CMC

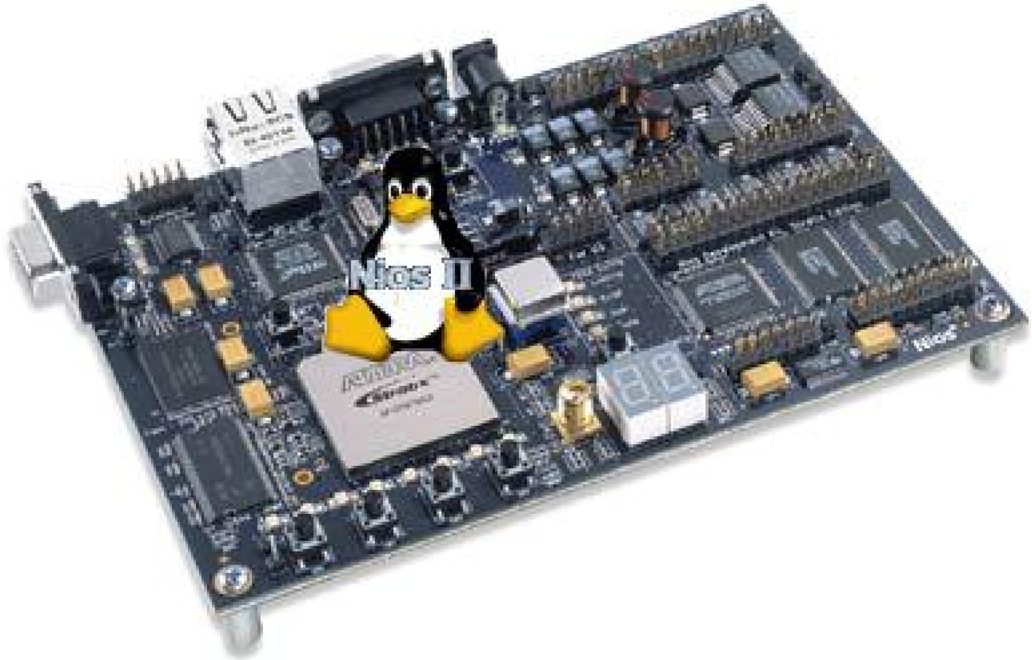
www.cmc.ca

Nios II Linux Tutorial

for the System-Level Prototyping Station for Embedded Systems

ICI-154
V1.0

July 22, 2005



License

Copyright

Trademarks

Revision History

REVISION	ACTIVITY	DATE
1.0 Beta	Prepared for beta testing by lead clients.	March 23, 2005
1.0	Initial release in the P&S Catalogue	July 22, 2005

Table of Contents

List of Figures

List of Tables

1. Introduction

In February 2005, CMC shipped round two of the System Level Prototyping Stations (SLPS) for Embedded Systems to the universities that are members of the System-on-Chip Research Network. Software pre-installed on this Embedded Systems SLPS includes Altera's Quartus II 4.2, the Nios II Development Kit 1.1 and Microtronix Nios II Linux 1.3. With these three tools, users can create a complete embedded system that includes the Nios II hardware platform, Embedded Linux operating system and user-defined applications.

To complement these deliverables, CMC created this tutorial with a reference design. It is intended to help users get familiar with the SLPS for Embedded Systems, particularly with how the elements in the tool chain work together to build a system.

This tutorial will teach you how to create a complete embedded system with Nios II and Embedded Linux. It starts at the very beginning: creating a platform using Quartus II software and building embedded software using the Nios II Integrated Development Environment (IDE). The tutorial will take you to the end: a functional and complete system on the Nios development board.

There are two main functions that the complete embedded system demonstrated in this tutorial will achieve: a readable-writable file system on Linux and networking function through Ethernet.

2. Environment Description

2.1 Required Components

It is assumed that the development environment at your site has been set up according to the document *Getting Started with the SLPS for Embedded Systems* (Report ICI-139). Therefore, set up information is not provided as part of this tutorial. The getting started document mentioned above also includes information on supporting vendor documentation. A hardcopy was shipped with the SLPS systems, and an electronic copy is available from CMC's Technology Gateway at:
https://www2.cmc.ca:2804/
(search for **SLPS Embedded System**
and select the link **System-Level Prototyping Station (SLPS) for Embedded Systems**)

The environment in this tutorial is made of the pre-installed software and hardware components listed below. If you don't have the required environment, follow the instructions in *Section 2.2, Upgrading Your Environment.*

The components required for this tutorial are:

- IBM PC with Windows XP
- Altera Nios Development Board, Stratix Pro Edition
- Quartus II 4.2 or later
- Nios II IDE 1.1 or later
- Microtronix Nios II Linux 1.3 or later
- Microtronix CompactFlash component
- USB-Blaster download cable
- Network cable connected to the local network

2.2 Upgrading Your Environment

If you do not have all the components listed above, the following might help:

1. Check versions of the two Altera components (Quartus II and Nios).
    a) If Quartus II is at V3.0 and Nios is at V3.10, upgrade these components to Quartus II V4.2 and Nios II IDE V1.1. Your site should have received this upgrade from Altera directly.
    b) If Quartus II is at V4.2 and Nios II IDE is at V1.1, you do not need to upgrade the Altera software.
    c) If Quartus II is at V5.0 and Nios II IDE is at V5.0, you do not need to upgrade the Altera software. Although this tutorial was created using Quartus II V4.2 and Nios II IDE V1.1, the new versions will still function in this tutorial.
2. Check to see if Nios II Linux V1.3 has been installed on your system. If not, obtain and install the deliverable **Nios II Linux V1.3** at from the following location:
   https://www2.cmc.ca:2804/
   (search for **Nios II Linux Tutorial**)
3. If your system is installed with Quartus II V 5.0 and Nios II IDE V 5.0, you do not need to perform this final step. All other users must obtain and install the deliverable **Microtronix CompactFlash** from the following location:
   https://www2.cmc.ca:2804/
   (search for **Nios II Linux Tutorial**)

3. Reference Design: System Specifications

The system we are going to create in this tutorial includes two parts: a simple web server and a file system with read-write mode. Therefore the system must have the following functional components:

- Nios II core platform
- Embedded Linux kernel
- File system with read-write mode
- Networking function
- Web server function

To implement such a system, you need the hardware and software listed below.

3.1 Hardware Requirements

- Nios II processor
- 16 MB memory
- CompactFlash component
- Ethernet
- Timer
- Input module
- Output module

3.2 Software Requirements

- Embedded Linux kernel with support for Ethernet, IDE and CompactFlash card
- File system support for ROMFS and EXT2FS
- Applications include ifconfig, mount, fdisk, e2fsprogs, sh, fileutils, boa, and basic system utilities. A customized BusyBox is also necessary.
- The software must be no more than 16 MB in total.

4. Hardware Platform Implementation

It is assumed that you are already familiar with Quartus II and SOPC (system-on-a-programmable-chip) Builder. Please refer to the Quartus II online tutorial and Nios II hardware tutorial for information on how to use these tools.

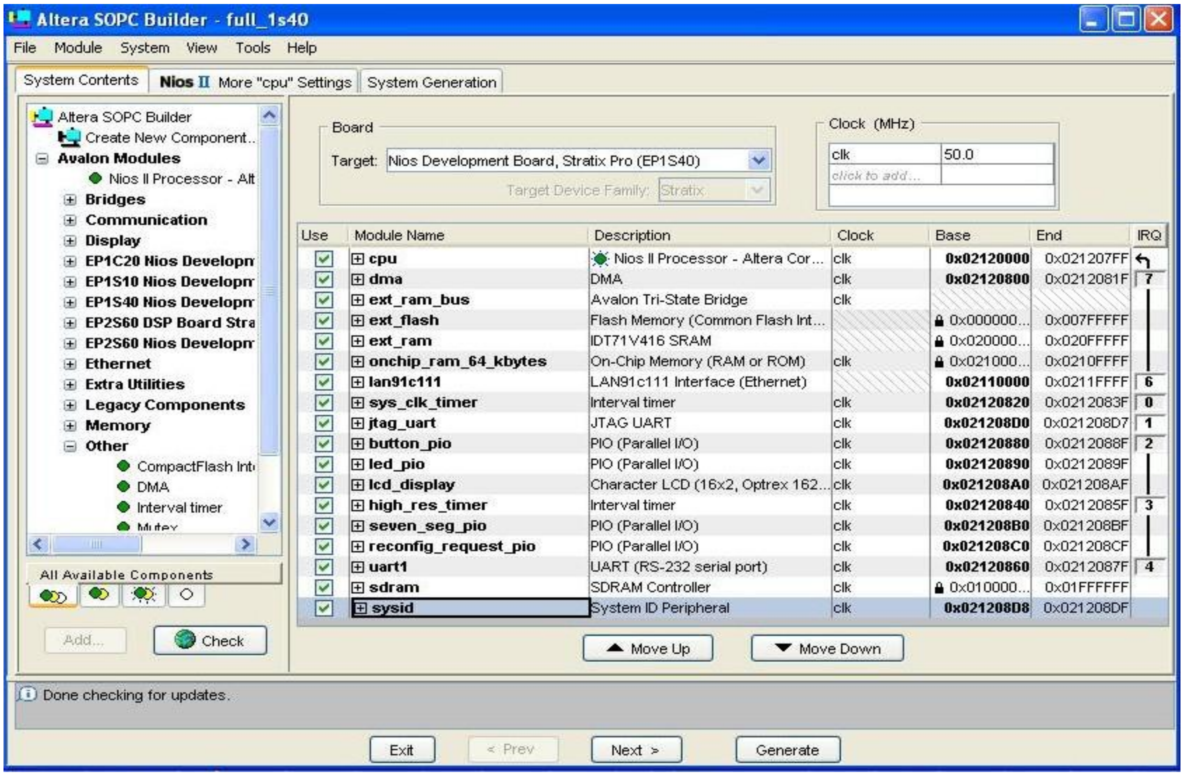
In order to implement a Nios II platform in a short time, instead of creating a platform from scratch, we are going to modify the existing platform at:
C:\altera\kits\nios2\examples\verilog\niosII_stratix_1s40\full_featured\

The full_featured example already has most of the components required by the embedded system, except a CompactFlash (CF) card. The following steps will add the CF card to the platform and remove some components that are not used by the embedded system in this tutorial.

Make sure that the CF component and Microtronix Nios II Linux have been installed on your machine. Refer to Section 2.2 for information on downloading these components.

