Nios II Embedded Evaluation Kit, Cyclone III Edition

User Guide

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

P25-36209-01

Document Date: November 2007

Part Number UG-01025-1.0

Contents

Appendix A. Video Pipeline Data Flow

Appendix B. Application Selector Details

Appendix C. Re-building the Factory Image

Appendix D. Frequently Asked Questions

Additional Information

1. Getting Started

Introduction

The Altera® Nios II Embedded Evaluation Kit, Cyclone III Edition includes a full-featured field-programmable gate array (FPGA) development board, LCD Multimedia Daughtercard, hardware and software development tools, documentation, and accessories needed to begin embedded and system on a programmable chip (SOPC) designs using FPGAs.

The development board includes an Altera Cyclone III FPGA and comes preconfigured with an FPGA hardware reference design stored in flash memory as well as several sample applications stored on the SD-Card Flash provided. Hardware designers can use the development board as a platform to prototype complex embedded systems. Software developers can use the hardware reference design plus sample software applications as a starting point for their own applications.

The list of applications is shown in table Table 1–1 below.

Table 1–1. List of Example Applications

Demo	Vendor
Application Selector	Altera
Picture Viewer	Altera
Mandelbrot C2H	Altera
Web Server	Altera
Spinning Cube	Altera
Tacquin Game	Imagem
Watch	Imagem
Avionics	Imagem
uC-GUI	Micrium
Photo Frame	PlanetWeb
SpectraWorks	PlanetWeb
DAVE 2D Graphics	TES

Nios II is an embedded processor system that can be configured by the user, meaning the actual hardware of the processor is easily customized for a particular application through the SOPC Builder feature of the Quartus II and SOPC Builder FPGA design software. This microprocessor design process involves four elements:

1. Specifying functionality of the core processor and memory
2. Adding peripherals such as memory interfaces, I/O, or interfaces to external devices (such as the LCD Display)
3. Connecting the I/O pins of the processor in the FPGA to the external devices
4. Writing C/C++ software application for your custom processor within the NIOS II IDE

In the vast majority of cases, all this hardware design can be accomplished using drop-down menus and drag and drop operations in SOPC Builder.

The sophistication of the Altera design tools brings many hardware processor designs within the reach of the embedded software engineer. The detailed construction of the processor is managed by Quartus II and SOPC Builder. All the embedded designer has to do is to specify functionality and basic connectivity of the processor. They can then concentrate on the software with the knowledge that the microprocessor encapsulates all of the best trade-offs for their particular design.

For simplicity, the Nios II Embedded Evaluation Kit, Cyclone III Edition provides a standard hardware example named the Nios II Standard System that serves as a starting point for embedded software development. The Nios II Standard System is a pre-generated hardware system that includes

- Nios II core (32-bit soft processor)
- LCD Controller
- Multi-port memory controllers
- Communication Interface controllers

Using the pre-generated Nios II Standard System enables embedded developers to focus on application development without having to concern themselves with the details of generating the FPGA hardware system.

The Application Selector, Picture Viewer, Mandelbrot, and Web Server examples are provided in source-code on the development kit CD-ROM and described in this document.

About this User Guide

This user guide describes how to start using the Altera® Nios ® II Embedded Evaluation Kit, including unpacking the kit, installing required software, and running the Application Selector utility and other design examples. This user guide addresses the following topics:

- How to set up, power up, and verify correct operation of the Nios II Embedded Evaluation board
- Nios II standard hardware for the Embedded Evaluation board
- Where to get and how to install the Altera Development Suite Tools
- How to install the *Nios II Embedded Evaluation Kit, Cyclone III Edition* CD-ROM
- How to start and run the Application selector utility
- Design examples
- Taking the next step
- Frequently asked questions

For a full description of the development boards and their design and use, refer to the *Cyclone III FPGA Starter Board Reference manual* and *LCD Multimedia Daughtercard Reference Manual*.

This document was written to help you quickly get started and provide an overview of some of the applications. In the appendices more detail about key hardware components and the structure of the application selector utility have been presented. However, we opted to provide extensive source-code comments rather than formal documentation regarding the other applications.

We are actively interested in knowing if this structure for the documentation is adequate for you to be able to develop your applications. If you have comments or suggestions on what we can do to improve the user experience through our documentation, please contact us through nios_docs@altera.com.

To ensure that you have the most up-to-date information on this product, go to the Altera website @ **www.altera.com/products/devkits/altera/kit-cyc3-embedded.html**

Before You Begin

Before proceeding, check the contents of the kit:

- Nios II Embedded Evaluation Board
- *Nios II Embedded Evaluation Kit* CD-ROM

- Altera Complete Design Suite DVD
- SD-Card Reader (USB 2.0)
- SD-Card Flash
- USB Cable
- 12 V DC power supply with the appropriate cable for your region

Software Installation

The Altera Complete Design Suite DVD contains the Quartus II Tools and Nios II Embedded Design Suite tools and OpenCore Plus versions of Altera IP (including the Nios II processor core). The Nios II Embedded Evaluation Kit, Cyclone III Edition CD ROM contains the documentation and examples that were created (and you can modify) with these tools and IP.

To use the Nios II Embeded Evaluation Kit you will first need to install the following software:

- *Nios II Embedded Evaluation Kit* CD-ROM
- Nios II Embedded Development Suite, Altera Complete Design Suite

Installing the Nios II Embedded Evaluation Kit, Cyclone III Edition CD-ROM

The *Nios II Embedded Evaluation Kit* CD-ROM contains the following items:

- Documentation, including
  - *Quick Start Guide*
  - *Hardware and Software Tutorial*
  - *Nios II Embedded Evaluation Kit, Cyclone III Edition User Guide* (this document)
  - *LCD Multimedia Daughtercard Reference Manual*
  - *Cyclone III FPGA Starter Board Reference Manual*
- Example Applications
- Board Design Files

To install the *Nios II Embedded Evaluation Kit* CD-ROM, perform the following steps:

1. Insert the *Nios II Embedded Evaluation Kit* CD-ROM into the CD-ROM drive.

   The CD-ROM should start an auto-install process. If it does not, browse to the CD-ROM drive and double-click on the **setup.exe** file.

2. Follow the online instructions to complete the installation process. The installation program copies the Nios II Embedded Evaluation Kit, Cyclone III Edition files to the hard-disk and creates an icon labeled: **Programs > Altera > Nios II Embedded Evaluation Kit, Cyclone III Edition <version #>**, which is accessible from the Windows Start menu. Use this icon to launch the Windows-style development kit GUI.

The Nios II Embedded Evaluation Kit, Cyclone III Edition installation program creates a directory structure for the installed files (Figure 1–1).

Figure 1–1. Nios II I Embedded Evaluation Kit Installed Directory Structure

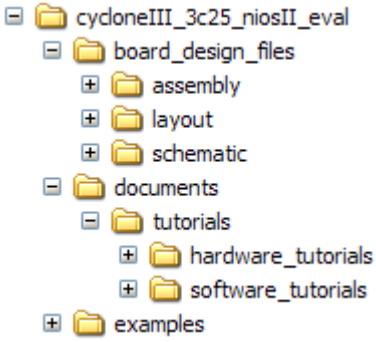


Table 1–2 lists the file directory names and a description of their contents.

Table 1–2. Installed Directory Contents

Directory Name	Description of Contents
board_design_files	Contains schematic, layout, assembly, and bill of material board design files.

Table 1–2. Installed Directory Contents

Directory Name	Description of Contents
documents	Contains the Nios II Embedded Evaluation Kit, Cyclone III Edition documentation, including hardware and software tutorials.
examples	Contains design examples for the Nios II Embedded Evaluation Kit, Cyclone III Edition, Application Selector Utility, and Nios II Embedded Evaluation Standard hardware system.
factory_recovery	Contains image files for the SD-Card and flash that can be used to reprogram the board to the factory default state.

Installing the Altera Complete Design Suite Software

Load the Altera Complete Design Suite DVD into the DVD player, and click on **Install free package** on the startup screen (Figure 1–2). Follow the on-screen instructions and accept all defaults. After installing the software, request and install a license to enable it.

During installation of the Quartus II software, choose to install the MegaCore IP Library when presented the option and follow the on-screen instructions.

For information on obtaining a license file, refer to "“Licensing the Quartus II Software” on page 1–6.

If you have difficulty installing the Quartus II software, refer to *Installing the Quartus II Software* in the *Quartus II Installation & Licensing Manual for PCs* found at **www.altera.com**.

The Quartus II software is the primary FPGA development tool used to create the reference designs used in this development kit.

The Nios II Embedded Design Suite is the set of tools used to compile, debug, and run the software that runs on the Nios processor in the provided example designs.

Licensing the Quartus II Software

Before using the Quartus II software, you must request a license file from the Altera website at www.altera.com/licensing and install it on your PC.When you request a license file, Altera e-mails you a license.dat file that enables the software. To obtain a license, perform the following steps:

1. Go to the Altera website at www.altera.com/licensing.
2. Click Quartus II Web Edition Software.

Figure 1–2. Installing the Altera Complete Design Suite Software

3. Follow the on-line instructions to request your license. A license file is e-mailed to you. Save this file on your computer.
4. Run the Quartus II software.
5. Choose **License Setup** (Tools menu).
6. Under **License File**, indicate the location and license file name you received and saved onto your computer in step 3.

