Using Embedded Linux with Nios II Processor
User Guide

BSP Version: 0.1.0.0
Document Version: 1.1
Document Date: 03 January 2011
About this Guide

Introduction

This document explains how to create your own Nios II processor system for Linux and run a free, open source Linux distribution on a pre-built system.

Table below shows the revision history of the user guide.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>03 January 2011</td>
<td>Second Release.</td>
</tr>
<tr>
<td>1.0</td>
<td>September 2010</td>
<td>First Release.</td>
</tr>
</tbody>
</table>

How to Contact SLS

For the most up-to-date information about SLS products, go to the SLS worldwide website at http://www.slscorp.com. For additional information about SLS products, consult the source shown below.

<table>
<thead>
<tr>
<th>Information Type</th>
<th>E-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product literature services, SLS literature services, Non-technical customer services, Technical support.</td>
<td><a href="mailto:support@slscorp.com">support@slscorp.com</a></td>
</tr>
</tbody>
</table>
Typographic Conventions

The document uses typographic conventions shown as below.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold Type with Initial Capital Letters</td>
<td>All Headings and Sub Headings Titles in a document are displayed in bold type with initial capital letters; Example: Overview, Development Environment</td>
</tr>
<tr>
<td>Bold Type with Italic Letters</td>
<td>All Definitions, Figure and Table Headings are displayed in Italics. Examples: Figure 1-1. Development Environment</td>
</tr>
<tr>
<td>1. 2.</td>
<td>Numbered steps are used in a list of items, when the sequence of items is important such as steps listed in the procedure.</td>
</tr>
<tr>
<td>•</td>
<td>Bullets are used in a list of items when the sequence of items is not important.</td>
</tr>
<tr>
<td><img src="hand.png" alt="hand" /></td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td><img src="caution.png" alt="caution" /></td>
<td>The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.</td>
</tr>
<tr>
<td><img src="warning.png" alt="warning" /></td>
<td>The warning indicates information that should be read prior to starting or continuing the procedure or processes.</td>
</tr>
<tr>
<td><img src="feet.png" alt="feet" /></td>
<td>The feet direct you to more information on a particular topic.</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Configuring the Kernel</td>
<td>26</td>
</tr>
<tr>
<td>Linux Distribution Configuration</td>
<td>26</td>
</tr>
<tr>
<td>Linux Kernel Configuration</td>
<td>29</td>
</tr>
<tr>
<td>Device Drivers Configuration</td>
<td>42</td>
</tr>
<tr>
<td>Memory Technology Device (MTD) support</td>
<td>43</td>
</tr>
<tr>
<td>SCSI Device Support</td>
<td>44</td>
</tr>
<tr>
<td>Network Device Support</td>
<td>46</td>
</tr>
<tr>
<td>I2C Support</td>
<td>49</td>
</tr>
<tr>
<td>SPI Support</td>
<td>52</td>
</tr>
<tr>
<td>Input Device Support</td>
<td>54</td>
</tr>
<tr>
<td>PS2 Keyboard Support</td>
<td>56</td>
</tr>
<tr>
<td>Altera Touchscreen Support</td>
<td>58</td>
</tr>
<tr>
<td>Character Devices</td>
<td>60</td>
</tr>
<tr>
<td>Configuring JTAG UART</td>
<td>60</td>
</tr>
<tr>
<td>Configuring PIO buttons</td>
<td>64</td>
</tr>
<tr>
<td>Graphics Support</td>
<td>65</td>
</tr>
<tr>
<td>USB Host Support</td>
<td>68</td>
</tr>
<tr>
<td>SD Card Support</td>
<td>72</td>
</tr>
<tr>
<td>File System</td>
<td>75</td>
</tr>
<tr>
<td>VFAT File System Support &amp; JFFS2 File System Support</td>
<td>75</td>
</tr>
<tr>
<td>Configuring JFFS2 File System</td>
<td>78</td>
</tr>
<tr>
<td>Network File System Support</td>
<td>80</td>
</tr>
<tr>
<td>Compiling the kernel</td>
<td>95</td>
</tr>
<tr>
<td>Running the BSP</td>
<td>96</td>
</tr>
<tr>
<td>Applications On Running BSP</td>
<td>98</td>
</tr>
<tr>
<td>Mounting VFAT on SD-Card</td>
<td>99</td>
</tr>
<tr>
<td>Mounting a JFFS2 File System</td>
<td>101</td>
</tr>
<tr>
<td>Input Devices Applications</td>
<td>102</td>
</tr>
<tr>
<td>Touch Panel</td>
<td>103</td>
</tr>
<tr>
<td>PS2 Keyboard</td>
<td>106</td>
</tr>
<tr>
<td>Button PIO</td>
<td>108</td>
</tr>
<tr>
<td>I2C Applications</td>
<td>110</td>
</tr>
<tr>
<td>I2C Detect</td>
<td>110</td>
</tr>
<tr>
<td>I2C EEPROM Read and Write</td>
<td>111</td>
</tr>
<tr>
<td>I2C Audio Controller</td>
<td>112</td>
</tr>
<tr>
<td>TFTP Applications</td>
<td>113</td>
</tr>
<tr>
<td>TFTP Client</td>
<td>113</td>
</tr>
</tbody>
</table>
TFTP Server.............................................................................................................................................. 114
TELNET Application.................................................................................................................................. 115
BOA Application .......................................................................................................................................... 115
FTP Application.......................................................................................................................................... 115
Dropbear Application................................................................................................................................. 115
LCD Application.......................................................................................................................................... 116
1. Getting Started

Overview

This tutorial is designed to make you aware of the usage of Linux in Embedded Systems and its advantages.

FPGAs are highly flexible development platforms for custom embedded systems. Using Altera tools, any combination of hardware designs that includes the Nios II processor and a set of standard as well as custom peripherals can be created. Running Linux on such a customized environment is beneficial but can be a bit challenging if not given a proper start. It is therefore recommended that embedded developers always start with a standard hardware reference platform.

For BSP developers supporting custom hardware designs, the best place to start is the sample BSP provided in the training. As incremental changes are made to the hardware system, you can modify the factory BSP in lock-step, and upgrade your Linux kernel accordingly. It is recommended that all BSP development and enhancements begin with the factory BSP and built upon incrementally.

We assume that you are familiar with the Nios II, Linux and StratixIV Development Board.