4.1 Modify the Existing Platform

4. Copy the example **full_featured** to a working directory. In this case, it is **C:\Tutorial_Linux**
5. Open the project by selecting **Quartus II | File |Open Project…** From the **Open Project** window, browse to the directory C:\Tutorial_Linux\full_featured\ and select the **full_featured.qpf**, click **Open**
6. In the **full_featured.bdf** window, double click on the **full_1s40** instance to open it in the **SOPC Builder** tool.
7. In the **Altera SOPC Builder** window, click on the **System Contents** panel.
8. You can see all the components of the design example platform as shown in Figure 1.



Board

Target: Nios Development Board, Stratix Pro (EP1S40)

Target Device Family: Stratix

Clock (MHz)

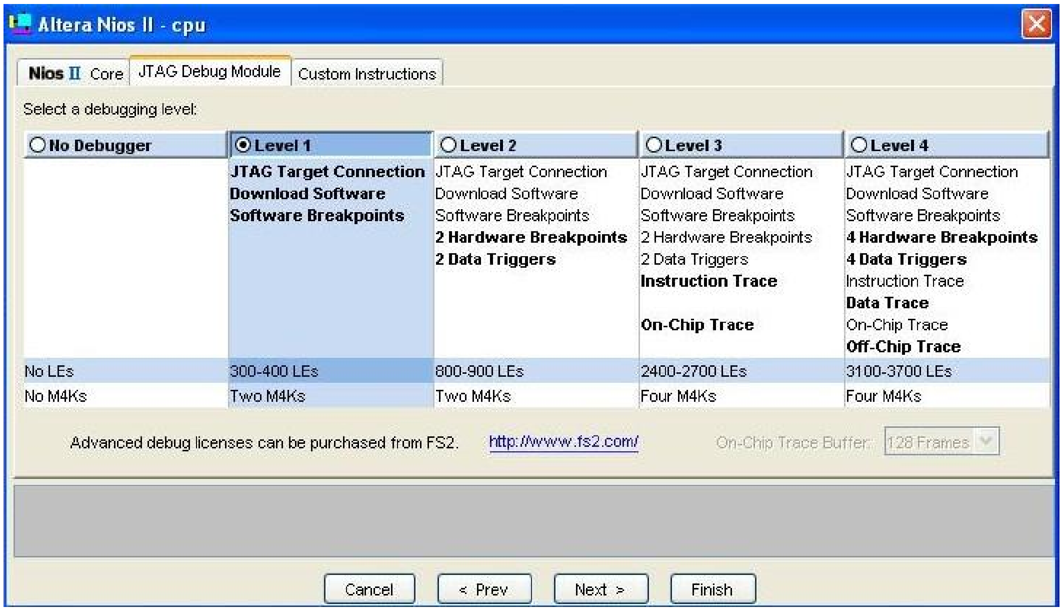
clk	50.0
click to add...	

Use	Module Name	Description	Clock	Base	End	IRQ
☑	⊞ cpu	Nios II Processor - Altera Cor...	clk	0x02120000	0x021207FF	
☑	⊞ dma	DMA	clk	0x02120800	0x0212081F	7
☑	⊞ ext_ram_bus	Avalon Tri-State Bridge	clk			
☑	⊞ ext_flash	Flash Memory (Common Flash Int...		0x000000...	0x007FFFFF	
☑	⊞ ext_ram	IDT71V416 SRAM		0x020000...	0x020FFFFF	
☑	⊞ onchip_ram_64_kbytes	On-Chip Memory (RAM or ROM)	clk	0x021000...	0x0210FFFF	
☑	⊞ lan91c111	LAN91c111 Interface (Ethernet)		0x02110000	0x0211FFFF	6
☑	⊞ sys_clk_timer	Interval timer	clk	0x02120820	0x0212083F	0
☑	⊞ jtag_uart	JTAG UART	clk	0x021208D0	0x021208D7	1
☑	⊞ button_pio	PIO (Parallel I/O)	clk	0x02120880	0x0212088F	2
☑	⊞ led_pio	PIO (Parallel I/O)	clk	0x02120890	0x0212089F	
☑	⊞ lcd_display	Character LCD (16x2, Optrex 162...	clk	0x021208A0	0x021208AF	
☑	⊞ high_res_timer	Interval timer	clk	0x02120840	0x0212085F	3
☑	⊞ seven_seg_pio	PIO (Parallel I/O)	clk	0x021208B0	0x021208BF	
☑	⊞ reconfig_request_pio	PIO (Parallel I/O)	clk	0x021208C0	0x021208CF	
☑	⊞ uart1	UART (RS-232 serial port)	clk	0x02120860	0x0212087F	4
☑	⊞ sdram	SDRAM Controller	clk	0x010000...	0x01FFFFFF	
☑	⊞ sysid	System ID Peripheral	clk	0x021208D8	0x021208DF	

Done checking for updates.

Figure 1: Original full_featured Platform in SOPC Builder

9. Set the JTAG Debug Module by double-clicking on the **cpu** component. The **Altera Nios II** window appears as shown in Figure 2.



Select a debugging level:

○ No Debugger	● Level 1	○ Level 2	○ Level 3	○ Level 4
	JTAG Target Connection **Download Software** **Software Breakpoints**	JTAG Target Connection Download Software Software Breakpoints **2 Hardware Breakpoints** **2 Data Triggers**	JTAG Target Connection Download Software Software Breakpoints 2 Hardware Breakpoints 2 Data Triggers **Instruction Trace** **On-Chip Trace**	JTAG Target Connection Download Software Software Breakpoints **4 Hardware Breakpoints** **4 Data Triggers** Instruction Trace **Data Trace** On-Chip Trace **Off-Chip Trace**
No LEs	300-400 LEs	800-900 LEs	2400-2700 LEs	3100-3700 LEs
No M4Ks	Two M4Ks	Two M4Ks	Four M4Ks	Four M4Ks

Advanced debug licenses can be purchased from FS2. http://www.fs2.com/

On-Chip Trace Buffer: 128 Frames

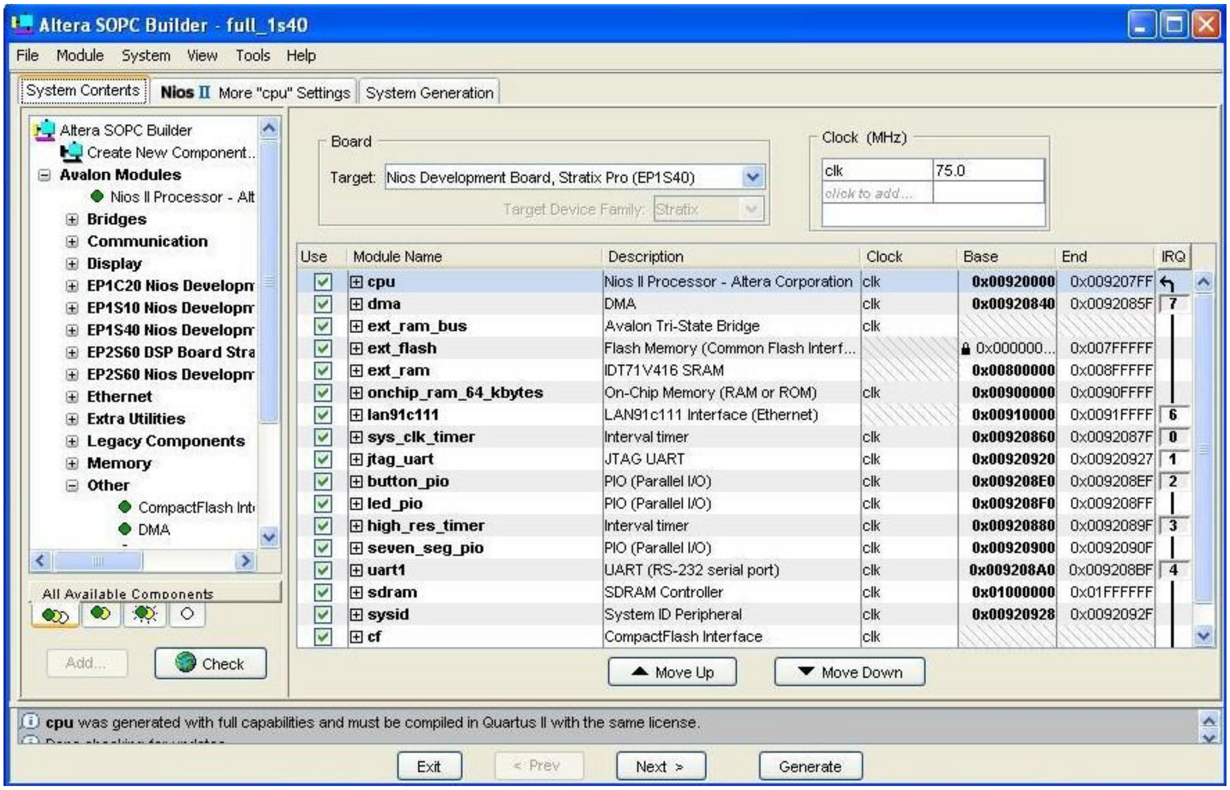
Figure 2: Set up JTAG Debug Module

10. Click on the **JTAG Debug Module** panel, and select **Level 1**
11. Click on **Finish**
12. Remove the **lcd_display** component by right-clicking on the component name, and select the **Delete** command from the drop-down menu.
13. Remove the **reconfig_request_pio** component in the same way as above.
14. To add the CF card, from **All Available Components** (on the left), select **Avalon Modules | Other | CompactFlash Interface**
15. Unlock the base address from **ext_ram**, **onchip_ram_64_kbytes** and **sdram** as follows: Right-click on the **Base** field of each component and select the **Lock Base Address** command.
16. Select **System | Auto-Assign Base Address**
17. Assign the IRQ as described in Table 1.

Table 1: IRQ Assignment

Component Name	Sys_clk_timer	Jtag_uart	Button_pio	High_res_timer	dma
IRQ No.	0	1	2	3	7
Component Name	Uart1	cf_0.ide	cf_0.ctl	Lan91c111	
IRQ No.	4	5	8	6	

18. Change the Clock Frequency from 50.0 MHz to 75.0 MHz.
19. Figure 3 shows the modified platform in SOPC Builder.
20. Click on the **Generate** icon at the bottom of the window to generate the modified system.



Board

Target: Nios Development Board, Stratix Pro (EP1S40)

Target Device Family: Stratix

Clock (MHz)

clk	75.0
click to add...	