2. Development Board Setup

Features

The Nios II Embedded Evaluation Kit features:

- Cyclone III Starter Board
  - Cyclone III EP3C25F324 FPGA
  - Configuration
    - Embedded USB-Blaster™ circuitry (includes an Altera EPM3128A CPLD) allowing download of FPGA configuration files via the users USB port
  - Power and analog devices from Linear Technology
  - Memory
    - 256-Mbit DDR SDRAM
    - 1-Mbyte Synchronous SRAM
    - 16-Mbytes Intel P30/P33 flash
  - Clocking
    - 50-MHz on-board oscillator
  - Switches and indicators
    - Six push buttons total, 4 user controlled
    - Four user-controlled LEDs
- LCD Daughtercard
  - LCD Touch-screen Display
    - 800 X 480 pixel size
  - 10-bit VGA DAC
  - Video Decoder
  - 24-bit Audio Codec
  - RS232 transceiver
  - SD Flash
  - 10/100 Mbps Ethernet Controller (PHY)
  - Connectors
    - VGA Output
    - Composite Video in
    - Serial connector (RS-232 DB9 port)
    - PS/2
    - Ethernet Connector (RJ 45)
    - SD Card Socket

Requirements

If not already installed, you should:

- Install the Altera® Complete Design Suite software on the host computer
- Install the *Nios II Embedded Evaluation Kit* CD ROM

- Install the USB-Blaster™ driver software on the host computer. The Cyclone III FPGA starter development board includes integrated USB-Blaster circuitry for FPGA programming.

The USB Blaster driver software is provided with the Quartus II software installation. Communication between the host computer and the development board requires that the USB-Blaster driver software be set up.

Power Up the Development Board

To power up the development board, perform the following steps:

1. Ensure that the red on/off switch (SW1) - on the back-side of the development board is in the **OFF** position (up).
2. Connect the USB-Blaster cable from the host computer to the USB-Blaster port on the development board.
3. Connect the 12 V DC adapter to the development board and to a power source.

CAUTION

Only use the supplied 12 V power supply. Power regulation circuitry on the board could be damaged by supplies greater than 12 V.

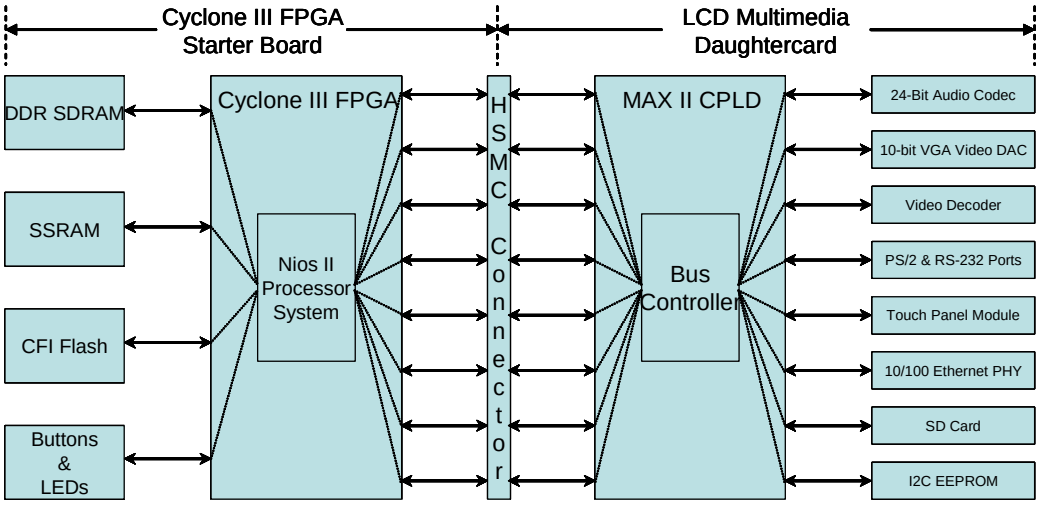
4. Press the **Power Switch** (SW1).
5. A Welcome screen displays as shown in Figure 2–1.

Figure 2–1. Development Board Setup - Welcome Screen

3. Nios II Standard Hardware for the Embedded Evaluation Board

The board in the Nios II Embedded Evaluation kit is comprised of the components shown in the Figure 3–1 below.

Figure 3–1. Block Diagram of Nios II Embedded Evaluation Kit, Cyclone III Edition



If you examine your Nios II Embedded Evaluation Kit, Cyclone III Edition, you will find that it is comprised of 2 boards, the Cyclone III FPGA Starter Board and the LCD Multimedia Daughtercard. On the Cyclone III FPGA Starter board resides the Cyclone III 3c25 FPGA which configures from flash with the Nios II Standard Hardware system on startup.

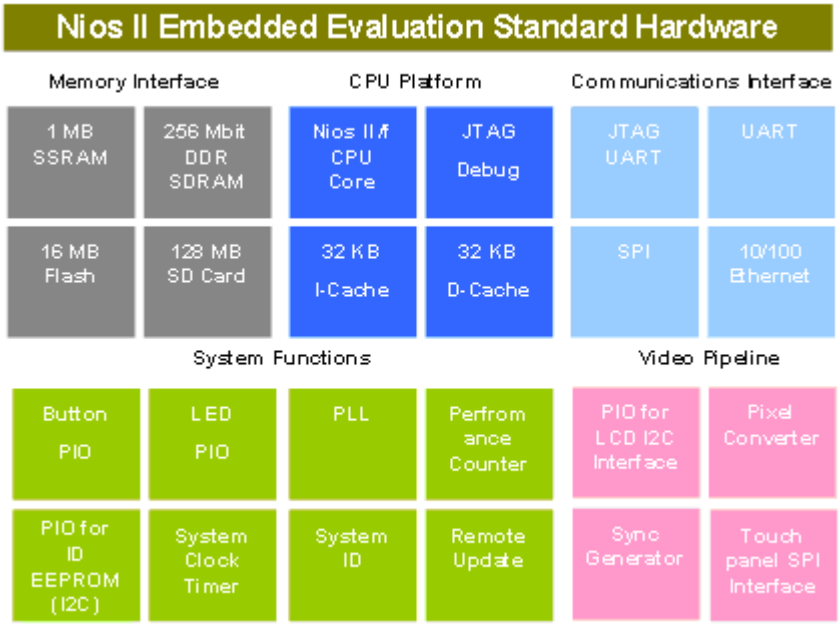
The HSMC Connector shown in Figure 3–1 is actually a flex extension cable with HSMC connectors on each end going between the two boards. This detail was removed for simplicity.

On the LCD Multimedia Daughtercard resides a MAX II CPLD whose function is to relay data and control signals to the various peripheral devices as shown in the Figure 3–1. The MAX II CPLD performs voltage translation and de-multiplexing of video pipeline signals to the LCD

Touch panel. The video pipeline signals have been multiplexed inside the FPGA and de-multiplexed by the MAX II CPLD to provide a full range of functionality on the daughter card over a limited number of pins on the HSMC connector. (see the *LCD Multimedia Daughtercard Reference Manual* for details)

Within the FPGA is the Nios II Standard System. It is a pre-generated Nios II processor based hardware system that can be used as a starting point for embedded application development. The components in this embedded microprocessor system are shown in Figure 3–2.

Figure 3–2. Nios II Embedded Evaluation Standard Hardware System Block Diagram



CPU Platform

The CPU platform for the Nios II Standard System consists of

- Nios II/f cpu core
- JTAG Debug Port

- 32KB Instruction Cache
- 32KB Data Cache

System Functions

PLL

The PLL accepts the global input clock source from the 50MHz on-board oscillator and generates the following clocks

- 100 MHz CPU Clock
- 100 MHz SSRAM Clock
- 66.5 MHz DDR SDRAM Clock
- 60 MHz Peripheral Clock (“slow peripherals”)
- 40 MHz Remote System Update Clock

System Clock Timer

General purpose system timer.

Performance Counter

Counter used for debug and system performance analysis.

System ID

Used to sync the hardware system generation with the software generation tools.

Remote System Update Block

Used for automatic configuration at boot-time from the on-board active parallel flash. The Nios II processor writes reset address of the hardware system stored in flash for reconfiguration

LED PIO

Output only control block for LED1-LED4

Pushbutton PIO

Input only control block for the on-board pushbuttons.

PIO for ID EEPROM (I2C)

Used to communicate with the EEPROM ID chip which stores information about the board including the touch panel calibration data and Ethernet MAC address

The I2C interface is implemented using software and general purpose I/Os connected to the Nios II Standard System.

Memory Interface

There are four different types of on-board memory or storage devices. The memory controllers for three of these devices are provided as part of Altera's IP Suite. These include

- SSRAM Controller
- DDR SDRAM Controller
- CFI Flash Controller
- SD Card

The controller, API, and FAT File System for the SD-Card used in the Nios II Standard System is provided under license agreement by El Camino (http://www.elcamino.de)

For technical details on the components in standard hardware system refer to *Quartus II Handbook Volume 5: Embedded Peripherals.*

Communication Interfaces

There are several communication interfaces included in the Nios II Standard System:

JTAG UART

Used for Serial communication and debugging Nios II applications via the on-board USB-Blaster circuitry.