You will learn here the following:

1. Development Environment Setup
2. Designing a Nios II Hardware Reference Design
3. Compiling and Running Linux with BSP
4. Creating User Application
5. Configuring Linux Kernel

Development Environment

Nios II embedded development environment consists of two systems are:

1. Host system: Host system is used for compiling, linking, remote debugging and associated development activities.
2. Target system: Target system is used for such as the Stratix IV GX FPGA Development Kit, application development and testing (Figure 1-1.). Board acts as a target for application development. User must have NEEK board and Terasic THDB-SUM board for testing different IPs connected using HSMC PORTA and PORTB respectively to target board.

Figure 1-1. Development Environment

Development Host

A PC with Linux OS acts as a development host. It must have the following software installed:

- Linux for Nios II processor development software
  The Linux tool chain for the Nios II processors were tested against Fedora core10 and CentOS 5.3 software. We recommend that you start with these desktop software versions. Alternatively you can try another Linux versions.
  http://www.centos.org/docs/5/
  http://docs.fedoraproject.org/installation-quick-start-guide/

Following development packages must needed on your Development Host, git-all, git-gui, tcsh, make, gcc, ncurses-devel,bison, libglade2-devel, byacc, flex, gawk, get-text, ccache, zlib-devel, gtk2-devel, lzo-devel, pax-utils
Getting Started

For FPGA configuration flash programming and host-target communication using the Altera USB Blaster, you need to install the driver for the Altera USB Blaster. To install the USB-Blaster driver on Linux, follow the steps from below link.


Plug one end of a USB cable to the USB port on the Altera Stratix IV GX FPGA Development Kit and other end to a USB port on the Linux host to access on-board USB-Blaster. Type the following command to verify that the USB-Blaster is working properly. Wiki Web site is located at:

http://www.nioswiki.com/OperatingSystems/UClinux/QuartusforLinux

#jtagconfig

1. The console displays the devices connected to the USB port as shown below:

   1) USB-Blaster [USB 4-1.1]
   024090DD   EP4SGX230/ES
   020A40DD   EPM2210

   The syntax may vary for different Linux distributions.

Development Target

The Stratix IV GX FPGA Development Kit is used as a Development Target.

Configuring the Development Board

To configure the development board, check all the switches are in default position. If not, then follow the steps below:

1. Set Rotary Switch SW2 at ‘0’ position.

2. Set all switches of user DIP switch bank SW3 in (OFF) ‘1’ position.
System Setup

3. Set switches 1, 2, 4 in (OFF) ‘1’ position and remaining switches in (ON) ‘0’ position of board setting switch SW4.

4. Set switch 4 in (OFF) ‘1’ position and remaining in (ON) ‘0’ position of PCIe switch SW5.

5. Set switch 1 in (OFF) ‘1’ position and remaining in (ON) ‘0’ position of JTAG switch SW6.

This section explains hardware and software required and the system setup to run Linux on the Nios II processor. See Figure 1-1.

Follow the steps below to make the system setup:

1. Connect Stratix IV GX FPGA Development Kit to a 100/1000 Mbps Ethernet switch.
   The host PC should be connected to the aforementioned Nios II target through the Ethernet switch.

2. Connect one end of the standard USB Cable to the host Linux PC and the other end to the Stratix IV GX FPGA Development Kit.

Download the bsp-lnx-s4gxdk-110103-0.1.0.0.tar.bz2 from http://www.slscorp.com/pages/bsplnxs4gxdk.php

### Table 1-1. BSP Contents

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>v2.6.34</td>
</tr>
<tr>
<td>GCC</td>
<td>v4.1.2</td>
</tr>
<tr>
<td>Ethernet Driver</td>
<td>Included</td>
</tr>
<tr>
<td>JTAG Driver</td>
<td>Included</td>
</tr>
<tr>
<td>Serial port Driver</td>
<td>Included</td>
</tr>
<tr>
<td>LED Driver</td>
<td>Included</td>
</tr>
<tr>
<td>Push Button Driver</td>
<td>Included</td>
</tr>
<tr>
<td>PS2 Keyboard Driver</td>
<td>Included</td>
</tr>
<tr>
<td>LCD Driver</td>
<td>Included</td>
</tr>
<tr>
<td>Touch Panel Driver</td>
<td>Included</td>
</tr>
<tr>
<td>USB Host 2.0 Driver</td>
<td>Included</td>
</tr>
<tr>
<td>I2C Driver</td>
<td>Included</td>
</tr>
</tbody>
</table>
### Table 1-1. BSP Contents

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFFS2 and VFAT Driver</td>
<td>Included</td>
</tr>
<tr>
<td>SD Card Driver</td>
<td>Included</td>
</tr>
</tbody>
</table>
2. Designing a Nios II Hardware Reference Design

Introduction

This section describes how to create a Nios II hardware reference design on Altera Stratix IV GX FPGA Development Kit. The board, when configured as a Nios II target, will boot and run Linux and allow host-target communication and Flash programming over USB cable. The Linux Host should have Nios II processor development package installed. Figure 2-1. below shows the setup.

Figure 2-1. Hardware Setup

The Nios II Target, the Altera Stratix IV GX FPGA Development Kit has the following key components:

- **Flash Memory**
  Once the on-board Flash memory is programmed with the FPGA configuration image for the Nios II hardware reference design, Stratix IV Edition, the option bits for the MAX II configuration controller and a prebuild kernel image with initramfs; the development board on power up will boot up as a Nios II target running Linux.

- **USB Interface**
  For host-target communication and high-speed Flash programming.

For more information on the Altera Stratix IV GX FPGA Development Kit refer to the documentation at: http://www.altera.com/products/devkits/altera/kit-siv-gx.html
Creating Hardware Design

Here, we have provided the sample System for Stratix IV GX FPGA Development Kit.

Using the SOPC Builder tool, create a minimum processor system design that includes the following features.

Please consult on-line documentation from www.altera.com on how to use the SOPC Builder tool.

