRAM
Error Handler

Low-level initialization steps include reconfiguring or disabling the L4 watchdog 0 timer, invalidating the instruction cache and branch predictor, remapping the on-chip RAM to the lowest memory region, and setting up the data area.

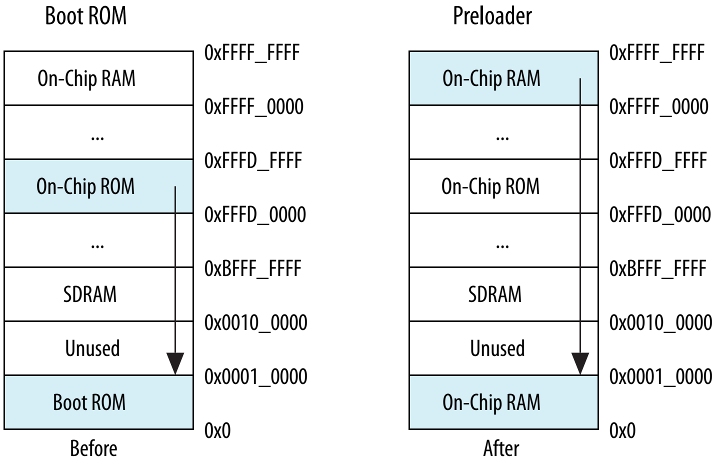
Upon entering the preloader, the L4 watchdog 0 timer is active. The preloader can either disable, reconfigure, or leave the watchdog timer unchanged. Once enabled after reset, the watchdog timer cannot be disabled, only paused.

The instruction cache and branch predictor, which were previously enabled by the boot ROM code, need to be invalidated.

The preloader needs to remap the exception vector table because the exception vectors are still pointing to the exception handler in the boot ROM when the preloader starts executing. By setting the L3 interconnect remap bit 0 to high, the on-chip RAM mirrors to the lowest region of the memory map. After this remap, the exception vectors will use the exception handlers in the preloader image.

The figure below shows the memory map before and after remap.

Figure A-8: Remapping the On-Chip RAM



The preloader can reconfigure all HPS clocks. During clock reconfiguration, the preloader asserts reset to the peripherals in the HPS affected by the clock changes.

The preloader configures HPS I/O pins through the scan manager and pin multiplexing through the system manager. The preloader initiates the freeze controller in the scan manager to freeze all the I/O pins and put them in a safe state during I/O configuration and pin multiplexing.

The I/O assignments for the HPS are configured as part of the IOCSR configuration in the preloader. Effectively, it's a bitstream that is sent to the device as part of the initialization code in the preloader.

The IOCSR is the complete bit settings for the I/Os embedded into the preloader code as a long array of configuration bits. There is an IOCSR for the EMIF portion of the HPS and for the other pins, they are physically separate bit streams. They are applied around the same time in the preloader. In the preloader code, you should be able to search for "IOCSR" and see a couple of long hex strings get manipulated.

The IOCSR in the preloader code is generated by the preloader generator from the **.hiof** file in the hps_isw_handoff folder. That file is generated when your design goes through the assembler, based on the usual **synthesis** > **filter** > **ASM** flow.

The SDRAM goes through full initialization for cold boot or a partial initialization for warm boot. For full initialization, the preloader configures the SDRAM PLL before releasing the SDRAM interface from reset. SDRAM calibration adjusts I/O delays and FIFO settings to compensate for any board skew or impairment in the board, FPGA portion of the device, or memory device. For partial initialization, SDRAM PLL configuration and SDRAM calibration is not necessary.

The preloader looks for a valid next stage boot image in the next stage boot device by checking the boot image validation data and checksum in the mirror image. Once validated, the preloader copies the next stage boot image from the next stage boot device to the SDRAM.

Before software passes control to the next stage boot software, the preloader can write a valid value (0x49535756) to the preloader `initswstate` register under the `romcodegrp` group in the system manager. This value indicates that there is a valid boot image in the on-chip RAM. When a warm reset occurs, the boot ROM code can check the `initswstate` register for the magic value to determine if it needs to reload the preloader image into the on-chip RAM.