Use	Module Name	Description	Clock	Base	End	IRQ
☑	⊞ cpu	Nios II Processor - Altera Corporation	clk	0x00920000	0x009207FF	↰
☑	⊞ dma	DMA	clk	0x00920840	0x0092085F	7
☑	⊞ ext_ram_bus	Avalon Tri-State Bridge	clk			
☑	⊞ ext_flash	Flash Memory (Common Flash Interf...		🔒 0x000000...	0x007FFFFF	
☑	⊞ ext_ram	IDT71V416 SRAM		0x00800000	0x008FFFFF	
☑	⊞ onchip_ram_64_kbytes	On-Chip Memory (RAM or ROM)	clk	0x00900000	0x0090FFFF	
☑	⊞ lan91c111	LAN91c111 Interface (Ethernet)		0x00910000	0x0091FFFF	6
☑	⊞ sys_clk_timer	Interval timer	clk	0x00920860	0x0092087F	0
☑	⊞ jtag_uart	JTAG UART	clk	0x00920920	0x00920927	1
☑	⊞ button_pio	PIO (Parallel I/O)	clk	0x009208E0	0x009208EF	2
☑	⊞ led_pio	PIO (Parallel I/O)	clk	0x009208F0	0x009208FF	
☑	⊞ high_res_timer	Interval timer	clk	0x00920880	0x0092089F	3
☑	⊞ seven_seg_pio	PIO (Parallel I/O)	clk	0x00920900	0x0092090F	
☑	⊞ uart1	UART (RS-232 serial port)	clk	0x009208A0	0x009208BF	4
☑	⊞ sdram	SDRAM Controller	clk	0x01000000	0x01FFFFFF	
☑	⊞ sysid	System ID Peripheral	clk	0x00920928	0x0092092F	
☑	⊞ cf	CompactFlash Interface	clk			

ⓘ **cpu** was generated with full capabilities and must be compiled in Quartus II with the same license.

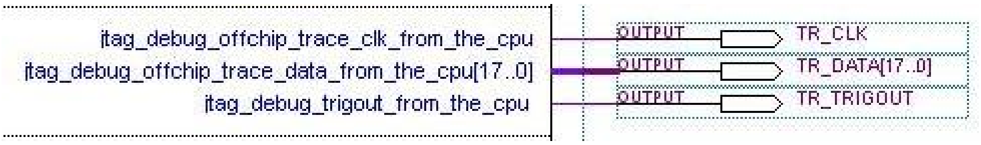
Figure 3:The Modified Platform in the SOPC Builder

4.2 Modify Example and Compile the Quartus II Project

To help you modify the existing **full_featured** example, a pinout table of CompactFlash is provided in Appendix A.

The following is the steps describe how to modify the example:

1. After the generate process of SOPC Builder is finished, click on **Exit**
   You will be asked "Save changes to full_featured.bdf?"
2. Click **Yes**
3. In the **full_featured.bdf** window, since the platform has been changed, you will find some mismatch between the input/output ports and the full_ls40 instance. To match the full_ls40 instance with the input/output ports, you will need to remove/add some ports. Please follow the steps below to remove and add them.
4. Remove the output ports for the **JTAG module**, **reset_request_pio** and **LCD module**. For all of the modules shown in Figure 4, you need to remove the related ports.


jtag_debug_offchip_trace_clk_from_the_cpu
jtag_debug_offchip_trace_data_from_the_cpu[17..0]
jtag_debug_trigout_from_the_cpu
OUTPUT
TR_CLK
OUTPUT
TR_DATA[17..0]
OUTPUT
TR_TRIGOUT

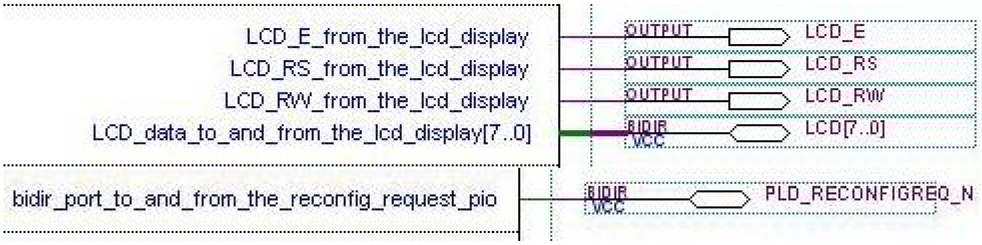
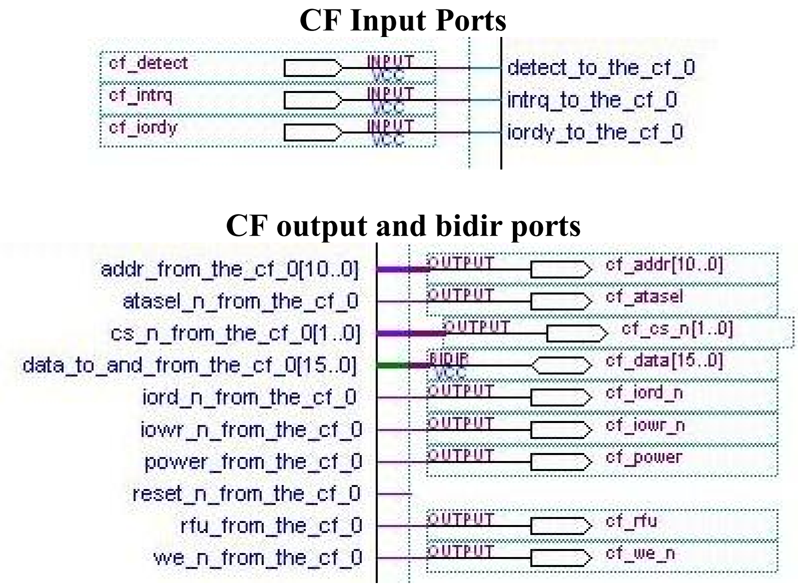

LCD_E_from_the_lcd_display
LCD_RS_from_the_lcd_display
LCD_RW_from_the_lcd_display
LCD_data_to_and_from_the_lcd_display[7..0]
OUTPUT
LCD_E
OUTPUT
LCD_RS
OUTPUT
LCD_RW
BIDIR
VCC
LCD[7..0]
bidir_port_to_and_from_the_reconfig_request_pio
BIDIR
VCC
PLD_RECONFIGREQ_N

Figure 4: Output Ports To Remove

5. Adjust the other input and output port connections including the system clock module, Ethernet module, ext_ram, sdram, led, button_pio, seven_seg_pio and uart modules.
6. For all of the modules shown in Figure 5. you need to add the related input ports and output ports for the CF (CompactFlash) module.

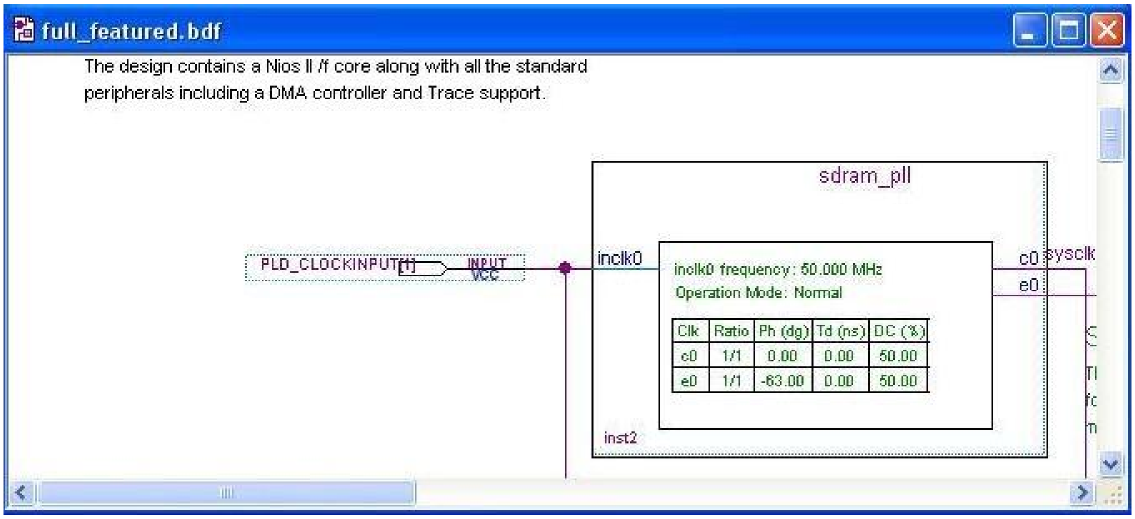

CF Input Ports

CF output and bidir ports

Figure 5: CF Input, Output and Bidir Ports To Remove

7. Double-click on the sdram_pll instance in the full_featured.bdf window (see Figure 6). The MegaWizard Plug-in Manager will give you a message like

"Delay shifts (time delay elements) are no longer supported in Stratix PLLs, Use Phase Shift feature instead to implement the desired time shift."


full_featured.bdf

The design contains a Nios II /f core along with all the standard peripherals including a DMA controller and Trace support.

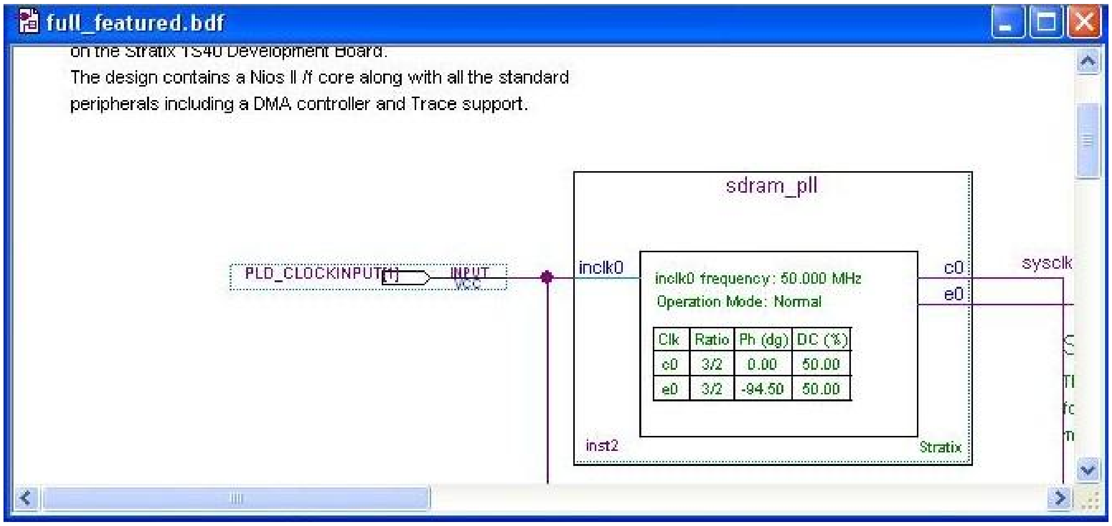
inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	Td (ns)	DC (%)
c0	1/1	0.00	0.00	50.00
e0	1/1	-63.00	0.00	50.00

Figure 6: Original sdram_pll Instance

8. Click **OK**
9. The **MegaWizard Plug-in Manager – ALTCLKLOCK** window appears. Just click on the **Finish** icon at the bottom of the window. Then click on the **Finish**, **OK** and **Yes** icons in the following windows until you are taken back to the full-featured.bdf window.