Our example system includes the following features:

- Nios II/f core
- Hardware multiplier
- MMU, use the default MMU settings
- 1K dual-port tightly coupled memory, connect one port to the tightly_coupled_instruction_master of Nios II and the other port to the tightly_coupled_data_master
- Assign "Fast TLB Miss Exception Vector" to the aforementioned tightly coupled memory
- Add DDR3 or SDRAM to the system, you need a minimum of 8MB and a maximum of 128MB
- One full-featured timer, not a hi-res timer
- A JTAG/serial UART
- External Flash
- Ethernet controller
- LED and Button PIO
- LCD controller
- SLS SD Host controller
- Touch Panel controller
- SLS PS2 Keyboard controller
- SLS I2C master for EEPROM, Audio and TV
- SLS I2S controller
- USB Host controller(USB20HC)

The block diagram given below will make the design clearer. See Figure 2-2.
Important things to note while you’re creating the hardware design are:

- Note in Linux, irq 0 means auto-detected, so you must not use irq 0 for ANY devices, except for the timer.
- Component naming is critical. They must match with the macro defined in your kernel. Please check the kernel source files below to make sure:
  - `/home/sls/Nios2-linux/Linux_source/linux-2.6/arch/nios2/boards/4s230/config.c`
  - `/home/sls/Nios2-linux/Linux_source/linux-2.6/arch/nios2/boards/4s230/include/asm/nios.h`

Memory Map and Linker Regions

The memory map of the Nios II processor system and the Linker sections are shown in Table 2-1 and Table 2-2 respectively.
All addresses that fall in the range 0x00000000 to 0x1FFFFFFF are direct mapped while addresses from 0x20000000 and above are managed by the Memory Management Unit (MMU). In order to optimize for fast system performance, the base addresses of all peripherals are mapped outside of the area managed by the MMU.

It is recommended that you allocate your user peripherals in the direct mapped memory range (0x00000000 to 0x1FFFFFFF). It is also recommended that you retain the memory allocations for the peripherals provided to you as part of the Nios II Hardware Reference Design for Linux, Stratix IV Edition.

<table>
<thead>
<tr>
<th>SR. No.</th>
<th>Device Name</th>
<th>Device Name in the Design</th>
<th>Address Range</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External Flash Memory</td>
<td>ext_flash</td>
<td>0x00000000-0x3FFFFFFF</td>
<td>67108864</td>
</tr>
<tr>
<td>2</td>
<td>Descriptor Memory</td>
<td>descriptor_memory</td>
<td>0x4000000-0x4001FFFFF8192</td>
<td>8192</td>
</tr>
<tr>
<td>3</td>
<td>Triple Speed Ethernet</td>
<td>MACtse_mac</td>
<td>0x4002000-0x40023FF</td>
<td>1024</td>
</tr>
<tr>
<td>4</td>
<td>Receive Scatter Gather DMA</td>
<td>sgdma_rx</td>
<td>0x4002400-0x400243F</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>Transmitter Scatter Gather DMA</td>
<td>sgdma_tx</td>
<td>0x4002440-0x400247F</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>TimerLCD lcd_sgdma</td>
<td>timer_1ms</td>
<td>0x4002480-0x40024BF64</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>LCD</td>
<td>lcd_sgdma</td>
<td>0x40024C0-0x40024FF</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>SLS USB 2.0 Host (USB20HC)</td>
<td>sls_usb20hc</td>
<td>0x4C00000-0x4C03FFF</td>
<td>16384</td>
</tr>
<tr>
<td>9</td>
<td>SLS USB20HC PHY RESET</td>
<td>usb20hc_phy_reset</td>
<td>0x4C04000-0x4C0401F</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>LED PIO</td>
<td>led_pio</td>
<td>0x4E00000-0x4E0001F</td>
<td>32</td>
</tr>
<tr>
<td>11</td>
<td>Button PIO</td>
<td>button_pio</td>
<td>0x4E00020-0x4E0003F</td>
<td>32</td>
</tr>
</tbody>
</table>
### Table 2-1. Memory Section Map

<table>
<thead>
<tr>
<th>SR. No.</th>
<th>Device Name</th>
<th>Device Name in the Design</th>
<th>Address Range</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>SLS I2C Master EEPROM</td>
<td>sls_i2c_m_id_eeprom</td>
<td>0x4E000080-0x4E000FF</td>
<td>128</td>
</tr>
<tr>
<td>13</td>
<td>SLS SD Host controller</td>
<td>sls_sdhc</td>
<td>0x4E001000-0x4E001FF</td>
<td>256</td>
</tr>
<tr>
<td>14</td>
<td>SLS PS2 controller</td>
<td>sls_ps2</td>
<td>0x4E002000-0x4E0023F</td>
<td>64</td>
</tr>
<tr>
<td>15</td>
<td>Touch Panel SPI</td>
<td>touch_panel_spi</td>
<td>0x4E002400-0x4E0027F</td>
<td>64</td>
</tr>
<tr>
<td>16</td>
<td>Touch Panel PEN</td>
<td>touch_panel_penIRQ_n</td>
<td>0x4E002800-0x4E0029F</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>SLS I2C Master Audio &amp; TV</td>
<td>sls_i2c_m_aud_tv</td>
<td>0x4E003000-0x4E0037F</td>
<td>128</td>
</tr>
<tr>
<td>18</td>
<td>SLS I2S controller</td>
<td>sls_i2s</td>
<td>0x4E003800-0x4E003BF</td>
<td>64</td>
</tr>
<tr>
<td>19</td>
<td>JTAG</td>
<td>jtag_uart</td>
<td>0x4EFFFB0-0x4EFFFB</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>UART</td>
<td>uart</td>
<td>0x4EFFFC0-0x4EFFFF</td>
<td>64</td>
</tr>
<tr>
<td>21</td>
<td>TLB_MISS_RAM 1K Memory</td>
<td>tlb_miss_ram_1k</td>
<td>0x7FFF400-0x7FFF7FF</td>
<td>1024</td>
</tr>
<tr>
<td>22</td>
<td>DDR3 SDRAM controller</td>
<td>ddr3_top</td>
<td>0x80000000-0xFFFFFFF</td>
<td>134217728</td>
</tr>
</tbody>
</table>

### Table 2-2. Linker Section Map

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Linker Section Name</th>
<th>Linker Region Name</th>
<th>Memory Device</th>
<th>Memory Device Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.bss</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
<tr>
<td>2</td>
<td>.exceptions</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
<tr>
<td>3</td>
<td>.heap</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
</tbody>
</table>
Table 2-2.  Linker Section Map

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Linker Section Name</th>
<th>Linker Region Name</th>
<th>Memory Device</th>
<th>Memory Device Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>.rodata</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
<tr>
<td>5</td>
<td>.rwdata</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
<tr>
<td>6</td>
<td>.stack</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
<tr>
<td>7</td>
<td>.text</td>
<td>ddr2_lo_latency_128m</td>
<td>DDR2 SDRAM</td>
<td>ddr2_lo_latency_128m</td>
</tr>
</tbody>
</table>

Compile the Hardware Design

Please consult the *Altera user documentation for Quartus II software* and the *SOPC Builder tool* for information on how to create and compile a new hardware design.
3. Compiling and Running
Linux with BSP

Introduction

Nios II Hardware Reference Design by SLS for Stratix IV GX FPGA Development Kit and the matching BSP provide a solid starting point for BSP Development. It is recommended that you always start with the sample BSP, when you create new device drivers or make iterative changes to the provided device drivers as hardware changes are made in the system.