Flash Memory Devices

The flash memory devices available for HPS boot are the NAND, SD/MMC, SPI, and quad SPI. The flash device can store the following kinds of files for booting purposes:

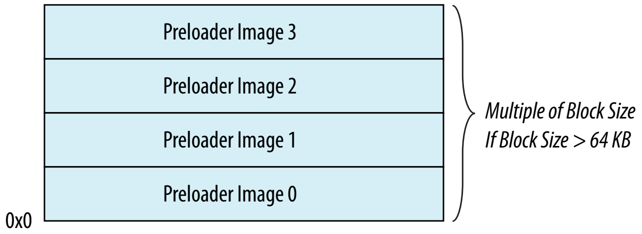
- Preloader binary file (up to four copies)
- Boot loader binary file
- Operating system binary file
- Application file
- FPGA programming files

Note: The preloader file must be stored in a partition with no file system.

NAND Flash Devices

The following figure shows that the preloader image is located at offsets which are multiples of the block size. For NAND device with block size equal or greater than 64kB, the preloader images are located in the first 4 blocks of the device. For NAND device with less than 64kB block size, preloader image needs to be placed in multiple of blocks. Since a block is the smallest area used for erase operation, any update to a particular image does not affect other images.

Figure A-9: NAND Flash Image Layout



Related Information

NAND Flash Controller on page 13-1
More information about the NAND flash memory

NAND Flash Driver Features Supported in the Boot ROM Code

Table A-3: NAND Flash Support Features

Feature	Driver Support
Device	Open NAND Flash Interface (ONFI) 1.0 raw NAND or electronic signature devices, single layer cell (SLC)

Feature	Driver Support
Chip select	CS0 only. Only CS0 is available to the HPS, the other three chip selects are routed out to the FPGA portion of the device
Bus width	x8 only
Page size	512 bytes, 2 KB, 4 KB, or 8 KB
Page per block	32, 64, 128, 384 and 512
ECC	512-bytes with 8-bit correction

CLKSEL Pin Settings for the Quad SPI Controller

Table A-4: NAND Controller CLKSEL Pin Settings

Setting	CSEL Pin			
	0	1	2	3
osc1_clk (EOSC1 pin) range	10–50 MHz	10–12.5 MHz	12.5–25 MHz	25–50 MHz
Device frequency (nand_x_clk/25)	osc1_clk/25, 2 MHz max	osc1_clk*20/25, 9.6 MHz max	osc1_clk*10/25, 9.6 MHz max	osc1_clk*5/25, 9.6 MHz max
controller clock (nand_x_clk)	osc1_clk, 50 MHz max	osc1_clk*20, 240 MHz max	osc1_clk*10, 240 MHz max	osc1_clk*5, 240 MHz max
mpu_clk	osc1_clk, 50 MHz max	osc1_clk*32, 400 MHz max	osc1_clk*16, 400 MHz max	osc1_clk*8, 400 MHz max
PLL modes	Bypassed	Locked	Locked	Locked

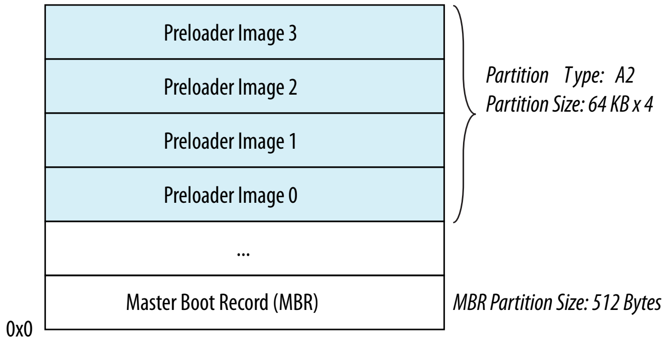
SD/MMC Flash Devices

The following figure shows the SD/MMC flash image layout. The master boot record (MBR) is located at the first 512 bytes of the memory. The MBR contains information of partitions (address and size of partition). The preloader image is always stored in partition A2. Partition A2 is a custom raw partition with no file system.

The start address of each image is based on the following formula:

Start address = partition start address + (<n> * 64 K), where <n> is the image number

Figure A-10: SD/MMC Flash Image Layout



The SD/MMC controller supports two booting modes:

- MBR (partition) mode
- The boot image is read from a custom partition (0xA2)
- The first image is located at the beginning of the partition, at offset 0x0
- Start address = partition start address
- Raw mode
- If the MBR signature is not found, SD/MMC driver assumes it is in raw mode.
- The boot image data is read directly from sectors in the user area and is located at the first sector of the SD/MMC.
- The first image is located at the start of the memory card, at offset 0
- Start address = 0

The MBR contains the partition table, which is always located on the first sector (LBA0) with a memory size of 512 bytes. The MBR consists of executable code, four partition entries, and the MBR signature. A MBR can be created by specific tools like the FDISK program.

Table A–5 lists the MBR structure.