   In the **full_featured.bdf** window you will see that parts co and e0 have been broken. After you reconnect these, you will find that the **sdram_pll** symbol has been updated as shown in Figure 7.


full_featured.bdf

on the Stratix 1S40 Development Board.
The design contains a Nios II /f core along with all the standard peripherals including a DMA controller and Trace support.

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	3/2	0.00	50.00
e0	3/2	-94.50	50.00

Figure 7: Updated sdram_pll Instance

10. Save the changes to **full_featured.bdf**
11. Go to your working directory and remove the .sof and .qdf files.
12. Go back to Quartus II and compile the project by selecting **Processing | Start Compilation**
13. When the compilation is over, you will be able to get the **full_featured.sof** for the modified platform.

5. Software Implementation

It is assumed that you are already familiar with the Nios II IDE and Nios II shell environment. Please refer to the IDE online tutorial for information on how to use the IDE.

You will need administrator privileges to use some software functions. Otherwise, you will need to ask your administrator to change the permission of the .bashrc to 644.

5.1 Create a Linux Kernel Project

Please follow the instructions below to create a Linux kernel project based on the hardware platform created in the previous section.

1. Open the Nios II IDE by selecting
   All Programs | Altera | Nios II Development Kit 1.1 | Nios II IDE
2. In the Nios II IDE window, select **File | New | Project…**
3. In the New Project window, select **Microtronix Nios II** on the left and select **Linux Kernel Project** on the right (refer to Figure 8).

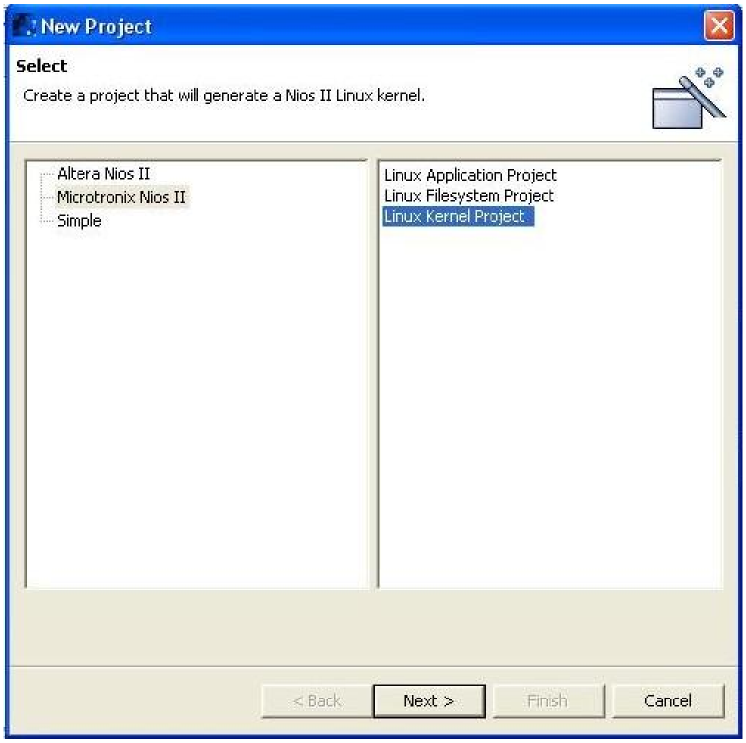
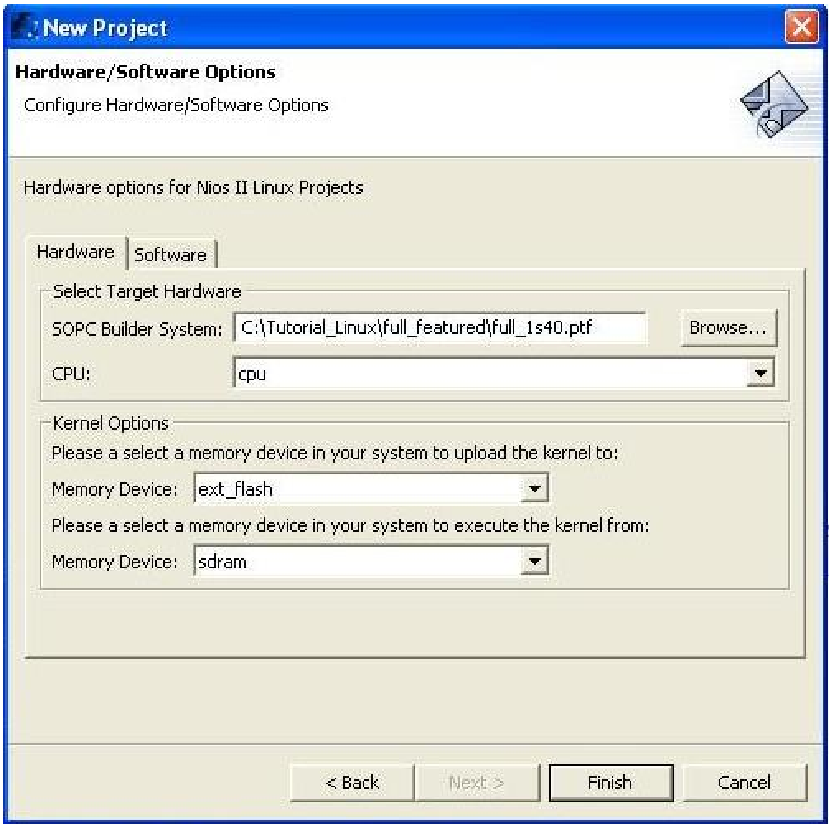


Figure 8: Creating a Linux Kernel Project

4. Click **Next**
5. Specify the project name as **tutorial_linux_kernel**
6. Click **Next**

7. As shown in Figure 9, in the **Hardware** panel specify the .ptf file in the **SOPC Builder System** field. This .ptf file was generated in the Quartus II project in Section 4. In this tutorial, the file is:
   C:\Tutorial_Linux\full_featured\full_1s40.ptf



Hardware/Software Options

Configure Hardware/Software Options

Hardware options for Nios II Linux Projects

Select Target Hardware

SOPC Builder System: C:\Tutorial_Linux\full_featured\full_1s40.ptf

CPU: cpu

Kernel Options

Please a select a memory device in your system to upload the kernel to:

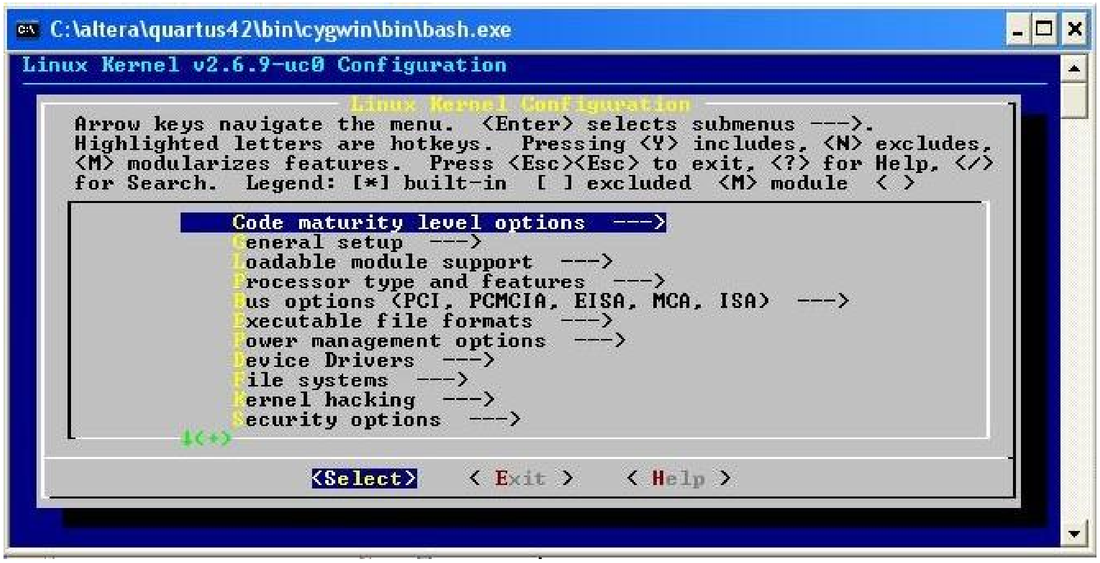
Memory Device: ext_flash

Please a select a memory device in your system to execute the kernel from:

Memory Device: sdram

Figure 9: Creating a Linux Kernel Project Continued

8. Click **Finish**
9. In the Navigator window, right-click on the Linux Kernel Project name **tutorial_linux_kernel**, and select the **Configure Kernel** command from the drop-down menu.
   The kernel configuration tool is invoked as shown in Figure 10.
10. Table 2. shows how you should configure the kernel.



```
C:\altera\quartus42\bin\cygwin\bin\bash.exe
Linux Kernel v2.6.9-uc0 Configuration

                     Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

              Code maturity level options  --->
              General setup  --->
              Loadable module support  --->
              Processor type and features  --->
              Bus options (PCI, PCMCIA, EISA, MCA, ISA)  --->
              Executable file formats  --->
              Power management options  --->
              Device Drivers  --->
              File systems  --->
              Kernel hacking  --->
              Security options  --->
          ↓(+)

              <Select>    < Exit >    < Help >
```

Figure 10: Kernel Configuration

Table 2: Kernel Configuration

Most of the default configurations can be kept but you must make sure that the following items are configured as shown.
Processor type and features Platform <Altera Stratix Pro.Development board support>--> <*> Altera Stratix Pro.Development board support
Device Drivers ATA/ATAPI/MFM/RLL support --> <*> ATA/ATAPI/MFM/RLL support <*> Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support <*> Include IDE/ATA-2 DISK support [*] Use multi-mode by default <*> generic/default IDE chipset support [*] Other IDE chipset support <*> Altera CF (IDE mode) interface (Avalon bus) support Networking support--> [*] Networking support [*] Network device support Ethernet <10 or 100Mbit> --> <*> SMC 91111 support
File systems <*> Second extended fs support <*> ROM file system support

11. After you have made the required changes to the kernel configuration, save the changes and exit from the Linux Kernel Configuration tool.
12. Go back to the **Navigator** window of the Nios II IDE.
13. Right-click on the Linux kernel project name **tutorial_linux_kernel**, and select the **Build Project** command from the drop-down menu.
14. After the project is successfully built, the Linux kernel image file **vmlinux.bin** is generated under the **tutorial_linux_kernel** project.