BSP

The BSP (Board Support Package) contains the following:
Quick reference with ready to go pre-built Linux images and SOF

1. Linux Image(with initramfs) without USB2.0 Host controller IP
2. Linux Image(with initramfs) with USB2.0 Host controller

To use Linux Image with USB2.0 Host controller this image Terasic THDB-SUM board HSMC must be connected to Stratix IV board's HSMC PORT B.

- Supported and tested Devices/Peripheral Drivers
  - Ethernet: Altera TSE driver (SLS)
  - Flash: Intel CFI Parallel Flash
  - Serial: Altera JTAG UART, Altera Serial UART
  - PIO: LEDs and Push Button Switches
  - SD Card : SD Host controller driver (SLS)
  - LCD: Altera LCD driver
  - Touch Panel: Altera Touch Panel driver (SPI based)
  - PS2 Keyboard:PS2 Keyboard driver (SLS)
  - I2C Master : I2C Master driver for EEPROM and Audio & TV (SLS)
  - USB 2.0 Host: USB20HC controller driver (SLS)
  - I2S Audio controller (SLS) driver (not added)

Configuring the BSP

The package downloaded earlier from www.slscorp.com is to be used here. Please follow the steps mentioned below:

1. Copy the BSP source bsp-lnx-s4gxdk-110103-0.1.0.0.tar.bz2
   at the development folder on your linux PC and extract it.
#cd /home/sls/
#tar -xjf bsp-lnx-s4gxdk-110103-0.1.0.0.tar.bz2

The **Nios2-Linux** folder will be created. It contains following three folders.

<table>
<thead>
<tr>
<th>Directory Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildTools</td>
<td>Contains pre-built bin tools gcc 4.1.2 for nios2-linux</td>
</tr>
<tr>
<td>Linux_source</td>
<td>Contains kernel and application</td>
</tr>
<tr>
<td>System-Board</td>
<td>Contains system file for specific board. It contains only for 4SGX230 board files</td>
</tr>
</tbody>
</table>

2. Set the Bintools path on your terminal.

   
   #PATH=$PATH:/home/sls/Nios2-Linux/BuildTools/tool-chain-mmu/x86-linux2/bin

3. Build the Linux image.

   #cd/home/sls/Nios2-Linux/Linux_source/uClinux-dist/
   #make menuconfig
   
   The menuconfig screen displays as shown in **Figure 3-1**.

   **Figure 3-1. Menu Configuration Screen**

4. Select **Vendor/Product Selection**. See **Figure 3-2**.
5. Select Vendor (vendor_name) and make sure that Altera is selected as shown in Figure 3-3. To select/de-select the vendor, highlight the vendor name (using arrow keys) and press space-bar or Enter to select or de-select.

Figure 3-3. Vendor selection

6. Select **Altera Products** (product_name) to select the product. See Figure 3-4.
7. Select nios2. See Figure 3-5.

Figure 3-4. Vendor/Product Selection

Figure 3-5. Altera Product Selection

8. Press E to exit the Vendor/Product Selection section.

9. Press E again to exit the kernel configuration. You will be asked whether to save the configuration or not. See Figure 9

10. Press E again to exit the kernel configuration.
Compiling the BSP

To compile the BSP, follow the steps below:

1. Type the following command to compile the BSP:
   
   ```
   #make
   ```

   After compilation, you will get different images in the image folder located at:

   ```
   /home/sls/Nios2-linux/Linux_source/uClinux-dist/images/
   ```

   The `linux.initramfs.gz` file is an elf image with initramfs.

Running the BSP

To run the BSP on Nios II reference design, follow the steps below:

1. Download the sof file `sys_qii100sp1_linux_bsp_s4gxdb.sof` located at
   ```
   /home/sls/Nios2-linux/System-Board/4s230_default.
   ```

2. Download elf file `linux.initramfs.gz` located at
   ```
   /home/sls/Nios2-linux/Linux_source/uClinux-dist/images/
   ```

3. Download the ELF image using the following command:
   ```
   #nios2-download -g linux.initramfs.gz
   ```

4. After successful downloading of SOF and ELF, Linux terminal displays the results as shown in Figure 3-6.

5. Type the following command to open the Nios II terminal.
   ```
   #nios2-terminal
   ```

   Now, this is the embedded Linux running on the 4SGX230 FPGA. We have downloaded the hardware design with the Nios II processor first and then downloaded the image with the kernel and drivers. See Figure 3-7.
Current Kernel configuration does not include support for USB20 Host Controller. **Stratix IV HSMC PORT A** should be connected with **NEEK** board.

**Figure 3-7. Running Linux On the Board**

6. Type `ls` to see the directory contents. Similarly we can use the commands like `cd`, `password` and other in the same way as we use in Linux. See **Figure 3-8**.

**Login:**

*Username:* root  
*Password:* nios2linux
If the ethernet cable is connected to a network, we can also view the status, assign IP Address to the board and access other machines in the network as mentioned in the following steps. See Figure 3-9.

7. Type the following command to view the status.
   
   `ifconfig eth0`

Figure 3-9. Eternet Configuration Status

---

8. Type the following command to assign IP address to the 4SGX230 board.
   
   `ifconfig eth0 192.168.0.181`

Figure 3-10. Assigning IP Address

---

The IP address assigned above is only for example. Please ask your instructor to get the IP address to be assigned to 4SGX230 board.
9. Type the following command to access other machine in the network.
   ping 192.168.0.41 -c 5

Figure 3-11. Accessing Other Machine in the Network

10. Please consult your instructor to get the IP address of other machine in the network.

    We have learned how to run the given BSP on the board. The next chapters will explain how to create your own application and modify kernel settings.
4. Creating User Application

This section explains you about adding a user application named hello in the BSP. This application prints **hello world** on the Nios II terminal. Follow the steps below to add a new user application.

1. Open **Linux** terminal.

2. Locate the directory **sls_test_app** from **/home/sls/Nios2-Linux/Linux_source/uClinux-dist/user/sls_test_app** directory.