Table A-5: MBR Structure

Offset	Size (Byte)	Description
0x000	446	Code area
0x1BE	16	Partition entry for partition 1
0x1CE	16	Partition entry for partition 2
0x1DE	16	Partition entry for partition 3
0x1EE	16	Partition entry for partition 4
0x1FE	2	MBR signature: 0xAA55

The standard MBR structure contains a partition with four 16-bytes entries. Thus, memory cards using this standard table cannot have more than four primary partitions or up to three primary partitions and one extended partition.

Each partition type is defined by the partition entry. The boot images are stored in a primary partition with custom partition type (0xA2). The SD/MMC flash driver does not support a file system, so the boot images are located in partition A2 at fixed locations.

Table A-6: Partition Entry

Offset	Size (Byte)	Description
0x0	1	Boot indicator. 0x80 indicates that is it bootable.
0x1	3	Starting CHS value
0x4	1	Partition type
0x5	3	Ending CHS value
0x8	4	LBA of first section in partition
0xB	4	Number of sectors in partition

The boot ROM code configures the SD/MMC controller to default settings for the supported SD/MMC flash memory.

Related Information

SD/MMC Controller on page 14-1

Default settings of the SD/MMC controller.

Table A-7: SD/MMC Controller Default Settings

Parameter	Default	Register Value
Card type	1 bit	The card type register (`ctype`) in the SD/MMC controller registers (`sdmmc`) = 0x0
Timeout	Maximum	The timeout register (`tmout`) = 0xFFFFFFFF
FIFO threshold RX watermark level	1	The RX watermark level field (`rx_wmark`) of the FIFO threshold watermark register (`fifoth`) = 0x1

Parameter		Default	Register Value
Clock source		0	The clock source register (clksrc) = 0x0
Block size		512	The block size register (blksiz) = 0x200
Clock divider	Identification mode	32	The clock divider register (clkdiv)= 0x10 (2*16=32)
	Data transfer mode	Bypass	The clock divider register (clkdiv)= 0x00

CLKSEL Pin Settings for the SD/MMC Controller

Table A-8: SD/MMC Controller CLKSEL Pin Settings

Setting		CLKSEL Pin			
		0	1	2	3
osc1_clk (EOSC1 pin) range		10–50 MHz	10–12.5 MHz	12.5–25 MHz	25–50 MHz
ID mode	Device clock (sdmmc_cclk_out)	osc1_clk/128, 391 KHz max	osc1_clk/32, 391 KHz max	osc1_clk/64, 391 KHz max	osc1_clk/128, 391 KHz max
	Controller baud rate divisor	32	32	32	32
Data transfer mode	Device clock (sdmmc_cclk_out)	osc1_clk/4, 12.5 MHz max	osc1_clk*1, 12.5 MHz max	osc1_clk/2, 12.5 MHz max	osc1_clk/4, 12.5 MHz max
	Controller baud rate divisor (even numbers only)	1 (bypass)	1 (bypass)	1 (bypass)	1 (bypass)
Controller clock (sdmmc_clk)		osc1_clk, 50 MHz max	osc1_clk, 50 MHz max	osc1_clk, 50 MHz max	osc1_clk*2, 50 MHz max
mpu_clk		osc1_clk, 50 MHz max	osc1_clk*32, 400 MHz max	osc1_clk*16, 400 MHz max	osc1_clk*8, 400 MHz max
PLL modes		Bypassed	Locked	Locked	Locked

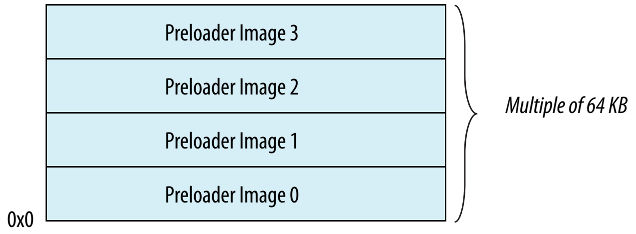
SPI and Quad SPI Flash Devices

The figure below shows the SPI and quad SPI flash image layout. The preloader image is always located at offsets which are multiples of 64 KB. The boot ROM code only supports QSPI devices with Mode Reset Command. Common manufacturers for Mode Reset Command are Spansion and Winbond.

The first image is located at offset 0 and followed by subsequent images. The start address of each image is based on the following formula:

Start address = (N * 64K), where N is the image number

Figure A-11: SPI and Quad SPI Flash Image Layout



The boot ROM code configures the quad SPI controller to default settings for the supported SPI or quad SPI flash memory.