UART

Serial communication link for general purpose communications and debug.

SPI

Used to communicate with the touch panel portion of the LCD Touchscreen.

10/100 Ethernet Controller

The Ethernet controller uses the Triple-speed Ethernet MAC to communicate with the PHY on the LCD Multimedia Daughtercard.

Video Pipeline

The video pipeline outputs the appropriate pixel data and sync signals to the LCD Touch Panel. It provides high bandwidth memory access that allows for flicker free display on the color LCD.

A more detailed description of the data flow for the video pipeline can be found in Appendix A.

The video pipeline is comprised of:

PIO for LCD I2C controller

The I2C pins are used to configure the LCD panel for brightness and set the gamma correction curves.

SPI touch panel controller

Used to communicate with the touch panel ADCs.

Pixel Converter

Logic block that converts parallel 32-bit R-G-B-0 data to an 8-bit data stream. This is required because of the pin-limitation placed on the system by the HSMC connector. The video data-stream is multiplexed in the FPGA on the Cyclone III Starter Board and de-multiplexed in the MAX II device on the LCD Multimedia Daughtercard.

Sync Generator

Generates the horizontal and vertical sync signals for each frame displayed on the LCD touch screen.

For more information on the video pipeline (pixel converter and Video Sync Generator) and GPIO components refer to *Quartus II Handbook Chapter 5 Embedded Peripherals*.

4. Application Selector Utility

Overview

The Nios II Embedded Evaluation Kit, Cyclone III Edition application selector is the default utility which allows users to quickly select, load, and run different applications using the LCD touch panel. Applications are stored using a FAT file system on an SD Card and retrieved at load-time. An application consists of a hardware and a software image, both of which are loaded and run by the application selector.

The FAT File System used on the SD-Card is licensed by El Camino. www.elcamino.de

In addition to the pre-packaged applications which come with the Nios II Embedded Evaluation Kit, Cyclone III Edition, more applications are available from Altera or through third party vendors. Also, you can easily convert your own applications to be loadable by the application selector.

Running the Application Selector

This section describes the general operation of the Application Selector utility.

Starting the Utility

To start the application selector, connect power to the Nios II Embedded Evaluation Kit board, Cyclone III Edition, and switch on the power (SW1). If the board is already powered, reset the board by pressing the button labeled **RECONFIGURE**.

You may have to press the **RECONFIGURE** button twice to properly reset the board.

The application selector will boot from flash, and a splash screen will appear while the application selector searches for applications on the SD Card. Then the main menu will appear and a list of loadable applications will be displayed (see Figures 4–1).

If the application selector does not start when power is applied or when the board is reset, see the FAQ "How do I restore the factory image?".

When the main menu appears, you will see a scrollable list of applications. These are the applications which were found on the SD Card and are now available to load. You can highlight any of the applications

by touching them. If there are more than five applications on the SD Card, you can scroll through the list by touching the scroll-up and scroll-down buttons on the right hand side of the screen.

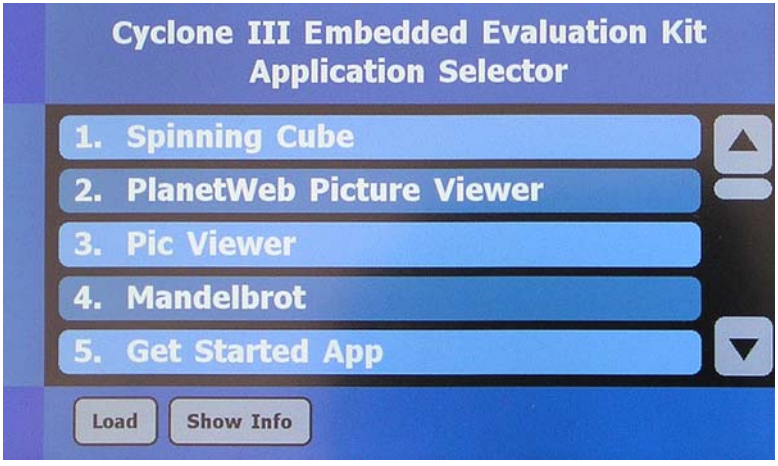
Viewing Information about an application

To get more information about a particular application, highlight the application by touching it then touch the button labeled **Show Info**. If there is additional information available for the application you highlighted, a scrollable text window will appear. To return to the main menu, touch the button labeled **OK**.

Loading and Running an application

When you've selected the application you want to load, touch the button labeled **Load**. The application will begin loading, and a small window will be displayed showing the progress. Loading will take between 2 and 30 seconds, depending on the size of the application, and whether it was previously cached in on-board flash memory.

Figure 4–1. View of the Application Selector User Interface



For more detailed information about the Application Selector Utility, see Appendix B: Application Selector Details.

5. Design Examples

About the Design Examples

The Nios II Embedded Evaluation kit comes with several applications that showcase the versatility of the Nios II processor in various applications such as imaging, graphics, networking etc.

To aid in the learning process of software developers, three design examples have been provided in source code form in the examples directory in the Nios II Embedded Evaluation Kit CD. These design examples are:

- Picture Viewer
- Mandelbrot
- Web Server

For each of these applications a basic overview and discussion of operation is given. However much more detailed information can be found in the source-code which is loaded when the Nios II Embedded Evaluation Kit CD ROM is installed.

The Nios II Embedded Evaluation kit also contains more applications provided from third party vendors to showcase available graphics libraries and middleware that have been ported to the Nios II processor, but these design are shipped in binary form not as source code.

Picture Viewer Application

Picture Viewer

The picture viewer application is built for the standard hardware system to show-case video display via the LCD touch panel.

The picture viewer application takes JPEG images or bitmaps stored on the SD Card and displays them on the LCD Touch Panel. The Nios II CPU decodes the images and stores the pixels in a video buffer in DDR. A Scatter-Gather DMA is used to transfer pixel data from the video buffer to the video pipeline.

You can customize the picture viewer application's image selection by adding your own images in to the folder on the SD-Card entitled **images**.

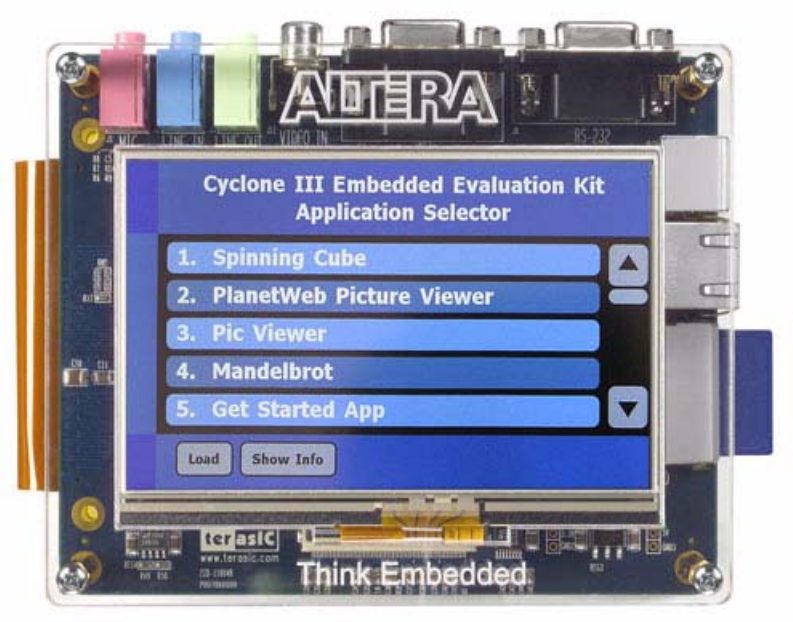
Operation

The Picture Viewer application displays a new picture on the LCD screen after a settable delay (1,2,3,4,5,10,15,20 seconds). If decoding the image takes longer than this delay time, then the image is displayed as soon as it has been decoded. The image is scaled to optimally fit the LCD screen.

The operation of Picture Viewer application is explained below:

1. Power on the board by pressing the switch **SW1**. You will see the **Application Selector** menu on the LCD Touch Screen Display. See Figure 5–1.

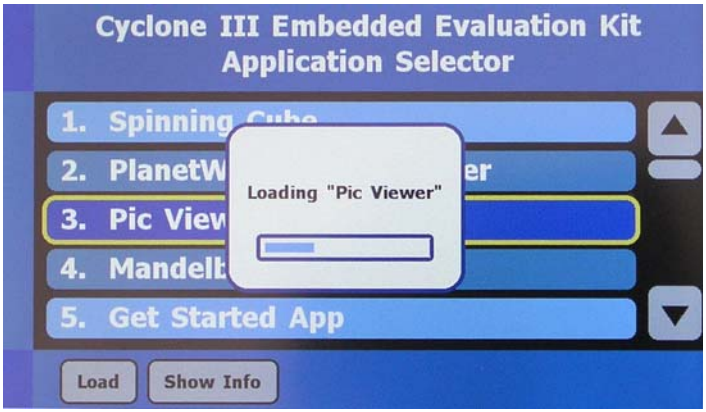
Figure 5–1. Application Selector Menu



2. Select the **Pic Viewer** option by touching it in the application selector menu.

3. Touch the **Load** button located on the bottom left corner of the Touch Screen to load the Pic Viewer application. You will see the progress bar on the screen. See Figure 5–2.