3. Type following to create **hello.c** file.
   
   vi hello.c

4. Type the following code in the file.
   
   ```c
   #include <stdio.h>

   int main()
   {
   
   printf ("\n\nHello World! \n\n");
   
   return 0;
   }
   ```
5. Modify the **Makefile** as mentioned below to compile the hello application.

Type the following command to open the Makefile.

```
vi Makefile
```

The user application and the object file are defined by the macros EXEC_USER and EXEC_OBJS respectively. See Figure 4-2.
6. Locate the folder **uClinux-dist** from `/home/sls/Nios2-Linux/Linux_source`.

7. Type the following command to compile the BSP:
   
   ```bash
   #make
   ```

   After compilation, you will get different images in the image folder located at:
   `/home/sls/Nios2-linux/Linux_source/uClinux-dist/images/`

   The **linux.initramfs.gz** file is an elf image with initramfs.

8. Make sure that the **SOF** file is downloaded.

9. Download the **ELF** image using the following command:
   
   ```bash
   #nios2-download -g linux.initramfs.gz
   ```

10. After successful downloading of SOF and ELF, Linux terminal displays the results as shown in **Figure 4-3**.
11. Type the following command to open the Nios II terminal.

```
# nios2-terminal
```

Now, this is the embedded Linux running on the 4SGX230 FPGA. We have downloaded the hardware design with the Nios II processor first and then downloaded the image with the kernel and drivers.

12. Type `ls` to see the directory contents. Similarly, we can use the commands like `cd`, `password` and other in the same way as we use in Linux. See Figure 4-5.
Figure 4-5. Running ls Command

13. Type the following command to locate the **hello application** in the **bin** folder.
   ```
   cd bin
   ```

14. Type the following to run the application.
   ```
   hello
   ```

15. The message “Hello World!” will be displayed on the terminal. See Figure 4-6.
Now you have learned how to create your own custom application. You can go back and modify your application, compile the kernel again and download the modified image again to run your custom application. The next chapter will explain you about modifying the kernel settings.
5. Customizing the Kernel

Generate a System Header File

Your hardware design has fixed peripheral component base addresses, which the Linux device drivers access through a static header file called *custom_fpga.h*. This file must be regenerated manually, each time the system memory map changes.

When you make any changes to the hardware design using the SOPC Builder tool, it automatically generates a `.sopcinfo` file after you recompile the hardware design. The `.sopcinfo` file contains information on the hardware design, including the system memory map. You must manually run the `sopc-create-header-files` command on the `.sopcinfo` file in order to generate the `custom_fpga.h`.

You can learn more about the `sopc-create-header-files` with the --help option from the Nios II Command Shell as shown below:

Follow the steps below to generate a System Header file:

1. Locate the `.sopcinfo` file from
2. Type the following command to create `custom_fpga.h` file.
   
   ```bash
   sopc-create-header-files --single custom_fpga.h
   ```
3. Type following command to copy the `custom_fpga.h` file to asm folder.
   
   ```bash
   cp custom_fpga.h /home/sls/Nios2-linux/Linux_source/linux-2.6/arch/nios2/boards/4s230/include/asm
   ```

Configuring the Kernel

To configure the kernel, follow the steps mentioned below.

Linux Distribution Configuration

1. Set the Bintools path on your terminal.
   
   ```bash
   #PATH=$PATH:/home/sls/Nios2-Linux/BuildTools/toolchain-mmu/x86-linux2/bin
   ```
2. Build the Linux image.
   