Related Information

Quad SPI Flash Controller on page 15-1

Quad SPI Controller Default Settings

Table A-9: Quad SPI Controller Default Settings

Parameter	Default Setting	Register Value
SPI baud rate.	Divide by 4	The master mode baud rate divisor field (`bauddiv`) of the quad SPI configuration register (`cfg`) in the quad SPI controller registers (`qspiregs`) = 1.

Read opcode.	CLKSEL = 9 or 1: Normal read
	CLKSEL = 2 or 3: Fast read
		The read opcode in non-XIP mode field (`rdopcode`) in the device read instruction register (`devrd`) = 0x3 (for normal read) and 0xB (for fast read).

Instruction type.	Single I/O (1 bit wide)	The address transfer width field (`addrwidth`) and data transfer width field (`datawidth`) of the `devrd` register = 0.

Parameter	Default Setting	Register Value
Delay in master reference clocks for the length that the master mode chip select outputs are deasserted between words when the clock phase is zero.	200 ns	The clock delay for chip select deassert field (`nss`) in the quad SPI device delay register (`delay`). Refer to `delay[31:24]` in Table SPI and Quad SPI Flash Delay Configuration.
Delay in master reference clocks between one chip select being deactivated and the activation of another. This delay ensures a quiet period between the selection of two different slaves and requires the transmit FIFO to be empty.	0 ns	The clock delay for chip select deactivation field (`btwn`) in the `delay` register = 0x0.
Delay in master reference clocks between the last bit of the current transaction and the first bit of the next transaction. If the clock phase is zero, the first bit of the next transaction refers to the cycle in which the chip select is deselected.	20 ns	The clock delay for last transaction bit field (`after`) in the `delay` register. Refer to `delay[15:8]` in Table SPI and Quad SPI Flash Delay Configuration.
Added delay in master reference clocks between setting `qspi_n_ss_out` low and first bit transfer.	20 ns	The clock delay with `qspi_n_ss_out` field (`init`) in the `delay` register. Refer to `delay[7:0]` in Table SPI and Quad SPI Flash Delay Configuration.
Number of address bytes.	3 bytes	The number of address bytes field (`numaddrbytes`) of the device size register (`devsz`) = 2. Note: Before a reset, you must ensure that the QSPI flash device is configured to 3 bytes address mode for the boot ROM to function properly.

Quad SPI Controller CSEL Pin Settings

Table A-10: Quad SPI Controller CSEL Pin Settings

Setting	CSEL Pin			
	0[49]	1	2	3
`osc1_clk` (`EOSC1` pin) range	10–50 MHz	20–50 MHz	25–50 MHz	10–25 MHz

[49] Not applicable when on WARM reset.

Setting	CSEL Pin			
	0[49]	1	2	3
Device clock (sclk_out)	osc1_clk/4, 12.5 MHz max	osc1_clk/2, 25 MHz max	osc1_clk*1, 50 MHz max	osc1_clk*2, 50 MHz max
Controller clock (qspi_clk)	osc1_clk, 50 MHz max	osc1_clk*2, 100 MHz max	osc1_clk*4, 200 MHz max	osc1_clk*8, 200 MHz max
Controller baud rate divisor (even numbers only)	4	4	4	4
Flash read instruction (1 dummy byte for READ_FAST)	READ	READ	READ_FAST	READ_FAST
mpu_clk	osc1_clk, 50 MHz max	osc1_clk*8, 400 MHz max	osc1_clk*8, 400 MHz max	osc1_clk*16, 400 MHz max
PLL modes	Bypassed	Locked	Locked	Locked

SPI and Quad SPI Flash Delay Configuration

The delay register in the quad SPI controller registers (qspiregs) configures relative delay into the generation of the master output signals.

The SPI or quad SPI flash memory needs to meet the following timing requirements:

- T_{SLCH} (delay[7:0]): 20 ns
- T_{CHSH} (delay[15:8]): 20 ns
- T_{SHSL} (delay[31:24]): 200 ns

Table A-11: SPI and Quad SPI Flash Delay Configuration

CSEL Pin	T_{ref_clk} (ns)	T_{sclk_out} (ns)	Device Delay Register		
			delay[7:0]	delay[15:8]	delay[31:24]
0	20	80	1	1	6
1	10	40	2	2	16
2–3	5	20	4	4	36

The formula to calculate the delay is

delay[7:0] = T_{SLCH}/T_{qspi_clk}

delay[15:8] = T_{CHSH}/T_{qspi_clk}

delay[31:24] = $(T_{SHSL} - T_{sclk\_out})/T_{qspi\_clk}$

[49] Not applicable when on WARM reset.