Figure 5–2. Loading the Picture Viewer Application

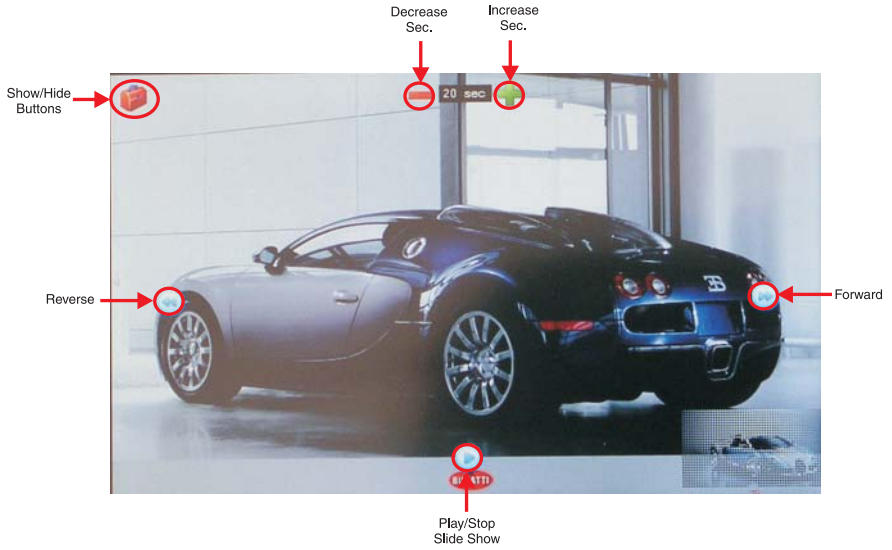


4. After loading the Pic Viewer application you will see the a slide show of pictures stored on the SD card. Figures 5–4. shows the first image stored on the SD card. The miniature view on the bottom right corner shows the next image of the slide show.

Figure 5–3. Running the Picture Viewer Application - Displaying First Image

5. The next image will be displayed after the delay period. See Figure 5–4.

Figure 5–4. Running the Picture Viewer Application



You can control the slide show as explained below:

- To display the next image before the delay time is finished, touch the **Forward** button located at the right center of the touch panel.
- To display the previous image, touch the **Reverse** button located at the left center of the touch panel.
- To play or stop the slide, touch the **Play** **/Stop** button.
- On the top center of touch panel you will see the **Delay-period** (in **Sec.**). You can increase or decrease the delay period by touching the **Plus** or **Minus** buttons respectively. The maximum delay period you can set is **20** seconds. The minimum is 1 second. The default delay period is **5** seconds.

- You can hide the control buttons by clicking on the **Hide** button located at the top left corner of the touch screen. To show the control buttons again touch anywhere on the LCD Touch panel.
- On the bottom right corner, you will see the miniature view of the next picture being decoded in the background.

6. The slide show continues until you tap the **Stop** button.
7. To return to the Application Selector menu press the **Reconfigure** push-button switch on Cyclone III Starter Board.

Mandelbrot Application

The Mandelbrot Set is a mathematical set of complex numbers that form a fractal. The Mandelbrot set is generated from a surprisingly simple formula involving only multiplication and addition to produce a shape of great organic beauty and infinite subtle variation. Though the Mandelbrot set is intriguing in itself, the Mandelbrot C2H Demo on the Nios II Embedded Evaluation kit showcases a powerful solution to a common engineering problem: increasing the performance of a system bound by processing through put.

This example design shows a greater than 100x improvement in performance between software only and software with hardware accelerators! The C to Hardware tool was used to take working software code and automatically generate the hardware accelerators that provide this performance improvement.

There are two processes at play here:

1. The calculation of the Mandelbrot Set to generate pixel data
2. The rendering of the pixel data on the LCD screen

Traditional processors will perform these functions purely in software. Options available to increase throughput once the processor and clock frequency are selected are extremely limited. The unfortunate trade-off of porting the entire application to a faster more expensive processor with a higher clock frequency (and hence power consumption) is unacceptable for cost and power sensitive applications.

The Nios II Embedded Evaluation kit, features not a traditional processor but a Nios II-based FPGA and using automated hardware acceleration. The Nios II C to Hardware (C2H) Acceleration Compiler, takes standard ANSI C code, in this case the Mandelbrot algorithm and automatically generates hardware accelerators.

In the hardware accelerated version of the design the Nios II processor handles common video functions such as the rendering the image, panning, zooming etc. The hardware accelerator concentrates on generating the pixels by computing the Mandelbrot function all in time for the next frame.

You can use the design to observe the differences between a general purpose processor executing software and a group of hardware accelerators performing the same functionality. When comparing the software-only version with the software plus hardware accelerators you should expect to see a 250 times speed improvement between the software and hardware in the image rendering.

Using the Mandelbrot application

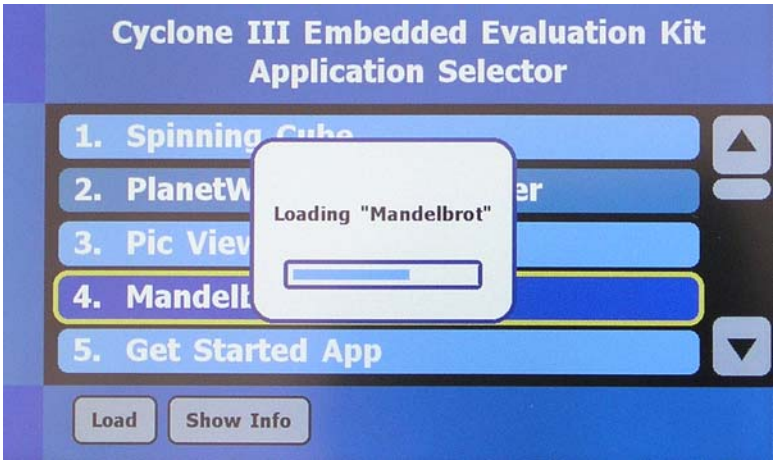
The Mandelbrot application utilizes the LCD and touchscreen for all user interactions. When the application starts you will be prompted with a blue welcome screen that you must touch to continue.The operation of Mandelbrot application is explained below:

1. Power on the board by pressing the switch **SW1**. You will see the **Application Selector** menu on the LCD Touch Screen Display.

2. Select **Manelbrot Application** by touching it in the application selector menu.

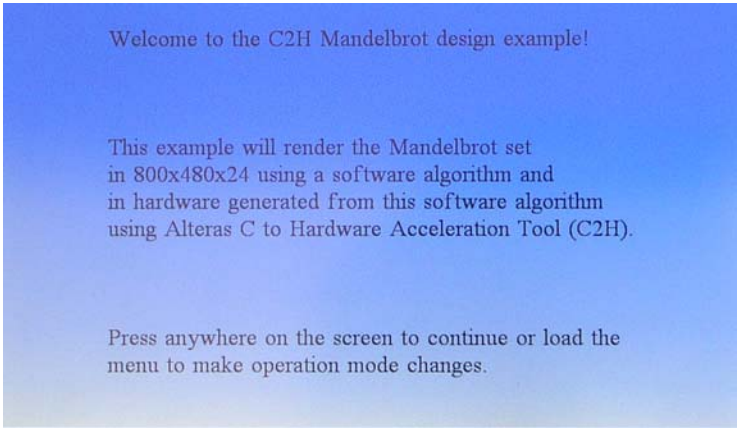
3. Touch the button marked **Load**. TheLCD Touch panel display begins loading the Manelbrot application as shown in Figure 5–5.

Figure 5–5. Loading the Mandelbrot Application



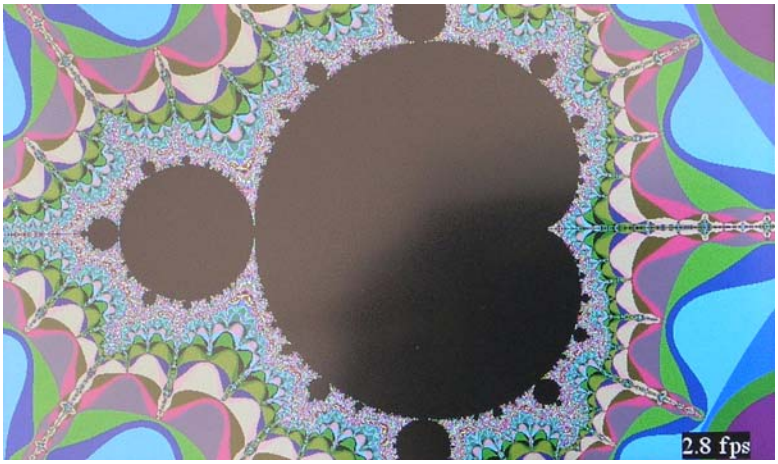
4. After complete loading of application, you will see a **welcome screen** as shown in Figure 5–6.