   ```bash
   #cd /home/sls/Nios2-linux/Linux_source/uClinux-dist/
3. Type the following command to modify kernel settings.
   
   ```
   #make menuconfig
   ```
   
   The uClinux Distribution Configuration dialog box opens. See Figure 5-1.

   **Figure 5-1. uClinux Distribution Configuration Dialog Box**

4. Press ↓ and select Kernel/Library/Defaults Selection.

5. Press Enter.

6. Kernel/Library/Defaults Selection dialog box appears. See Figure 5-2.
Figure 5-2. Kernel/Library/Defaults Selection

7. Select the following options: See Figure 5-3.
   - Customize Kernel Settings
   - Customize Application/Library Settings

Figure 5-3. Kernel/Library/Defaults Selection Configuration
8. Press E to exit.
9. You will return to the uClinux Distribution Configuration dialog box.
10. Press E to exit.
11. The Save dialog box opens.
12. Press Y to save the configuration. See Figure 5-4.

**Figure 5-4. Saving Linux Distribution Configuration**

13. The Linux Kernel Configuration window opens. See Figure 5-5.

**Linux Kernel Configuration**

14. Select the following options:
   - Enable loadable module support
   - Enable the block layer
   - Networking support
15. Select **NiosII Configuration**.

16. Select **Nios II board configuration**. See Figure 5-6.
17. Select board configuration (4S230 dev board). See Figure 5-7.
18. Select 4S230 dev board. See Figure 5-8.
19. Press **Enter**. You will return to **Nios II board configuration** dialog box.

20. Select **FPGA configuration (CUSTOM_FPGA)**. See **Figure 5-9**.
21. Select CUSTOM_FPGA. See Figure 5-10.
22. Press Enter. You will return to Nios II board configuration dialog box.

23. Press <Esc> <Esc>, you will return to NiosII configuration page.

24. Select Additional NiosII Device Drivers. See Figure 5-11.
25. Select following options: See Figure 5-12.

- Enable NiosII PIO driver
- Enable NiosII PIO LED driver
26. Press **Enter**.

27. Press **<Esc>**<Esc>, you will return to **NiosII configuration** page. See **Figure 5-6**.

28. Select **NiosII specific compiler options**. See **Figure 5-13**.
29. Select **Enable MUL instruction**. See Figure 5-14.
30. Press <Esc> <Esc>, you will return to NiosII configuration page. See Figure 5-6.

31. Press <Esc> <Esc>, you will return to Linux Kernel configuration page. See Figure 5-5.

32. See Figure 5-15. Select Networking Support.
33. The Networking support dialog box opens. See Figure 5-16.

34. Select Networking Options
35. Press Enter.

36. The Networking Options dialog box opens. See Figure 5-17.

37. Select the following options:
   - Packet socket
   - Packet socket: mmapped IO
   - Unix domain sockets
   - TCP/IP networking
   - IP: kernel level autoconfiguration
   - IP: DHCP support
   - BOOTP support
   - INET: socket monitoring interface

**Figure 5-17. Networking Options (2)**
38. Press <Esc> <Esc>.

39. Press <Esc> <Esc>.

40. You will return to the Linux Kernel configuration page dialog box.

Device Drivers Configuration

See Figure 5-18.

Figure 5-18. Device Drivers

41. Select the following options. See Figure 5-19.

- Memory Technology Device (MTD) support
- Block devices
- Network device support
- I2C Support
- SPI Support
- USB Support
Memory Technology Device (MTD) support

42. Select Memory Technology Device (MTD) support.

43. MTD support is used for JFFS2 File system to create Flash partitions. See Figure 5-20.
Figure 5-20. Memory Technology Device (MTD) Support (2)

SCSI Device Support

44. Needs for USB Storage device support. See Figure 5-21.
Figure 5-21. SCSI Device Support (1)

45. Select following options:

- SCSI device support
- SCSI disk support
- SCSI generic support  See Figure 5-22.
Network Device Support

46. The Device Drivers dialog box opens. See Figure 5-23.

47. Select Network device support.
48. Press Enter.

49. The Network Device Support dialog box opens. See Figure 5-24.

50. Select Ethernet (10 or 100Mbit).
51. Press Enter.

52. The Ethernet (10 or 100Mbit) dialog box opens. See Figure 5-25.

53. Select Altera Triple Speed Ethernet MAC support (SLS).

54. Press <Esc> <Esc>.
Press <Esc> <Esc>.
I2C Support

55. I2C support is used for I2C based EEPROM device and Audio and TV based on SLS I2C IP. See Figure 5-26.

56. Select I2C Support
Select **I2C Hardware Bus Support**. See Figure 5-27.
58. I2C Hardware Bus Support. Select **SLS I2C Master Controller.** See Figure 5-28.

59. Press `<Esc> <Esc>`.

60. Press `<Esc> <Esc>` to go Device Driver selection menu.
SPI Support

61. SPI support is used SPI based Touch Panel and Flash.

62. Select SPI Support. See Figure 5-29.
63. Select **Altera SPI Controller**. See Figure 5-30.

64. Press `<Esc> <Esc>` to go Device Driver selection menu.
Input Device Support

65. Support for input devices like **PS2 keyboard** and **Touch Panel controller**. See **Figure 5-31**.
Customizing the Kernel

Figure 5-31. Input Device Support

66. Select following options: See Figure 5-32.

- Generic input layer (needed for keyboard, mouse ...)
- Event interface
- Event debugging
PS2 Keyboard Support

67. Select Keyboards. See Figure 5-33.
68. Select SLS PS2 Keypad driver. See Figure 5-34.

69. Press <Esc> <Esc>.
Configuring the Kernel

Figure 5-34. SLS PS2 Keypad Driver

Altera Touchscreen Support

70. Select Touchscreens. See Figure 5-35.
Figure 5-35. Touchscreens

71. Select ADS7846/TSC2046/AD7873 and AD(S)7843 based touchscreens. See Figure 5-36.

72. Press <Esc> <Esc>.

73. Press <Esc> <Esc> to go Device Driver selection menu.
Configuring JTAG UART

74. The Device Drivers dialog box opens. Select Character devices. See Figure 5-37.
75. The **Character Devices** dialog box opens. See Figure 5-38.

- Select **Serial drivers**.
For **JTAG UART**, select the following options: See Figure 5-39.

- Altera JTAG UART support
- Altera JTAG UART console support
Customizing the Kernel

Figure 5-39. Serial Drivers Configuration

If you want to use UART instead of JTAG UART then select the following options: See Figure 5-40.

- Altera UART Support
- (4) Maximum number of Altera UART ports
- (115200) Default baudrate for Altera UART port
- Altera UART console support
Figure 5-40. Altera UART support

77. Press <Esc> <Esc>. You will return to Character devices dialog box.

Configuring PIO buttons

78. Select Nios PIO button support. See Figure 5-41.