FPGA Configuration

You can configure the FPGA portion of the SoC device with non-HPS sources or by utilizing the HPS. Software executing on the HPS configures the FPGA by writing the configuration image to the FPGA manager in the HPS. Software can control the configuration process and monitor the FPGA status by accessing the control and status register (CSR) interface in the FPGA manager.

Related Information

- **FPGA Manager** on page 4-1
  More information about configuring the FPGA through the HPS FPGA manager
- **http://www.altera.com/literature/hb/cyclone-v/cv_52007.pdf**
  For more information about configuring the FPGA in general, refer to the *Configuration, Design Security, and Remote System Upgrade* appendix in the Cyclone V Device Handbook, Volume 1.

Full Configuration

The HPS uses the FPGA manager to configure the FPGA portion of the device. The following sequence suggests one way for software to perform a full configuration:

- CONF_DONE = 1 and nSTATUS = 1 indicates successful configuration.
- CONF_DONE = 0 or nSTATUS = 0 indicates unsuccessful configuration. Complete steps 12 and 13, then go back and repeat steps 3 to 10 to reload the configuration image.

- If DCLK is unused, write a value of 4 to the DCLK count register (dclkcnt).
- If DCLK is used, write a value of 20,480 (0x5000) to the dclkcnt register.

If the HPS resets in the middle of a normal configuration data transfer before entering user mode, software can assume that the configuration is unsuccessful. After the HPS resets, software must repeat the steps for full configuration.

1. Set the cdratio and cfgwdth bits of the ctrl register in the FPGA manager registers (fpgamgrregs) to match the characteristics of the configuration image. These settings are dependant on the MSEL pins input.
2. Set the nce bit of the ctrl register to 0 to enable HPS configuration.
3. Set the en bit of the ctrl register to 1 to give the FPGA manager control of the configuration input signals.
4. Set the nconfigpull bit of the ctrl register to 1 to pull down the nCONFIG pin and put the FPGA portion of the device into the reset phase.
5. Poll the mode bit of the stat register and wait until the FPGA enters the reset phase.
6. Set the nconfigpull bit of the ctrl register to 0 to release the FPGA from reset.
7. Read the mode bit of the stat register and wait until the FPGA enters the configuration phase.
8. Clear the interrupt bit of nSTATUS (ns) in the gpio interrupt register (fpgamgr-regs.mon.gpio_porta_eoi).
9. Set the axicfgen bit of the ctrl register to 1 to enable sending configuration data to the FPGA.
10. Write the configuration image to the configuration data register (data) in the FPGA manager module configuration data registers (fpgamgrdata). You can also choose to use a DMA controller to transfer the configuration image from a peripheral device to the FPGA manager.
11. Use the fpgamgrregs.mon.gpio_ext_porta registers to monitor the CONF_DONE (cd) and nSTATUS (ns) bits.
12. Set the axicfgen bit of the ctrl register to 0 to disable configuration data on AXI slave.

13. Clear any previous DONE status by writing a 1 to the `dcntdone` bit of the DCLK status register (`dclkstat`) to clear the completed status flag.
14. Send the `DCLKs` required by the FPGA to enter the initialization phase.
15. Poll the `dcntdone` bit of the `DCLK` status register (`dclkstat`) until it changes to 1, which indicates that all the `DCLKs` have been sent.
16. Write a 1 to the `dcntdone` bit of the `DCLK` status register to clear the completed status flag.
17. Read the `mode` bit of the `stat` register to wait for the FPGA to enter user mode.
18. Set the `en` bit of the `ctrl` register to 0 to allow the external pins to drive the configuration input signals.

Document Revision History

Table A-12: Document Revision History

Date	Version	Changes
June 2014	2014.06.30	Maintenance release
February 2014	2014.02.28	Correction to "Leading the Preloader" section
December 2013	2013.12.30	• Updated figures in the Booting and Configuration Introduction section. • Updated the Rest and Boot ROM sections. • Updated the Shared Memory Block table. • Updated register names in the Full Configuration section.
November 2012	1.3	• Expanded shared memory block table. • Added CLKSEL tables. • Additional minor updates.
June 2012	1.2	Updated the HPS boot and FPGA configuration sections.

Date	Version	Changes
May 2012	1.1	• Updated the HPS boot section. • Added information about the flash devices used for HPS boot. • Added information about the FPGA configuration mode.
January 2012	1.0	Initial release.