Figure 5–6. Welcome Screen of Mandelbrot Application



5. When you tap the touchscreen, the hardware accelrated version of the Mandelbrot application will begin running, changing co-ordinates and zooming in and out of the complex space with default mode settings. See Figure 5–7

Figure 5–7. Running the Mandelbrot Application



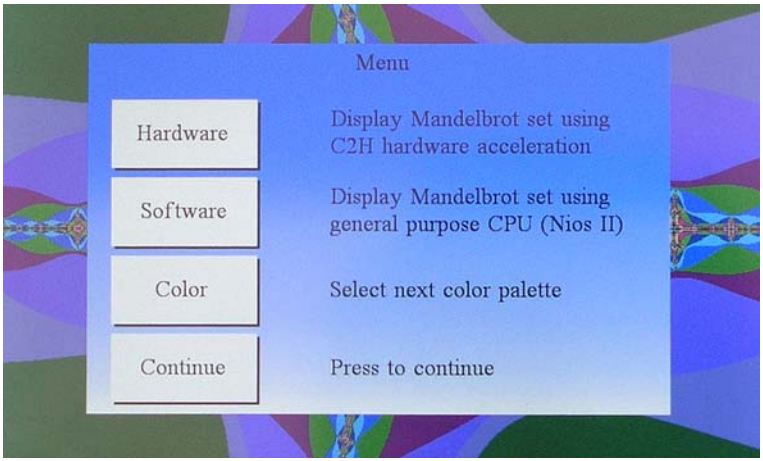
☞ The default mode used in the design uses hardware acceleration.

6. To change modes, color palettes, or pause the design simply tap the touch panel to bring up the menu.

7. The menu will offer you the choice of using hardware or software rendering. To select software rendering press the **Software** button followed by the **Continue** button. See Figure 5–8.

☞ It is important to note that software rendering can be very slow so you may have to wait a long time for a single frame to be displayed.

Figure 5–8. Mandelbrot Application Menu



8. To change the color palette used in the final image simply press the **Color** button followed by the **Continue** button. See Figure 5–8.

While the menu is being displayed, all rendering will be paused as well. If you opened the menu and wish to continue without changing any settings press the **Continue** button.

Whether the design is rendering data using hardware or software, benchmark data is being collected and displayed to the screen.

- When hardware rendering is selected the benchmark data is updated every 5 frames.
- When software rendering is selected the benchmark data is updated every frame.

The benchmark data is displayed in the bottom right of the screen and it represents the instantaneous frames per second being rendered and displayed.

Consider the implications of what you have observed: What one would traditionally do with an expensive, power hungry GHz processor was just accomplished on an inexpensive Cyclone III FPGA, running at 100 MHz. Such is the power of hardware acceleration using FPGAs.

Operation

The design performs panning and zooming on the complex plane which gives a video like effect. Every time a new frame is rendered, a new set of coordinates must be calculated. These coordinates contain a center point, zoom factor, and maximum number of iterations. Knowing the center point and zoom factor the top left point of the screen is then determine and passed to the Mandelbrot algorithm. The maximum number of iterations is used to determine how much effort is spent per pixel before it is determined that the point is included in the Mandelbrot set (these points appear as black pixels). The pixel calculation is based on the following software segment:

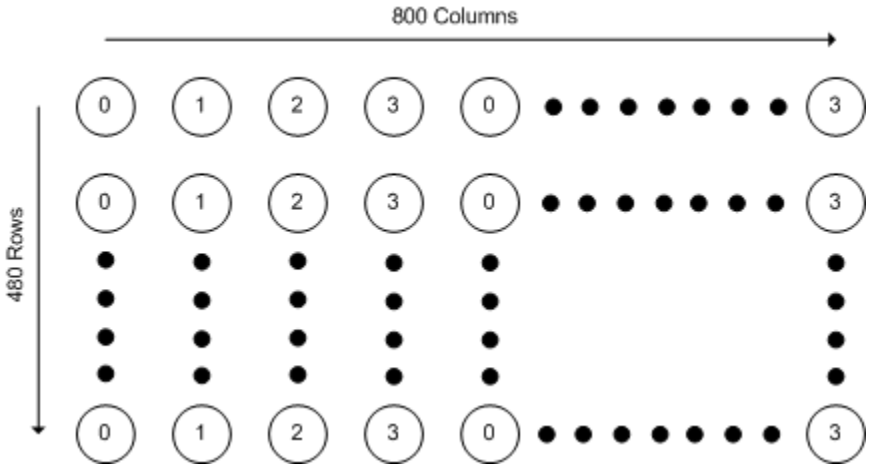
```
inline int int_mandelbrot(long long cr, long long ci,
int max_iter)
{
  long long xsqr=0, ysqr=0, x=0, y=0;
  int iter=0;
  // go ahead and shift these up to the new decimal
offset
  ci = ci<<28;
  cr = cr<<28;
  while( ((xsqr + ysqr) < 0x4000000000000000LL) &&
(iter < max_iter) )
  {
    xsqr = x * x;
    ysqr = y * y;
    y = ((2 * x * y) + ci) >> 28;
    x = (xsqr - ysqr + cr) >> 28;
    iter++;
return(iter);
}
```

The implementation is fixed point with all values pre-scaled by 0x10000000. The loop will continue until the number of iterations reaches 'max_iter' or $x^2 + y^2$ converges to the value of 4. This function is called for each pixel so for this design that would be 384000 times since the screen resolution is 800x480. The value of 'iter' is used as the index into the color palette which picks the color of the pixel displayed on the screen. Even though the main processor supports dynamic branch prediction and contains cache memory, this operation of filling the screen can be very time consuming.

The approach taken for the C2H accelerated version is to offload this algorithm to pipelined and parallel hardware. Each Mandelbrot engine contains dedicated multiply, addition, and subtraction logic to perform multiple operations in parallel. Each Mandelbrot accelerator operates on a quarter of the frame and is only called once per frame. The workload is

distributed on a pixel basis so each accelerator handles every fourth pixel and a Frame Stuffer accelerator is responsible for reassembling the pixels together to form a frame.

Figure 5–9. Mandelbrot Engine Pixel Interleaving



While the hardware accelerators are calculating a frame, the coordinates of the next frame are prepared by the main i.e. the Nios II processor. The hardware accelerators contain a C pragma that instructs the compiler to implement non-blocking accelerators which allows both the main processor and Mandelbrot hardware accelerators to operate in parallel. The main processor polls the Frame Stuffer accelerator to determine if the entire frame has been rendered.

Web Server Application

The web server application is built for the standard hardware system to show-case web applications running on Nios II and specifically highlighting remote reconfiguration over Ethernet.

This design example shows an HTTP server using the sockets interface of the NicheStack TCP/IP Stack, Nios® II Edition running on MicroC/OS-II real time operating system.

The server can process basic requests to serve HTML, JPEG, and GIF files from the Altera® read-only zip file system. Additionally, it demonstrates control of various board elements from the web page.

Operation

A Web Server serving webpages from a FAT file system residing on the SD card on the Nios II Embedded Evaluation Kit Development Board.

REQUIREMENTS

This example runs on the standard and web_server hardware designs provided with the kit. The NEEK development board is required.

PERIPHERALS USED

This example exercises the following peripherals:

- Triple speed Ethernet MAC (named **lan91c111** in SOPC Builder)
- STDOUT device (**UART** or **JTAG UART**)
- NEEK LCD display (optional, provides additional information.)

BOARD/HOST REQUIREMENTS

This example requires an Ethernet cable connected to the development board's RJ-45 jack, and a JTAG connection with the development board. If the host communication settings are changed from JTAG UART (default) to use a conventional UART, a serial cable between board DB-9 connector and the host is required.

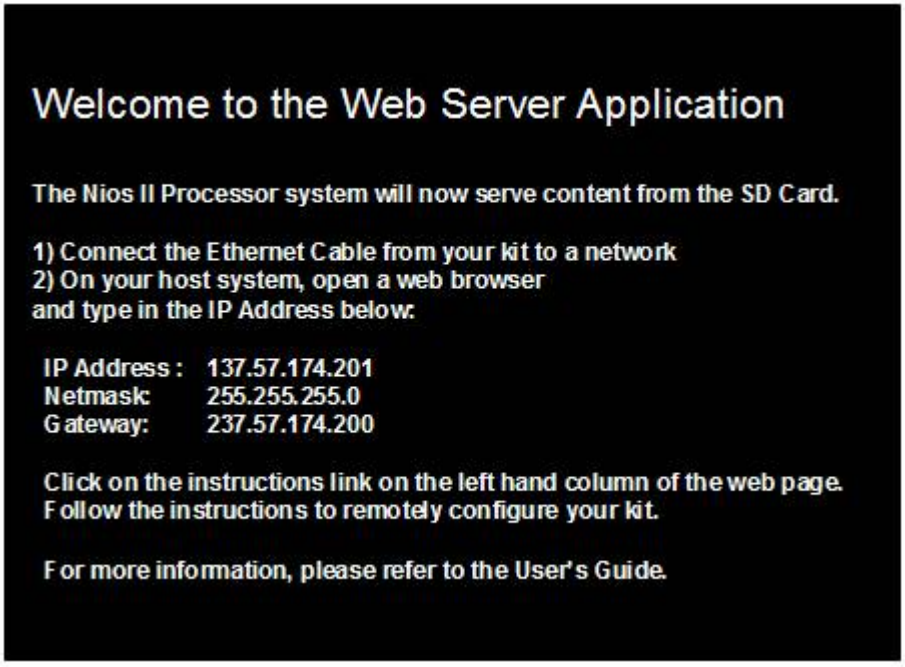
Operating Instructions

1. Using an **Ethernet cable** connect the **Ethernet RJ-45 jack** on the LCD Multimedia Daughtercard to the internet.

2. Using the application selector, scroll to select and **load** the **Altera Web Server**.

3. Please wait while the web server application establishes a connection to the internet and acquires an IP Address via DHCP. On completion a **welcome screen** appears with the acquired IP Address as shown below:

Figure 5–10. Welcome Screen



4. Open an explorer window and type: `http://<ip address>/` where ip address is the address shown in the command shell on the LCD Screen.