Customizing the Kernel

**Figure 5-41. Configuring PIO**

79. Press `<Esc>` `<Esc>`.

80. Press `<Esc>` `<Esc>`.

81. Press `<Esc>` `<Esc>`.

82. Press Y to save the configuration settings.

83. You will return to Linux terminal.

**Graphics Support**

- LCD Support

84. Select Graphics Support. See Figure 5-42.
Figure 5-42. Graphics Support

85. Select Support for frame buffer devices. See Figure 5-43.
86. Select Altera LCD IP Support. See Figure 5-44.
87. Press <Esc> <Esc> to go Device Driver selection menu.

**USB Host Support**

88. USB Host drivers are supported by **SLS USB 2.0 Host controller IP**.

89. Select **USB Support**. See Figure 5-45.
Customizing the Kernel

Figure 5-45. USB Support (1)

90. Select Support for the **Host-side USB**. See Figure 5-46.
Figure 5-46. Host-side USB

91. Select SLS Embedded USB20HCv1.5 support. See Figure 5-47.
92. Select USB Mass Storage support. See Figure 5-48.

93. Press <Esc> <Esc> to go Device Driver selection menu.
SD Card Support

94. Depends on VFAT filesystem support.

95. Select MMC/SD/SDIO card Support. See Figure 5-49.
Figure 5-49. SD Card Support

96. Select MMC block device driver. See Figure 5-50.
97. Select **SD Host Controller (SLS)**. See Figure 5-51.

98. Press `<Esc> <Esc>` to go Device Driver selection menu.
Customizing the Kernel

Figure 5-51. SD Host Controller (SLS)

File System

VFAT File System Support & JFFS2 File System Support

Virtual File Allocation Table (VFAT) is a part of the Windows 95 and later operating system that handles long file names, which otherwise could not be handled by the original file allocation table (FAT) programming. VFAT file system is used with SD Card. Follow the steps below to configure the VFAT File system.

99. Press <Esc> <Esc>.

100. You will return to the Linux Kernel Configuration dialog box. See Figure 5-5.

- Select File systems. See Figure 5-52.
Figure 5-52. File System Selection

101. Press Enter.

102. The File Systems dialog box opens. See Figure 5-52.

103. Select the following options:
   - Enable POSIX file locking API
   - Dnotify support
   - Inotify file change notification support
   - Inotify support for userspace
   - Miscellaneous filesystems
   - Network File Systems

104. Press ↓ and select DOS/FAT/NT File systems. See Figure 5-53.
Figure 5-53. File Systems Configuration

105. Press Enter.

106. The **Dos/FAT/NT Filesystems** dialog box opens. See Figure 5-54. Select the following options:
   - **MSDOS fs support**
   - **VFAT (windows-95) fs support**
Figure 5-54. DOT/FAT/NT File Systems Settings

107. Press <Esc> <Esc>.

108. You will return to File systems dialog box.

Configuring JFFS2 File System

109. Select Miscellaneous filesystems. See Figure 5-55.
Customizing the Kernel

Figure 5-55. File System Dialog Box

110. Select following option: See Figure 5-56.
   - Journalling Flash File System v2 (JFFS2) support
   - JFFS2 write-buffering support
Network File System Support

NFS is a network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network as easily as if the network devices were attached to its local disks. If you want to use NFS file system on Ethernet then you have to configure the Ethernet IP driver and NFS file system. Ethernet IP driver is already configured. Follow the steps below to configure the NFS File system.

1. Press <Esc> <Esc>.
   - Select Networking Support. See Figure 5-57.
112. The **Network File Systems** dialog box opens. See Figure 5-58.

113. Select the following options:

- NFS client support
- NFS client support for NFS version 3
- NFS client support for NFSv3 ACL protocol extension
114. Press <Esc> <Esc>.

115. Press <Esc> <Esc>.

116. Now you will enter in **Customize Application/ Library Settings**. See Figure 5-59.

117. Select **Core Applications**. See Figure 5-60.
Customizing the Kernel

Figure 5-59. Library Configuration

![Library Configuration Image]
**Figure 5-60. Core Applications**

- Select **init**
- Select enable console shell
- Press `<Esc>` `<Esc>`

118. Select **Network Applications.** See Figure 5-61.
- Select **boa**
Customizing the Kernel

Figure 5-61. Network Applications

- Select Dropbear, Ethtool, FTPD, inetd See Figure 5-62.
Figure 5-62. Network Applications (Dropbear, Ethtool, FTPD, inetd)

- Select telnet and telnetd See Figure 5-63.
- Press <Esc> <Esc>
119. Select **Miscellaneous Applications** See Figure 5-64.

- Select **Test Applications (SLS)**
Figure 5-64. Miscellaneous Applications

- Select **i2c-tools** See Figure 5-65.
- Press <Esc> <Esc>
Figure 5-65. Miscellaneous Applications (i2c-tools)

- Select **BusyBox** See Figure 5-66.
Select Networking Utilities See Figure 5-67.
Customizing the Kernel

Figure 5-67. Networking Utilities BusyBox

- Select Hostname
- Select ifconfig, Enable status reporting output and Enable option “hw” See Figure 5-68.
**Figure 5-68. Enable T6v6 Support**

- Select ping, netstat, tftp, tftpd, udhcp client, uspsvd. See Figure 5-69.
- Press <Esc> <Esc>
Customizing the Kernel

Figure 5-69. Networking Utilities (udhcpd)

120. Select Miscellaneous Configuration See Figure 5-70.
Figure 5-70. Miscellaneous Configuration

- Select **Generic CGI** See Figure 5-71.
- Press `<Esc>` `<Esc>`
Customizing the Kernel

Figure 5-71. Miscellaneous Configuration Generic CGI

121. Press <Esc> <Esc>.

122. Press Y to save the configuration settings.

123. You will return to Linux terminal.

Compiling the kernel

To compile the kernel, follow the steps below:

1. Type the following command to compile the kernel:
   
   ```
   #make
   ```

   Figure 5-72. show the system compilation.
After compilation, you will get different images in the image folder located at:
/home/sls/Nios2-Linux/Linux_source/uClinux-dist/images/
The `linux.initramfs.gz` file is an elf image with initramfs.

Running the BSP

To run BSP on Nios II reference design, follow the steps below:

1. Download the `sys_qii100sp1_linux_bsp_s4gxdb.sof` file generated in the previous chapter or from the reference design located at See Figure 5-73.

   /home/sls/Nios2-linux/System-Board/4s230_default.

2. Download the elf file `linux.initramfs.gz` located at

   /home/sls/Nios2-linux/Linux_source/uClinux-dist/images/
Customizing the Kernel

Figure 5-73. Downloading ELF Image