5.2 Create a Linux File System Project

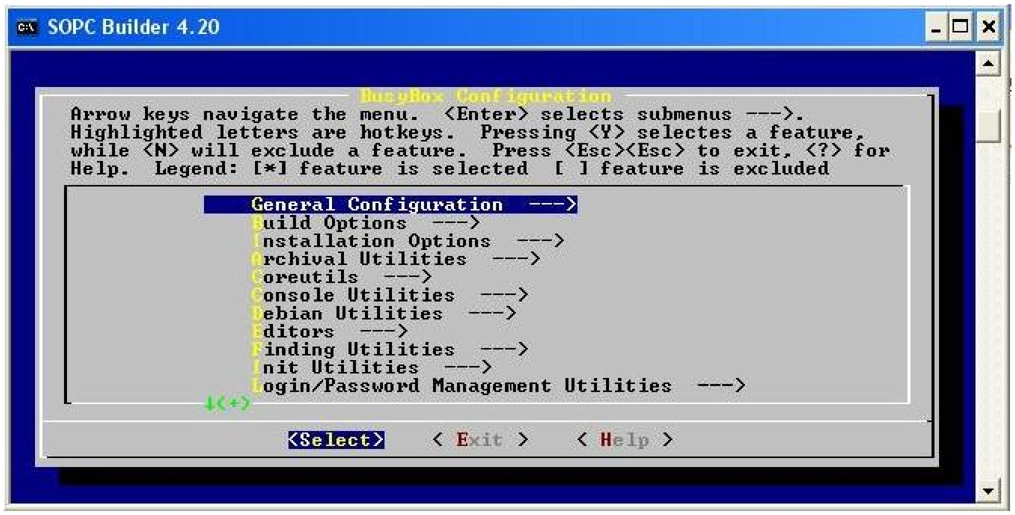
1. To create a new project, in the Nios II IDE window, select **File | New | Project…**
2. In the **New Project** window, select **Microtronix Nios II** on the left and select **Linux Filesystem Project** on the right.
3. In the Nios II Linux Filesystem Project window, specify the **Project name** as **tutorial_linux_file**
4. Click on **Next**
5. Specify the .ptf file in the **SOPC Builder System** field.
6. Click **Nex**t
7. In the **Target Filesystem Application Selection** window, under the **Pre-build Binary Packages**, select the checkboxes for the following applications to include them in the file system:
   base, boa, e2fsprogs, fdisk, fileutils, ping, mount and sh
8. Click **Finish**
9. In the Navigator window, expand the **tutorial_linux_file** project, check the /target/bin folder. Note that you won't find the busybox application in it. The busybox application will be built in the next section.

   Note: There is a web page (index.html) of the Microtronix uKit under /target/home/httpd/ folder. You can either keep it or replace it with your own web page.

5.3 Customize BusyBox Application

The pre-built BusyBox application is too big for the system we created in the previous section. Here we are going to customize the BusyBox to reduce its size.

1. From the following path, open a Nios II SDK Shell:
   All Programs | Altera | Nios II Development Kit 1.1 | Nios II SDK Shell
2. Enter the following:
   ```
   cd software/linux/busybox
   make menuconfig
   ```
3. The BusyBox configuration window appears as shown in Figure 11.



```
                       BusyBox Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> selectes a feature,
  while <N> will exclude a feature.  Press <Esc><Esc> to exit, <?> for
  Help.  Legend: [*] feature is selected  [ ] feature is excluded

               General Configuration  --->
               Build Options  --->
               Installation Options  --->
               Archival Utilities  --->
               Coreutils  --->
               Console Utilities  --->
               Debian Utilities  --->
               Editors  --->
               Finding Utilities  --->
               Init Utilities  --->
               Login/Password Management Utilities  --->
           ↓(+)

                  <Select>    < Exit >    < Help >
```

Figure 11: BusyBox Configuration

4. Follow the information shown in Table 3. to configure the BusyBox.

For purposes outside this tutorial, you can select any application inside of the BusyBox. Note that some of the applications are also included in the parallel Linux fileutils.exe. The following configuration just includes the basic applications to meet the reference design system requirements.

Table 3: BusyBox Configuration

Build Options -- > [*] Build BusyBox as a static binary (no shared libs) [*] Do you want to build BusyBox with a Cross Compiler? <nios2-elf-> Compiler prefix ($(ECLIPSE_WORKSPACE)/tutorial_linux_kernel/build) (Specify the Linux Kernel build directory here)
Installation Options -- > ($(ECLIPSE_WORKSPACE)/tutorial_linux_file/target) (Specify the Linux file system target directory here)
Editors -- > [*] vi
Linux Module Utilities -- > [*] insmod [*] Support version 2.6.x Linux [*] lsmod [*] Support lsmod query_module interface <add 638 bytes> [*] modprobe [*] rmmod [*] Support tained module checking with new kernels
Networking Utilities -- > [*] hostname [*] ifconfig [*] Enable status reporting output
Linux System Utilities -- > [*] umount

5. Save the changes and exit the BusyBox configuration.
6. From the SDK shell, type the following commands:
   ```
   make dep
   make
   make install
   ```
7. In the **Navigato**r window of the **Nios II IDE**, right-click on the **tutorial_linux_file** and select the **Refresh** command from the drop-down menu.
8. Check the **target/bin** directory under **the tutorial_linux_file** project in the **Navigator** window. You should find the BusyBox application has been installed in your file system project.

5.4 Build the tutorial_linux_file Project

1. In the **Navigator** window, right-click on the file system project name **tutorial_linux_file**, and select the **Build Project** command from the drop-down menu.
2. After the building project process is finished, the file system image file **romfs.bin** is generated under the **tutorial_linux_file** project.

Note : The size of the romfs.bin might exceed the memory limitation 2048K. To solve this problem, you can delete some of the applications in the /target/bin folder of the **tutorial_linux_file** project before you build it. However, be sure to keep the applications that you selected in Section 5.2, step 7.

6. Download Images

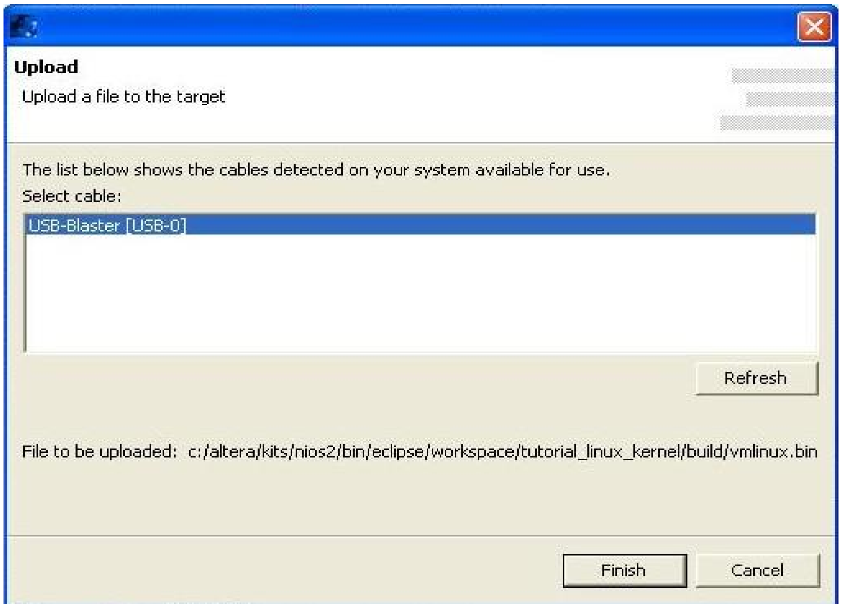
So far we have built a Nios II hardware platform, a Nios II Linux kernel and a file system. In this section we are going to upload all the image files to the target board.

Make sure that the PC is properly connected to the target board. Please refer to the setting up section of the ***Getting Started for SLPS-Embedded System*** document for information on how to properly connect the PC to the target board.

If you are using a time-limited OpenCore, then you will need to keep the Programmer open during the entire downloading process.

6.1 Download the Linux Kernel to the Target

1. Unplug the CompactFlash card from the socket of the board.
2. Power on the board.
3. In the **Navigator** window of the Nios II IDE, expand the **tutorial_linux_kernel** project by clicking on the ⊞ symbol on the left of the project name.
4. Find the **vmlinux.bin** file under the **tutorial_linux_kernel**
5. Right click on the file **vmlinux.bin**
6. Click on the **upload** command from the drop-down menu.
   An upload window will appear as shown in Figure 12.



Upload

Upload a file to the target

The list below shows the cables detected on your system available for use.