5. The web page you will see is called **index.html**. This page is being served up by the web server application from the **webserver_html** directory on the SD Card.

6. On the upper left hand side click on the link under **"Please read the instructions"**. Follow the instruction on **index.html** to update and remotely configure your Nios II Embedded Evaluation kit with a new design.

About Remote Reconfiguration Over Ethernet

Imagine you are working at your desk on a new FPGA design and your system is physically elsewhere (such as in the lab). Your FPGA design is now ready and both hardware and software needs to be updated in your

remote system. Having remote reconfiguration capability you can now update your system with the new FPGA image over an Ethernet connection. The implications of this are obvious; hardware and software updates can be made to any system, not just in your lab but in the field, a customer site or in the manufacturing facility as long as there is a persistent Ethernet connection!

Basic remote configuration features are included with this web server. The methodology chosen uses HTTP forms (HTML standard) to upload the hardware or software images and then a series of additional web pages to control the flow for programming flash and (optionally) resetting the system. Both the web content and the server side processing are provided to make this feature possible.

The basic steps are:

1. Fill out the upload form.
   - Select the image you'd like to upload
   - This is usually a .flash file produced (and tested) using the flash programmer.

   The .flash file format is an SREC file with addressing offset from the base address of your flash device. For this application, the **ext_flash** device is used.

2. Upload your selected image.
   - This could take a while, depending on the size of your image.
   - On completion, a new page will load giving you a new form for programming the flash.

3. Program the Flash
   - Click the button in the form located on the Remote Configuration section of this page and the flash will start programming.
   - Again, this could take a while, depending on the size of your image.
   - Upon completion, you'll see a **reset system** option in the Remote Configuration section.

4. Reset the System
   - If this is the only image you need to update, then reset your system to see if it functions.

- If you need to update more images (hardware or software) click the main hyperlink (takes you back to the main/index page) and repeat steps 1-3 as necessary.

Additional Information

If DHCP is available, the application will attempt to obtain an IP address from a DHCP server. Otherwise, a static IP address (defined in web_server.h) will be assigned after a time-out. This is an example HTTP server using NicheStack on MicroC/OS-II. The server can process basic requests to serve HTML, JPEG, and GIF files from the Altera FAT file system on an SD card. It is in no way a complete implementation of a full-featured HTTP server. This example uses the sockets interface.

A good introduction to sockets programming is the book Unix Network Programming by Richard Stevens. Additionally, the text "Sockets in C", by Donahoo & Calvert, is a concise and inexpensive text for getting started with sockets programming.

The HTTP server looks for content contained in the webserver_html directory at the top level of the SD card. Though default content is provided, the server will read any valid files that are placed into this directory.

Appendix A. Video Pipeline Data Flow

The video display subsystem embodied in the Nios II Embedded Evaluation Kit designs was intentionally-designed to be modular & flexible to make customization a snap. The design style used for the video pipeline highlights the use of several simple "microcores" which can be configured or customized for other video applications.

The video subsystem consists of these operational components, in roughly-logical order:

- A **frame-buffer** (which happens to reside in DDR memory)
- A memory-to-stream **DMA controller**, which reads memory 64 bits at a time and produces a stream of 64-bit data values.
- A width (**data format adapter**) to break the 64-bit stream into sequential 32-bit (pixel) values.
- A **FIFO**
- A **Pixel Format Converter**
- Another **Data Format Adapter**, to produce a stream of 8-bit values.
- A **sync-generator** (which you could think of as an LCD-display PHY)

If you actually look at the design, there are several other Avalon Streaming components in this flow. These have been omitted from this discussion for clarity because they are not **operational**. They are just timing-adapters which allow the operational pieces to fit together properly.

Starting from the end of the chain: The sync-generator just takes a stream of 8-bit-wide data values on its streaming input. Three consecutive 8-bit values make a single color pixel (R, G, B, R, G, B…) An start of packet (SOP)-pulse marks the start of each frame. The sync-generator drives external pins so that the pixel-stream appears on the display.

The DMA controller fetches pixel-data from the in-memory frame-buffer and drives it in row-major (raster) order on its streaming output-port and, through the video-subsystem pipeline, to the sync-generator termination.

The system has a FIFO because all systems like this always have FIFOs. It's there to "take up the slack" and keep the display fed even when the DDR memory is unavailable (due to contention, refresh, etc.).

The **Pixel Format Converter** subsystem assumes that the frame-buffer is storing 32-bit pixel values in (0:R:G:B) (8:8:8:8) format. The sync-generator, however, accepts 24-bit values. So the **Pixel Format Converter** takes-in a stream of 32-bit (0:R:G:B) pixels and produces a stream of 24-bit (R:G:B) values. This is done by throwing-away the unused 8 bits.

Once the Pixel Format Converter has produced a stream of 24-bit (8:8:8) (R:G:B) values, the data format adapter serializes the data into a stream of 8-bit (R, then G, then B) values. This is the input to the sync generator block which produces the horizontal and vertical timing signals.

The video pipeline used in the Nios II Standard System is just one implementation for video systems. FPGAs give you the power of flexibility to change this with just a few lines of code. For example, the next section describes what is necessary to support a 5:6:5 pixel format.

Creating a new 5:6:5 Pixel-Format component

The Nios II Standard System is designed to use a 32-bit 0:R:G:B data format. Suppose you wanted to change the entire display subsystem to work with 16-bit 5:6:5 pixels instead of 32-bit 0:R:G:B pixels. This can be accomplished with a few simple steps:

1. Copy the **Nios II Standard System** into your own project directory.
2. Create a new Verilog module called **pixel_converter_565** starting from the Verilog in **altera_avalon_pixel_converter.v**, modify this Verilog so it has a 16-bit data_in port and a 24-bit data_out port. All the other ports remain the same. You create the data_out value by inserting 8 new bits (3xR, 2xG, and 3xB) at the right points in the 16-bit word to "pad" it to a 24-bit word. You can use either zero- or LSB-padding.
3. Import your **Verilog module** into SOPC Builder using the Component Editor.
4. From an Nios II Standard System, replace the existing **Pixel Format Converter** with your new pixel_converter_565 component.
5. Edit the data-format-adapter named **lcd_64_to_32_bits_dfa**. Change its **(Output Interface Parameters)/(Data Symbols Per Beat)** from 4 to 2.
   - This changes its width-adaptation from 64 →32 to 64 →16.
   - Rename it to **lcd_64_to_16_bits_dfa**.
6. Regenerate your system.

Your software application is now responsible for filling the frame-buffer with aligned 16-bit 5:6:5 pixel data; and the firmware which controls the DMA must be modified to understand the different memory-buffer size implications of 16-bit (instead of 32-bit) pixels.

Appendix B. Application Selector Details

This section describes some details about the operation of the Application Selector.

SD Card

The Application Selector uses the SD Card for storing applications and data used by these applications (such as the pictures used by the picture viewer or the HTML pages used by the Web Server application). The SD Card must be formatted with the FAT 16 file system, and can be any capacity up to 2GB. Long file names are supported.

Application Files

Each loadable application consists of two flash files, and an optional text file, all stored on an SD Card.

The first flash file represents the software portion of the example and must be derived from an .ELF file as described in the section of this document titled "Creating Your Own Loadable Applications". This flash file can be named anything supported by the FAT16 file system, the only restriction being that the name must end with *_sw.flash*.

The second flash file represents the hardware portion of the example and must be derived from a *.SOF* file as described in the section of this document titled "Creating Your Own Loadable Applications". This file can be named anything supported by the FAT 16 file system, the only restriction being that the name must end with *_hw.flash*

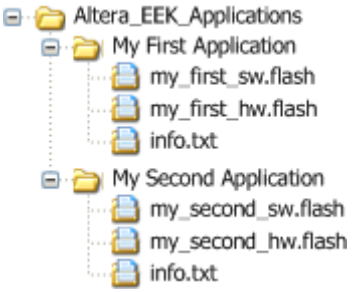
The optional "info.txt" file contains additional information about the application. In the application selector utility, touching the "Show Info" button while your application is highlighted, brings up a window showing the text contained in this file. The name of this text file must be *info.txt*, or the application selector will not recognize it.

SD Card Directory Structure

All loadable applications on the SD Card must be located in a top-level directory named Altera_EEK_Applications. Under the Altera_EEK_Applications directory, each application is located in its own subdirectory. The name of that subdirectory is important because the application selector utility uses that name as the title of the application when displaying it in the main menu. The name of the subdirectory is the title that will be displayed for your application in the menu. The subdirectory names can be anything so long as they adhere to the FAT file system long file name rules. Spaces are permitted.

Below is an example of how applications are organized on the SD Card.