```bash
[root@centos36 images]# ls
linux.initramfs.gz roots.initramfs.contents vmImage
linux.initramfs.gz.src roots.initramfs.gz vmlinu
nios2-download.pid roots.jffs2 zImage
[root@centos36 images]# nios2-download -g linux.initramfs.gz
Using cable "USB-Blaster [USB 4-1.1]", device 1, instance 0x00
Pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 628KB in 54.7s (114.9KB/s)
Verified OK
Starting processor at address 0xC0000000
[root@centos36 images]#
```

3. You will get Linux booting messages on the nios2-terminal window. See Figure 5-74.

```
#nios2-download -g linux.initramfs.gz
#nios2-terminal
```

Figure 5-74. Running Linux On the Board

```
BusBox v1.16.2 (2019-08-30 19:19:23 UTC) bash -- The humble shell
Enter ‘help’ for a list of built-in commands. /#
```
Login:
Username: root
Password: nios2linux

To mount the JFFS2 file system on memory, follow the steps below:

4. Type following command to unlock the memory block for erase.
   \#flash_unlock /dev/mtd7

5. Type following command to erase the memory block.
   \#flash_eraseall -j /dev/mtd7

6. Type following command to mount JFFS2 file system on /mnt directory.
   \#mount -t jffs2 /dev/mtdblock7 /mnt

7. Type following command to go to “mnt” directory.
   \#cd /mnt

8. Type the command to check mounted file system.
   \#df

   This message displays mounted file system on memory block 7. See Figure 5-75.

---

**Figure 5-75. Mounting JFFS2 File System**

![Mounting JFFS2 File System](image)

---

Applications On Running BSP

For these applications except USB Host, Altera NEEK board's HSMC port should be connected on Stratix IV board's HSMC Port A. For USB/Host application, Altera terasic THDB-SUM board's HSMC port should be connected on Stratix IV board's HSMC Port B.
Customizing the Kernel

Mounting VFAT on SD-Card

1. Before Power-On board insert SD-card in NEEK board's SD-Card slot. See Figure 5-76.

Figure 5-76. Mounting VFAT on SD-Card

2. SD-card is detected as mmcblk0 as shown in boot message. See Figure 5-77.
3. Mount SD-Card. See Figure 5-78. on /mnt/sdcard directory.
   # mount -t vfat /dev/mmcblk0 /mnt/sdcard

4. Check mounted file system using “df” or “mount” command.
Mounting a JFFS2 File System

1. For JFFS2 file system, Kernel must be configured for MTD and JFFS2 file system.

2. To check MTD partitions, use
   
   ```
   # cat /proc/mtd
   ```

3. To mount `/dev/mtd1` partition as JFFS2 file system on `/mnt/jffs` folder,
   
   ```
   #flash_unlock /dev/mtd1
   #flash_eraseall -j /dev/mtd1
   #mount -t jffs2 /dev/mtdblock1 /mnt/jffs
   ```

4. Check mounted file system using “df” or “mount” command.

5. Create any file or directory on mounted file system. See Figure 5-78.
Input Devices Applications

1. Check the boot message which displays configured input devices. See Figure 5-79.
Customizing the Kernel

**Figure 5-80. Input Devices Applications**

1. Touch panel is configured as input1 and event1. See Figure 5-81.
2. Run the input_driver_test application as shown,

   ```
   #input_driver_test /dev/input/event1
   ```
3. On success, touch the NEEK boards touchscreen, it will display co-ordinates values. See Figure 5-82.
4. Even the resulting messages can also be viewed using "`gmesg`" command. See Figure 5-83.
PS2 Keyboard

1. Connect PS2 Keyboard on PS2 port of NEEK board.
2. PS2 Keyboard is configured as input0 and event0.
3. Run the `input_driver_test` application as shown,
   
   ```
   #input_driver_test /dev/input/event0
   ```
4. On success, press any key of keyboard, it will display code values. See Figure 5-84.
5. Even the resulting messages can also be viewed using “gmesg” command. See Figure 5-85.

Figure 5-84. PS2 Keyboard

```
  
  #
  #
  #
  #
  # Input driver test /dev/input/event0
  IN50: succeeded to open /dev/input/event0
  1167610142.168595 type 1 code 47 value 1
  type 0 code 0 value 0
  1167610142.262121 type 1 code 47 value 0
  type 0 code 0 value 0
  1167610142.310938 type 1 code 47 value 1
  type 0 code 0 value 0
  1167610142.727312 type 1 code 47 value 0
  type 0 code 0 value 0
  1167610144.357109 type 1 code 37 value 1
  type 0 code 0 value 0
  1167610144.410376 type 1 code 37 value 0
  type 0 code 0 value 0
  1167610144.589114 type 1 code 40 value 1
  type 0 code 0 value 0
  1167610144.914501 type 1 code 40 value 0
  type 0 code 0 value 0
  1167610146.052520 type 1 code 37 value 1
  type 0 code 0 value 0
  1167610146.110194 type 1 code 39 value 1
  type 0 code 0 value 0
  1167610146.155704 type 1 code 37 value 0
  type 0 code 0 value 0
  1167610146.206448 type 1 code 39 value 0
  type 0 code 0 value 0
```
Button PIO

1. Open /dev/btn as background,
   
   `#cat /dev/btn &`

2. Pressing of any push button 1, 2, or 3 will display button number. See Figure 5-86.
3. To Kill these process, give kill command with pid of /dev/btn and press any push. See Figure 5-87.
I2C Applications

1. Check the boot message which displays configured i2c devices. See Figure 5-88.

   i2c-0 for eeprom and i2c-1 for audio-codec

---

Figure 5-88. I2C Applications

---

I2C Detect

1. i2c detect will display the address where i2c devices are connected. See Figure 5-89.

   #i2cdetect 0 or #i2cdetect 1

2. I2C EEPROM on NEEK board has address range between 0x50 to 0x57

3. Device address for I2C interface for audio codec is 0x1A.
Customizing the Kernel

Figure 5-89. I2C Detect

I2C EEPROM Read and Write

1. To read eeprom's byte value from address 0x01 with eeprom address value 0x51. See Figure 5-90.

   #i2cget 0 0x51 0x01 b

2. Address 0x01 has value 0x23

3. To write eeprom 1 byte 0x45 value at address 0x01

   #i2cset 0 0x51 0x01 0x45 b

4. Verify the value at address 0x01 using i2cget.
**I2C Audio Controller**

1. To check i2c audio codec, run application i2c_audio_bypass. See Figure 5-91.

2. Connect **LINE-IN** of NEEK board with Host system's **LINE-OUT** using aux cable.

3. Connect **LINE-OUT** of NEEK board with Speaker.
4. Run audio on player of your Host system with application
   
   `#i2c_audio_bypass`

---

**TFTP Applications**

1. Trivial File transfer protocol (tftp) is used for file transfer from Host PC to Stratix IV GX Development kit. See Figure 5-92.

2. To get the remote file from tftp server running on Windows or Linux Host.

   `tftp -g -r [File] [HOST]`

   `#tftp -g -r Sunset.jpg 192.168.0.26`
TFTP Applications

**Figure 5-92. Trivial File transfer protocol(tftp - 1)**

```
/ # ls
bin etc init mnt root sys usr
dev home lib proc sbin tmp var
/ #
/ # tftp -g -r Sunset.jpg 192.168.0.26
/ # ls
Sunset.jpg etc lib root tmp
bin home mnt sbin usr
dev init proc sys var
/ #
```

**TFTP Server**

1. To make Stratix IV GX Development Kit as TFTP Server. See Figure 5-93.

2. After Ethernet configuration, run this command,
   
   ```
   #udpsvd -vE 0.0.0.0 69 tftpd /home/tftpboot
   ```

3. Access files from Host system from Stratix IV GX Development Kit's tftpboot folder.
Customizing the Kernel

**TELNET Application**

1. It is simple utility to access Target board via Ethernet.
2. To access target board via telnet, give telnet command from Windows or Linux Host
   
   ```
   # telnet 192.168.0.181
   ```

**BOA Application**

1. Open any Internet browser on Host and type http://192.168.0.181.
   
   ```
   # boa -c /etc &
   http://192.168.0.181
   ```

**FTP Application**

2. Connect target board using FTP application On Host system, run this command.
   
   ```
   ftp 192.168.0.181
   ```

**Dropbear Application**

3. Connect the target board using SSH, On host system, run this command
   
   ```
   ssh root@192.168.0.181
   ```
4. This application will work if you have selected Test Applications (SLS) while configuring applications.

5. Run this command on terminal, you can see output on LCD
   
   # jpegview