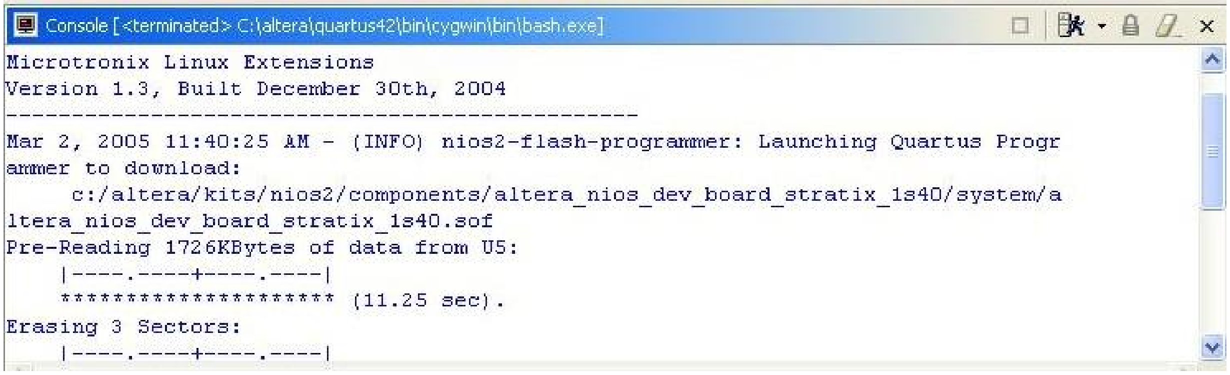
Select cable:

USB-Blaster [USB-0]

File to be uploaded: c:/altera/kits/nios2/bin/eclipse/workspace/tutorial_linux_kernel/build/vmlinux.bin

Figure 12: Uploading the Kernel

7. Select **USB-Blaster**
8. Click **Finish**
9. Wait until the uploading is done (refer to Figure 13).



Console [<terminated> C:\altera\quartus42\bin\cygwin\bin\bash.exe]

```
Microtronix Linux Extensions
Version 1.3, Built December 30th, 2004
------------------------------------------------
Mar 2, 2005 11:40:25 AM - (INFO) nios2-flash-programmer: Launching Quartus Progr
ammer to download:
     c:/altera/kits/nios2/components/altera_nios_dev_board_stratix_1s40/system/a
ltera_nios_dev_board_stratix_1s40.sof
Pre-Reading 1726KBytes of data from U5:
    |----.----+----.----|
    ********************* (11.25 sec).
Erasing 3 Sectors:
    |----.----+----.----|
```

Figure 13: Console Message for Kernel Uploading

6.2 Upload the File System to the Target

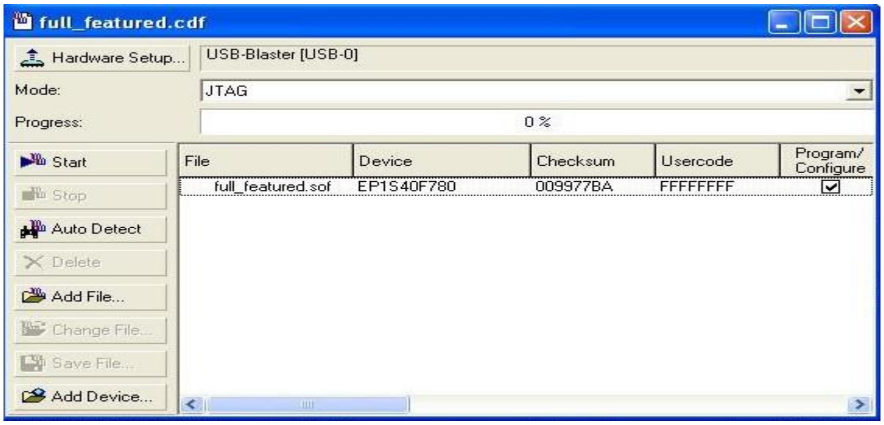
1. In the **Navigator** window of the Nios II IDE, expand the **tutorial_linux_file** project by clicking on the ⊞ symbol on the left of the project name.
2. Find the **romfs.bin** file under the **tutorial_linux_file**
3. Right-click on the file **romfs.bin**
4. Select the **upload** command from the drop-down menu.

5. An uploading window will pop up as shown in Figure 12.
6. Select **USB-Blaster**
7. Click **Finish**
8. Wait until the uploading is done. You will see a similar message from the console as shown in Figure 13.

6.3 Upload the FPGA Configuration File to the Target

With the target board properly connected to the PC and powered on, upload the FPGA configure file as follows:

1. Select **All Programs | Altera | Quartus II 4.2**
2. In the Quartus II window, select **Tools | Programmer** from the menu.
3. In the Programmer window, make sure that the hardware setup… has been set up as USB-Blaster (refer to Figure 14).



full_featured.cdf

Hardware Setup... USB-Blaster [USB-0]

Mode: JTAG

Progress: 0 %

File	Device	Checksum	Usercode	Program/ Configure
full_featured.sof	EP1S40F780	009977BA	FFFFFFFF	☑

Figure 14: Programmer Set up

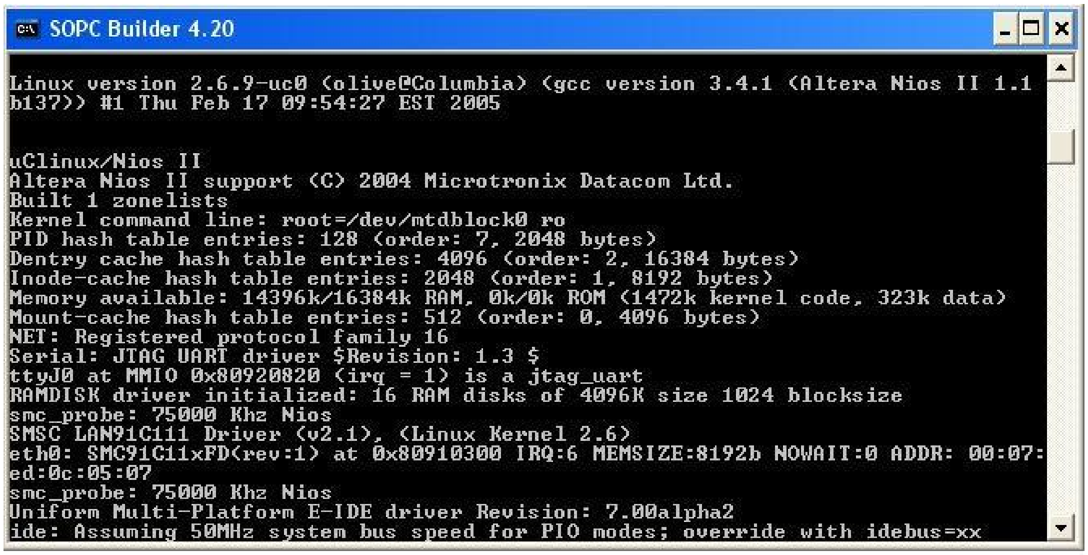
4. Click on the **Add File…** icon.
5. Browse to the location of the FPGA configure file **full_featured.sof** (in this tutorial, the location is C:\Tutorial_Linux\full_featured)
6. Open the **full_featured.sof**
7. Make sure to check the **program/configure** item.
8. Click on the **Start** icon to begin the loading and wait until the **progress** shows 100%.

7. Run and Configure the Embedded System

So far we have prepared the embedded software and the hardware FPGA configuration files and all of them have been uploaded to the memory on the target board. In this section, we are going to invoke the Embedded Linux on the target and make the necessary configuration to both the PC side and target side to implement the embedded system functions required in Section 3: System Specifications.

7.1 Invoke Linux

1. Open a Nios II SDK Shell by selecting
   All Programs | Altera | Nios II Development Kit 1.1 | Nios II SDK Shell
2. Type the command `nios2-terminal`from the SDK Shell. Note: If you have more then one downloading cable connected to your host, then you will need to specify which cable you will use to download using the following command :
   ```
   nios2-terminal -- calbe usb-blaster
   ```
3. You will see Linux boot up as shown in Figure 15.

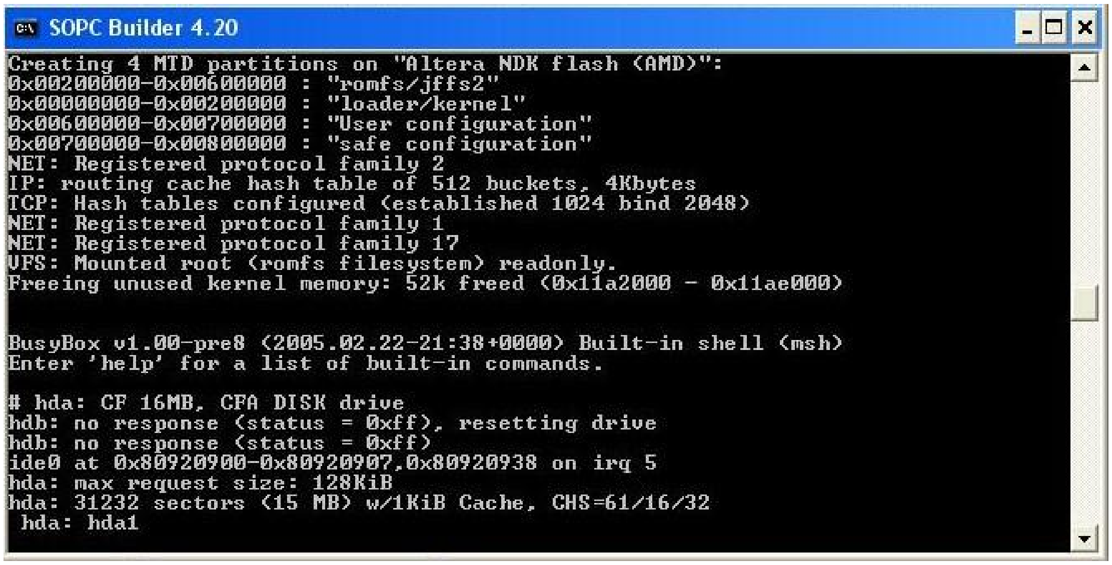


```
Linux version 2.6.9-uc0 (olive@Columbia) (gcc version 3.4.1 (Altera Nios II 1.1
b137)) #1 Thu Feb 17 09:54:27 EST 2005


uClinux/Nios II
Altera Nios II support (C) 2004 Microtronix Datacom Ltd.
Built 1 zonelists
Kernel command line: root=/dev/mtdblock0 ro
PID hash table entries: 128 (order: 7, 2048 bytes)
Dentry cache hash table entries: 4096 (order: 2, 16384 bytes)
Inode-cache hash table entries: 2048 (order: 1, 8192 bytes)
Memory available: 14396k/16384k RAM, 0k/0k ROM (1472k kernel code, 323k data)
Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
NET: Registered protocol family 16
Serial: JTAG UART driver $Revision: 1.3 $
ttyJ0 at MMIO 0x80920820 (irq = 1) is a jtag_uart
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
smc_probe: 75000 Khz Nios
SMSC LAN91C111 Driver (v2.1), (Linux Kernel 2.6)
eth0: SMC91C11xFD(rev:1) at 0x80910300 IRQ:6 MEMSIZE:8192b NOWAIT:0 ADDR: 00:07:
ed:0c:05:07
smc_probe: 75000 Khz Nios
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
```

Figure 15: Linux Boots Up

7.2 Create an EXT2FS on the CompactFlash (CF) card

1. Plug in the CF card which was unplugged from the board in the previous chapter.
2. You will see console messages relating to "hda" as shown in Figure 16.