Figure B–1. SD Card Directory Structure



CFI Flash

The Application Selector uses the on-board CFI flash to store several different things. Table B–1 shows a map of how the different sections of flash are used by the Application Selector.

Hardware images

CFI flash is used to store both the hardware image of the Application Selector itself, as well as up to 10 hardware images of applications which are being loaded.

The Application Selector hardware image is permanently stored in flash at offset 0x20000

Hardware images for the applications being loaded get written to flash at load time to an offset between 0x580000 and 0xD00000, depending on caching. Hardware image caching is described in more detail in the section titled "Hardware Image Caching"

Software Images

CFI flash is used to store the software images of both the Application Selector utility itself as well as software images of applications being loaded. All software images used by the application selector contain a boot copier which is pre-pended by the elf2flash utility during file conversion process described in the "Creating Your Own Loadable Applications" section. The boot copier copies the software code to program memory before running it.

The Application Selector software image is permanently stored in flash at offset 0x100000

Software images for the applications being loaded get written to flash at load time to offset 0x180000. Software images must be smaller than 4MB, or they will overwrite the application HW images located at offset 0x580000.

Application Boot Code

All applications which are loaded by the application selector must contain a Nios II CPU whose reset address is set to CFI flash at offset 0x0. For this reason, a generic bit of boot code is permanently programmed at offset 0x0 in the CFI Flash as part of the factory recovery image. This boot code is very small and only performs the following functions

- Flushes the Nios II instruction cache
- Flushes the Nios II instruction pipeline
- Branches to offset 0x180000

Offset 0x180000 is where the application software image is located after being loaded by the Application Selector, so when the FPGA is reconfigured, the Nios II CPU executes this boot code, which branches to the boot copier of the actual application software image, which then copies the application to program memory, then runs the application.

The Application Selector relies on a feature of the Cyclone III family of FPGAs called remote update. The remote update feature allows the Nios II CPU, to force the FPGA to reconfigure from a specific location in a parallel flash memory, such as the on-board CFI flash. The way the Application Selector is able to reconfigure itself with a new hardware image is by using Nios II to read the hardware image from the SD Card, program it to some location in CFI flash, then force the FPGA to reconfigure from that location using the remote update feature.

Flash Hardware Image Catalog

The CFI flash holds up to 10 of the most recently loaded application hardware images to speed the load times of applications which are loaded often. To keep track of which hardware images are currently

stored in flash, a flash image catalog is kept in CFI flash at offset 0x8000. The implementation details of this catalog are described in the Hardware Image Caching section below.

Table B–1. Memory Map of CFI Flash

Flash	Size	Flash Contents
0x000000 - 0x007FFF	32K	Application Boot Code
0x008000 - 0x00FFFF	32K	HW Image Catalog
0x010000 - 0x01FFFF	64K	Unused
0x020000 - 0x0FFFFF	896K	Selector HW Image
0x100000 - 0x17FFFF	512K	Selector SW Image
0x180000 - 0x57FFFF	4M	Application SW
0x580000 - 0xCFFFFF	7.5M	Application HW Images
0xD00000 - 0xFFFFFF	3M	Unused

Hardware Image Caching

Copying data from the SD Card to flash is slow due to both the read speed from the SD Card in SPI mode and the write speed of the CFI flash. However the remote update feature allows us to reconfigure the FPGA from anywhere in flash, so we can benefit by persistently holding (caching) a certain number of frequently used application hardware images in flash to avoid having to copy them from the SD Card every time the application is loaded.

The Application Selector utility can cache up to 10 application hardware images in CFI flash. When the user chooses an application to load from the SD Card using the Application Selector, the Application Selector first scans through its catalog of hardware images currently stored in CFI flash to see if any of them match the hardware image being requested. If one of the images cached in CFI flash does match, the Application Selector reconfigures from the offset of that cached hardware image instead of copying the image from SD Card to flash. This significantly reduces the load time.

Caching the hardware images requires the application selector to be able to quickly tell if an image in CFI flash is the same as one on the SD Card. To determine whether a hardware image in flash matches a hardware image on the SD Card, a 32-bit timestamp value is used as a tag. During the file conversion process, the sof2flash utility inserts a 32-bit timestamp in the hardware image *.flash* file as an S0-type record on the first line of the file. When the Application Selector is about to load a hardware image,

it inspects the *.flash* file on the SD Card. If the *.flash* file contains an S0 record on its first line which contains a 32-bit ASCII-encoded number, it is considered to be a valid timestamp tag.

The Application Selector then scans the flash catalog for entries which contain a matching timestamp. If a matching timestamp value is found, then it means the desired hardware image is already stored in flash, and can be used to directly reconfigure the FPGA without first copying it from the SD-Card into the flash. For details on the flash catalog, refer to the section below titled "Flash Hardware Image Catalog".

Flash Hardware Image Catalog

The flash hardware image catalog is a simple database which keeps track of what application hardware images are currently stored (cached) in flash. The flash catalog is located in sector 1 of the flash at offset 0x8000, and is 0x8000 (32K) bytes long.

The catalog mechanism uses a scheme referred to as "Zero = Spent, 'F' = Available", or ZSFA. This scheme avoids erasing entire flash sectors when only a few words need to be written to the flash. Using ZSFA, a word in the flash which is 0x0 is considered "spent" and cannot be used to store data. A word which is 0xFFFFFFFF is "available" since it is in its erased state. Every other value is considered a valid entry in the catalog.

The way ZSFA works is that whenever a catalog entry needs to be read, the sector is scanned from its lowest address until the first 0xFFFFFFFF value is encountered. Every non-zero value encountered along the way is a valid catalog entry. When a catalog entry needs to be written, the sector is scanned until the first 0xFFFFFFFF value is found, and the new catalog entry is written to that offset. To erase a catalog entry, you scan for it in the sector, then write 0x0 to it to mark it as "spent". The sector(s) containing the ZSFA catalog only need to be erased once enough data has been stored there that there are no more "available" entry spots available.

Each flash catalog entry consists of two sequential 32-bit words. The first word is the 32-bit timestamp value of a hardware image which is currently in flash. The second word is the 32-bit flash offset of the image itself. Entries are always created and erased as whole units, two 32-bit words at a time.

Creating Your Own Loadable Applications

It is easy to convert your own Nios II design into an application which is loadable by the Application Selector utility. All you need is a hardware image (a *Cyclone III 3C25 .SOF* file) and a software image which runs on that hardware (a *Nios II .ELF* file).

The only restrictions are:

1. The *.SOF* file must contain a CFI flash component.

2. The *.SOF* file must contain a Nios II CPU whose reset address is set to CFI Flash at offset **0x00000000**.

3. The size of the software image must be no larger than **4MB**.

☞ If you require a software image larger than 4MB, refer to the section of this document titled "Modifying the Application Selector"

Once you have your working *.SOF* and *.ELF* file pair, perform the following steps to convert them to a loadable application selector-compatible application.

1. Copy both the *.SOF* and *.ELF* files into a common directory of your choosing. This directory is where you will convert the files.

2. Copy the script:

   examples/application_selector/application_utilities/flash_file_conversion_script

   to the directory where you copied your .SOF and .ELF files. Optionally, copy it to a directory in the Nios II Command Shell search path i.e. *<nios2 install>/bin*

3. Open a Nios II Command Shell and change to the directory where you copied the *.SOF* and *.ELF* files.

4. Convert the *.ELF* and *.SOF* files by running the script:

   ```
   ./eek.sh <elf file>.elf <sof file>.sof
   ```

   The *eek.sh* script runs the Nios II Command Line *utilities sof2flash* and *elf2flash* to convert the *.SOF* and *.ELF* files to application selector-compatible *.FLASH* files.

   ☞ Feel free to open *eek.sh* in a text editor to see the exact commands which are run.

5. You will now see two new files in the directory, *<elf file>_sw.flash*, and *<sof file>_hw.flash*. These are the application files you will put on the SD Card

6. Now create a file named *info.txt* in the same directory.

This is the file which will be displayed in the Application Selector when the **Show Info** button is pressed for your application.

Fill *info.txt* with some descriptive text about your application's operation.

7. Create a new subdirectory and name it what you would like the title of your application to be shown as in the application selector.

8. Copy both *.flash* files and *info.txt* into the new directory.

9. Using an SD Card reader, copy the directory onto an SD Card into a directory named "*Altera_EEK_Applications*". The directory structure on the SD Card should look like this:

```
Altera_EEK_Applications
    <Name of Application>
        <elf name>_sw.flash
        <sof_name>_hw.flash
        info.txt
```

10. Place the SD Card in the Nios II Embedded Evaluation Kit, Cyclone III Edition board, and switch on the power. The Application Selector will start up, and you will now see your application appear as one of the selections

Rebuilding the Application Selector

This section describes how to rebuild the Application Selector utility from source code using the Nios II Software Build Tools. If you are new to developing software on the Nios II processor it is recommended that you first go through the tutorial **My First Nios II Software Tutorial**. This will walk you through compiling a simple project that runs on the Nios.