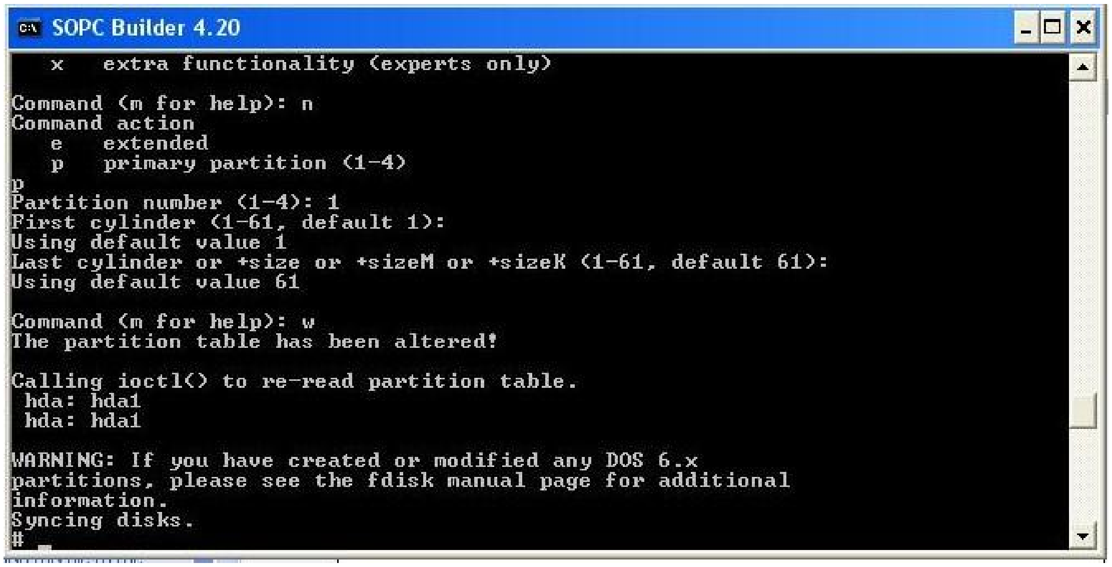
```
Creating 4 MTD partitions on "Altera NDK flash (AMD)":
0x00200000-0x00600000 : "romfs/jffs2"
0x00000000-0x00200000 : "loader/kernel"
0x00600000-0x00700000 : "User configuration"
0x00700000-0x00800000 : "safe configuration"
NET: Registered protocol family 2
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 2048)
NET: Registered protocol family 1
NET: Registered protocol family 17
VFS: Mounted root (romfs filesystem) readonly.
Freeing unused kernel memory: 52k freed (0x11a2000 - 0x11ae000)


BusyBox v1.00-pre8 (2005.02.22-21:38+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# hda: CF 16MB, CFA DISK drive
hdb: no response (status = 0xff), resetting drive
hdb: no response (status = 0xff)
ide0 at 0x80920900-0x80920907,0x80920938 on irq 5
hda: max request size: 128KiB
hda: 31232 sectors (15 MB) w/1KiB Cache, CHS=61/16/32
 hda: hda1
```

Figure 16: hda Console Message

3. Partition the CF card by typing the following command from the Linux Command Prompt: **`fdisk /dev/had`**

As shown in Figure 17:

1. From the main menu of the fdisk command, type the "**n**" command to build new partition.
2. When prompted for an extended or primary partition, issue "**p**" to select the primary partition.
3. When prompted for a partition number, issue the "**1**" command.
4. When prompted for the start sector, press the "**Enter**" key on your keyboard to take the default value.
5. When prompted for the end sector or size, press the "**Enter**" key on your keyboard to take the default value.
6. When you are brought back to the main menu, issue the "**w**" command to save the changes to the CF card.

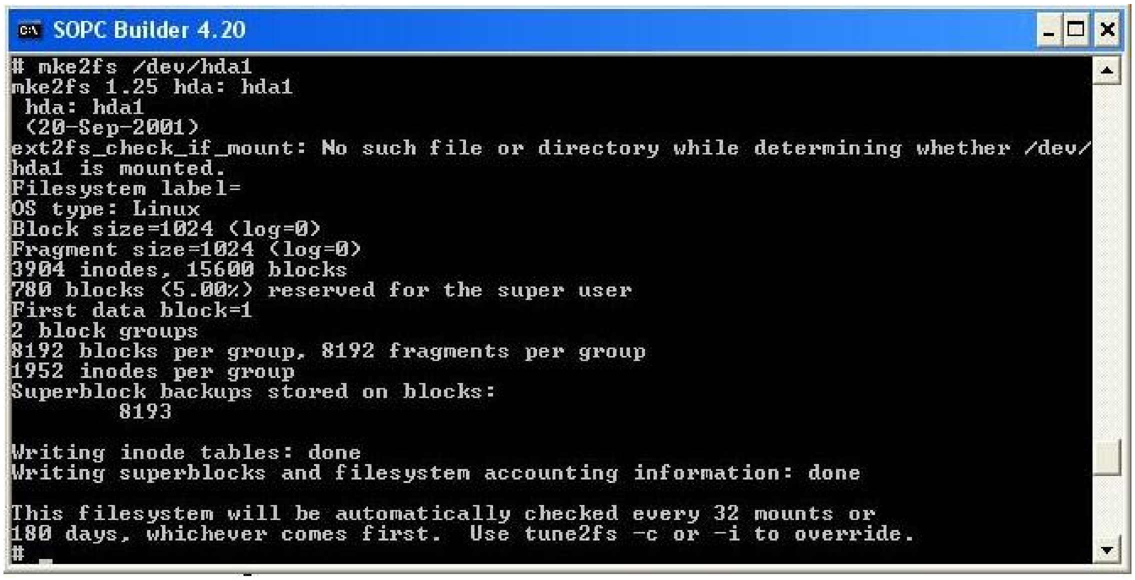


```
   x   extra functionality (experts only)

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-61, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-61, default 61):
Using default value 61

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
 hda: hda1
 hda: hda1

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
# _
```

Figure 17: Create CF Card Partition

Now you are back to the Linux Command prompt.

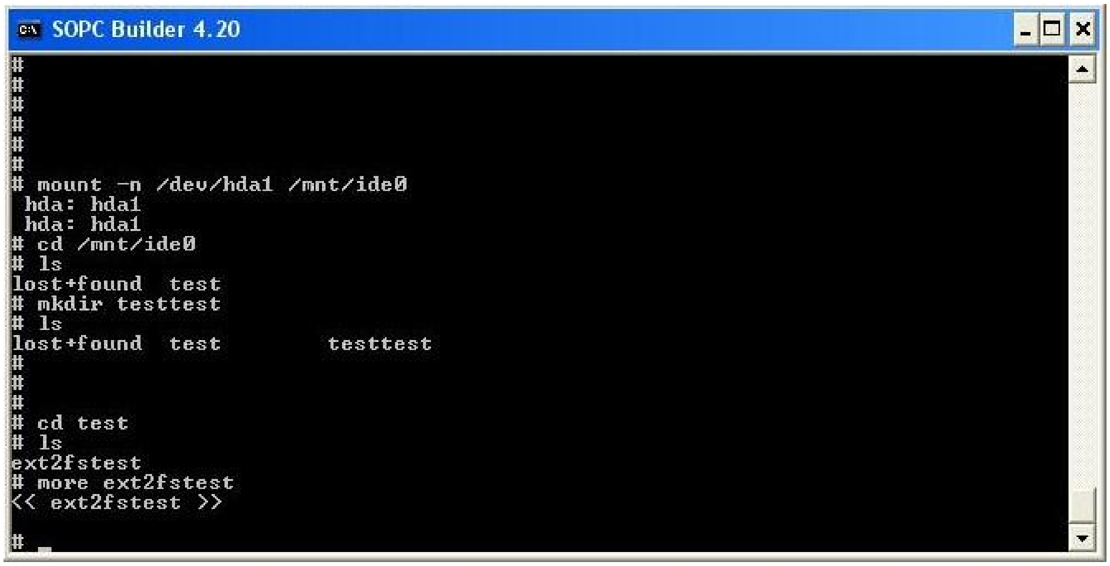
7. Make a **ext2fs** on the **/dev/hda1** by typing the command below:
   `mke2fs /dev/hda1`
8. Mount the /dev/hda1 to the Linux file system by typing the command below:
   `mount -n /dev/hda1 /mnt/ide0`
9. You will get the messages shown in Figure 18.



```
# mke2fs /dev/hda1
mke2fs 1.25 hda: hda1
 hda: hda1
 (20-Sep-2001)
ext2fs_check_if_mount: No such file or directory while determining whether /dev/
hda1 is mounted.
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
3904 inodes, 15600 blocks
780 blocks (5.00%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
1952 inodes per group
Superblock backups stored on blocks:
        8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
# _
```

Figure 18: Make ext2fs on CF Card

- So far you have got a read and writeable Extended 2 Filesystem (ext2fs) on your CF card and it can be accessed by Linux that is located under /mnt/ide0.
- Try to create a new folder **test** on the CF card to test the ext2fs you created just now (refer to Figure 19).



```
#
#
#
#
#
#
# mount -n /dev/hda1 /mnt/ide0
 hda: hda1
 hda: hda1
# cd /mnt/ide0
# ls
lost+found  test
# mkdir testtest
# ls
lost+found  test        testtest
#
#
#
# cd test
# ls
ext2fstest
# more ext2fstest
<< ext2fstest >>

# _
```

Figure 19: Test the ext2fs on the CF Card

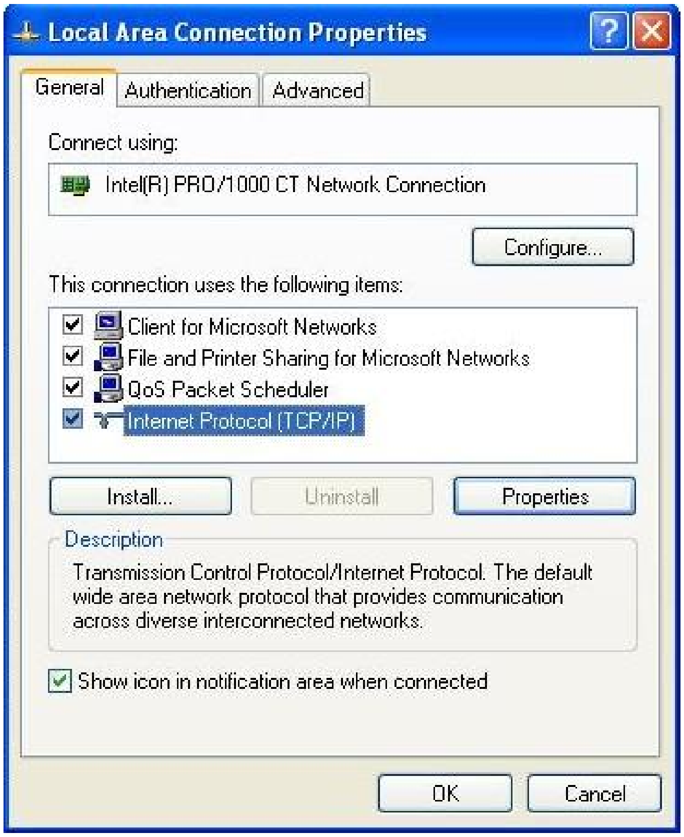
7.3 Configure the Network

In this section we are going to configure the network of both the target and the PC. The IP addresses used here are the IP addresses beginning with 192.168. that are usually used for hosts inside of a LAN. You can consult your network administrator for other IP addresses if you don't want to use those "fake" IP addresses.

The IP address for the PC here is 192.168.1.10 and the Netmask is 255.255.255.0.
The IP address for the target side is 192.168.1.20 and the Netmask is 255.255.255.0.

Please follow the steps below to make the networking configuration on the PC side.

1. To open the TCP/IP setting, select **Start | My Network Places**
2. Right-click on the **My Network Places** and select **Properties** from the drop-down menu.
3. The **Network Connections** window appears.
4. Right-click on the **Local Area Connection** and select **Properties** from the drop-down menu.
5. The Local Area Connection Properties window appears.
6. Select **Internet Protocol (TCP/IP)** in the **General** panel and click on the **Properties** icon under it (refer to Figure 20).



Connect using:

Intel(R) PRO/1000 CT Network Connection

This connection uses the following items:

☑ Client for Microsoft Networks
☑ File and Printer Sharing for Microsoft Networks
☑ QoS Packet Scheduler
☑ Internet Protocol (TCP/IP)

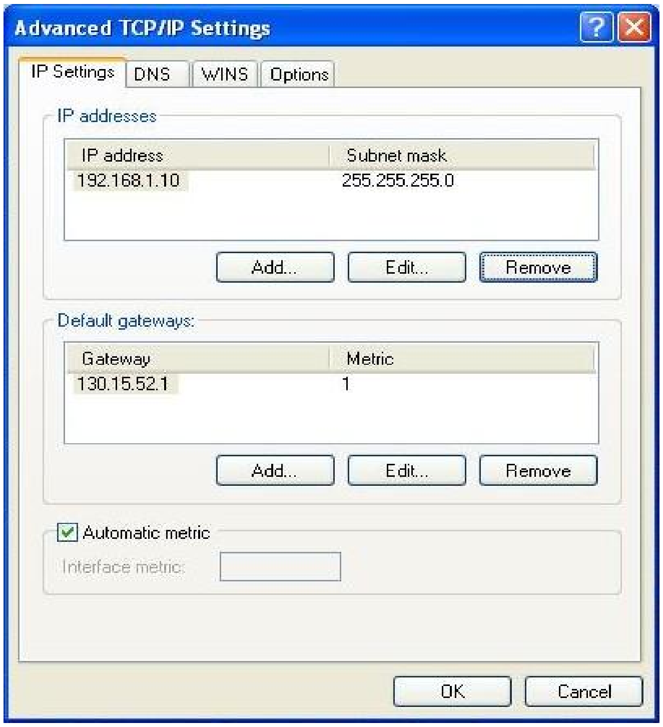
Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

☑ Show icon in notification area when connected

Figure 20: Local Area Connection Properties Window

7. The Internet Protocol (TCP/IP) Properties window appears.
8. Click on the icon **Advanced…** near the bottom of the window.
9. The **Advanced TCP/IP Settings** window appears.
10. Click on the **Add…** icon under the IP Addresses area in the IP Settings panel.
11. Add the IP address 192.168.1.10 and the netmask 255.255.255.0 to the IP Addresses (refer to Figure 21).



IP address	Subnet mask
192.168.1.10	255.255.255.0

Gateway	Metric
130.15.52.1	1

☑ Automatic metric

Interface metric:

Figure 21: Setting up IP Address

12. Click on the **OK** icon to exit the IP address-setting procedure.

Follow the instructions below to configure the network of the target:

- At the Linux command prompt, type:
  busybox ifconfig eth0 192.168.1.20 netmask 255.255.255.0

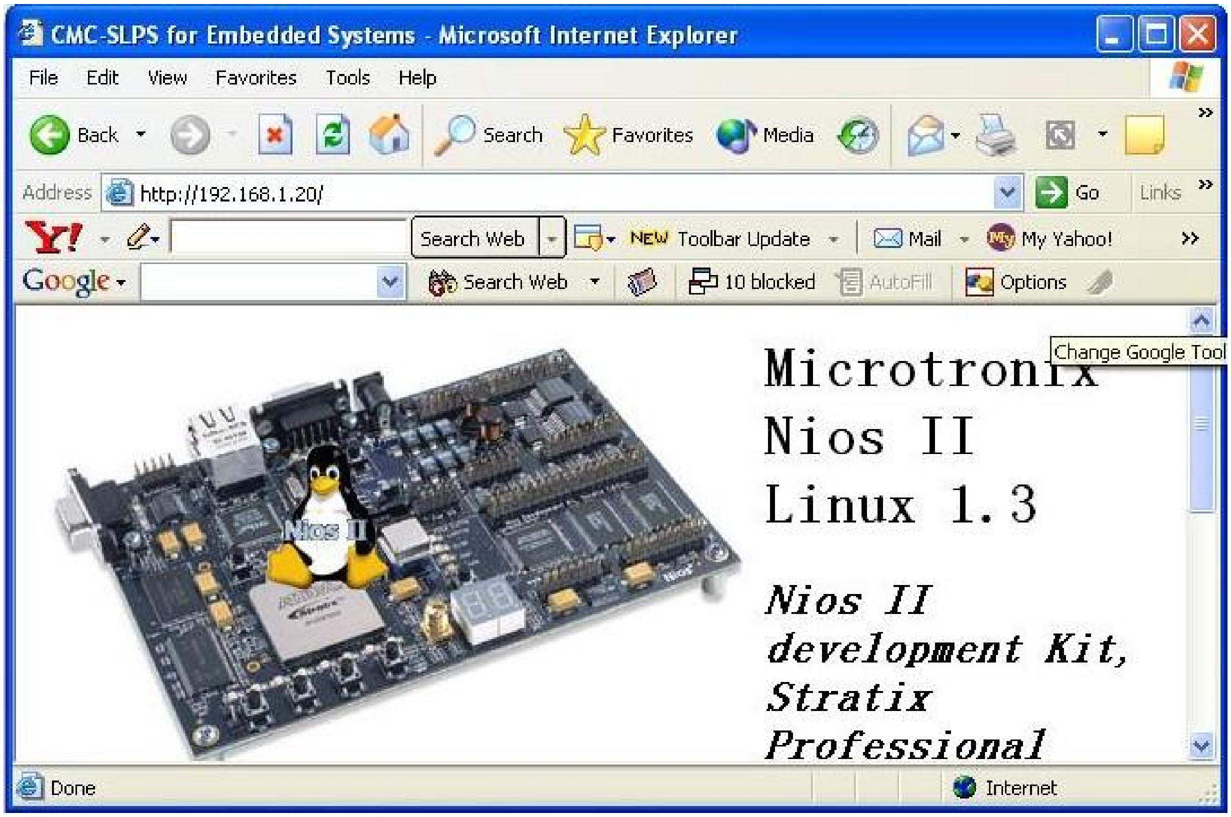
- Test the networking configuration by typing the following command:
  ping 192.168.1.10

If the networking configuration is properly set, you should see the response from the PC with the IP address 192.168.1.10.

Please follow the instruction below to start the web sever **boa** on the target board

- At the Linux command prompt, type:
  boa &

Now you have the network configured on both the PC and the target board. The web server **boa** also has been invoked on the board. To test the networking and the web server function, open a browser like Internet Explorer on the PC, and type 192.168.1.20 in the Address field, press the Enter key, then you will be able to access the web page pre-built on the target board, as shown in Figure 22. You can also create you own web page to replace the existing one in the file system project in the Nios II IDE.



Microtronix
Nios II
Linux 1.3

Nios II development Kit, Stratix Professional

Figure 22: Accessing Web Page from the Board

You have completed the tutorial: you have modified the full_featured platform to make it support the CompactFlash card and created the Linux kernel and file system projects and configured them to support networking function, a web server application and write/read mode to the file system of the embedded Linux.

For help with your own projects using embedded Linux and the SLPS, you may want to consult other users through the discussion forum accessible via CMC’s Technology Gateway: https://www1.cmc.ca/clients

Appendix A: CompactFlash Pinout Table

Pin on CF	CF Function	Connect to Pin on FPGA	Single Name	Pin Type
1	GND	GND		
2	D03	M4	data[3]	I/O
3	D04	N6	data[4]	I/O
4	D05	N1	data[5]	I/O
5	D06	N9	data[6]	I/O
6	D07	P3	data[7]	I/O
7	CE	J2	cs_n[0]	O
8	A10	M7	addr[10]	O
9	OE	K7	atasel_n	O
10	A09	K3	addr[9]	O
11	A08	H3	addr[8]	O
12	A07	L7	addr[7]	O
13	VCC	H4		
14	A06	L8	addr[6]	O
15	A05	H2	addr[5]	O
16	A04	H1	addr[4]	O
17	A03	L6	addr[3]	O
18	A02	L10	addr[2]	O
19	A01	J3	addr[1]	O
20	A00	L9	addr[0]	O
21	D00	N3	data[0]	I/O
22	D01	L2	data[1]	I/O
23	D02	N8	data[2]	I/O
24	WP	K4		
26	CD1	R3	detect	
27	D11	M3	data[11]	I/O
28	D12	N7	data[12]	I/O
29	D13	L1	data[13]	I/O
30	D14	N4	data[14]	I/O
31	D15	L3	data[15]	I/O
32	CE2	K8	cs_n[1]	O
33	VS1	GND		
34	OIORD	M9	iord_n	O
35	IOWR	M10	iowr_n	O
36	WE	L5	we_n	O
37	RDY/BSY	M5	intrq_n	I
38	VCC	H4		
39	CSEL	GND		
40	VS2	no connect		
41	RESET		reset_n	
42	WAIT	K1	iordy_n	I
43	INPACK	J4		

44		REG		G2	rfu	O
45		BVD2		J1		
46	BVD1		M8			
47	D081		N10		data[8]	I/O
48	D091		M2		data[9]	I/O
49	D101		N5		data[10]	I/O