Create a BSP

The first thing that's needed to build the software project is a board support package (BSP). To create a BSP perform these steps:

1. Open a **Nios II Command Shell.**

2. Change to the directory:

```
altera/72/kits/cycloneIII_3c25_niosII_eval/
examples/application_selector/software_examples/
bsp/hal_application_selector
```

3. Run the command **./create_this_bsp.**

Build the project

The next step is to build the Application Selector project. To build the project, perform these steps:

1. In the **Nios II Command Shell**, change to the directory:

```
altera/72/kits/cycloneIII_3c25_niosII_eval/
examples/application_selector/software_examples/
app/application_selector
```

2. Run the command **make all.**

For more information regarding BSPs and the Nios II Software Build Tools, see **Chapter 3 of the Nios II Software Developer's Handbook**

http://www.altera.com/literature/hb/nios2/n2sw_nii52014.pdf

Build the boot code

To rebuild the boot code which runs when an application is loaded and run from the Application Selector, perform these steps:

1. Open a **Nios II Command Shell.**

2. Change to the directory:

```
altera/72/kits/cycloneIII_3c25_niosII_eval/
examples/application_selector/
application_utilities/app_selector_boot_code
```

3. Run the command **make.**

Modifying the Application Selector

This section discusses the parts of the Application Selector you can modify in order to tailor the utility to your needs

Changing the CFI flash map

If your application needs to use the CFI flash in a particular manner which is not compatible with the Application Selector's default flash layout, you can modify the way some things are mapped in flash fairly easily.

General Guidelines

If you choose to modify the flash map, take great care in ensuring that you leave enough space in each block for the data you intend to store there. Otherwise, you may overlap sections and the Application Selector utility may overwrite important data and cause a failure.

Also, it is a good idea to completely erase the flash before altering the flash map. This will prevent stale, unused data from accidentally causing errors in the Application Selector Utility.

Application Selector Hardware Image

One of the flash layout restrictions with the Application Selector is that the Application Selector hardware image itself must reside at byte offset 0x20000 in flash. It cannot be changed. This is because the Cyclone III FPGA always performs its first configuration after reset from offset 0x20000.

Application Hardware Images

The section of flash which is used to hold and cache loadable application hardware images can be adjusted. The adjustments can be made by editing the file:

```
altera/72/kits/cycloneIII_3c25_niosII_eval/examples/
application_selector/software_examples/app/
application_selector/src/app_selector.h
```

Edit the lines:

```
#define AS_HW_IMAGE_OFFSET_START        0x580000

#define AS_HW_IMAGE_OFFSET_END          0xD00000
```

to reflect what section of flash you would like to use to hold and cache application hardware images. Note that one hardware image consumes 0xC0000 bytes (6 flash sectors), so ensure that **AS_HW_IMAGE_OFFSET_END - AS_HW_IMAGE_OFFSET_START** is always greater than or equal to 0xC0000. The Application Selector will cache as many images in this section as it is able to fit. For instance, the default section is 0x780000 bytes in size (60 flash sectors), so it is able to cache up to 10 loadable application hardware images.

Application Selector Software Image

The default location of the Application Selector software image is flash offset 0x100000. This is necessary because flash offset 0x100000 is the reset address of the Nios II CPU in the Application Selector hardware image. It's recommended that you do not change the location of the Application Selector software image because it also requires changing the reset vector of the Nios II CPU in the Application Selector hardware image, and recompiling that design in Quartus II.

Application Software Image

The application software image can be relocated in flash by performing the following steps:

1. In a text editor, open the file:

   ```
   altera/72/kits/cycloneIII_3c25_niosII_eval/
   examples/application_selector/
   application_utilities/app_selector_boot_code/
   app_selector_boot_code.s
   ```

2. Edit the line

   ```
   #define SW_APP_CODE 0x180000
   ```

   to reflect the flash offset where you would like to put the loadable application software images. Ensure that there is enough space allocated at that offset to hold your application software images.

3. In a text editor, open the file:

   ```
   altera/72/kits/cycloneIII_3c25_niosII_eval/
   examples/application_selector/software_examples/
   app/application_selector/src/app_selector.h
   ```

4. Edit the line

   ```
   #define AS_SW_IMAGE_OFFSET 0x180000
   ```

   to reflect the flash offset where you would like to put the loadable application software images. Ensure that there is enough space allocated at that offset to hold your application software images.

   You will need to rebuild both the boot code and the application selector utility for these changes to take effect.

Flash Catalog

The flash catalog can be relocated in flash and size-adjusted by performing the following steps.

1. In a text editor, open the file:

   ```
   altera/72/kits/cycloneIII_3c25_niosII_eval/
   examples/application_selector/software_examples/
   app/application_selector/src/app_selector.h
   ```

2. Edit the lines

   ```
   #define AS_FLASH_IMAGE_CATALOG_OFFSET   0x8000
   #define AS_FLASH_IMAGE_CATALOG_SIZE     0x8000
   ```

   to reflect the flash offset where you would like to put flash catalog.

   You will need to rebuild the boot code for these changes to take effect.

Appendix C. Re-building the Factory Image

This section describes the process of rebuilding the factory recovery image from source files. You may wish to modify and rebuild the factory recovery image if you've modified the application selector or boot code and would like a single recovery file which includes your modifications. Keep in mind that any modifications you make to the application selector or boot code, may make them incompatible with existing applications.

Each portion of the factory recovery image is described below, with instructions on how to create it and program it to flash. The last section here, titled "Combining factory recovery image files", includes instructions for creating a single factory recovery image that you can program into flash at any time to restore the factory configuration of the Embedded Evaluation Kit board.

To perform the tasks illustrated in this section, you must first open a Nios II command shell.

Boot Code

The first portion of the factory recovery image is the application boot code, located at flash offset 0x0. Appendix B describes the functionality of the boot code and how to rebuild it from the source files.

Building the boot code produces a file named *app_selector_boot_code.srec*. This file can be directly programmed to flash using nios2-flash-programmer in the Nios II command shell. The command is:

```
nios2-flash-programmer --base=0x4000000
app_selector_boot_code.srec
```

Hardware Image Catalog

The hardware image catalog section of flash is located at offset 0x8000. This section holds the locations of the currently cached hardware images in flash. Any time a factory recovery is performed, this section of flash should be erased to ensure no stale catalog entries exist. To erase this section of flash, enter the command:

```
nios2-flash-programmer --base=0x4000000 --erase
0x8000+0x8000
```

After erasing this section, you may wish to read back the erased contents into a file, so that you can combine this file into the final factory recover image. The command to read back this section into a file named catalog.srec is:

```
nios2-flash-programmer --base=0x4000000 --read catalog.flash
    --read-bytes 0x8000+0x8000
```

Application Selector Hardware Image

The application selector hardware image section contains the FPGA hardware image for the application selector utility. This section is located at flash offset 0x20000. The FPGA gets configured with this image upon power-up and after a board reset. The file you will need to create this portion of the factory recovery image is named *cycloneIII_embedded_evaluation_kit_application_selector.sof*, and is located in the application selector's Quartus II project directory. The command needed to create the application selector hardware portion of the factory recovery image is:

```
sof2flash --activeparallel --offset=0x20000
--input=
cycloneIII_embedded_evaluation_kit_application_selector.sof
--output=appsel_hw.flash
```

To program this .flash file to flash, run the command:

```
nios2-flash-programmer --base=0x4000000 appsel_hw.flash
```

Application Selector Software Image

The final portion of the factory recovery image is the application selector software image. This section is located at flash offset 0x100000. The Nios II processor resets to this address and runs this code every time the FPGA gets configured with the application selector hardware image. The file you will need to create this portion of the factory recovery image is named *ext_flash.flash*, and is located in the application selector software project directory. You need to run the following command from the application selector software project directory to create *ext_flash.flash* if it does not already exist:

```
make flash
```

Once "ext_flash.flash" is created, you can program it into flash with the following command:

```
nios2-flash-programmer --base=0x4000000 ext_flash.flash
```

Combining factory recovery image files

Once you've created flash (or srec) files for all the sections of the factory recovery image, you can combine them all into one file using the `cat` command:

```
cat app_selector_boot_code.srec catalog.flash appsel_hw.flash
ext_flash.flash > temp_restore.flash
```

However, you are still not done. Some of the individual files we combined contained non-data records in them. Some non-data records, such as "S0" records cannot appear anywhere in an SREC file except for the beginning, so you want to remove all the non-data records from the final factory recovery image. Data record types, are S1, S2, and S3, so you want to remove all the other types of records (S0, S5, S7, S8, and S9).You can use the command "sed" to perform this task. Use the following command to remove all non-data records from the new factory recovery image file:

```
sed '/^S[05789]/ d' temp_restore.flash >
    restore_cycloneIII_3c25.flash
```

You can now restore the Embedded Evaluation Kit board to its factory state by running the command

```
nios2-flash-programmer --base=0x4000000
restore_cycloneIII_3c25.flash
```

Appendix D. Frequently Asked Questions

This section below explains the frequently asked questions of Nios II Embedded Evaluation Kit.

Why do I get the error "Can't find valid feature line for core SD_MMC_SPI_CORE (EC11_0002) in current license; Error: Error (10003): Can't open encrypted VHDL or Verilog HDL file" when I try to re-generate the Nios II Standard hardware design?

The Nios II Standard hardware design contains the SD MMC SPI CORE which is a component that has been provided by a third party vendor, El Camino. To compile this core in your SOPC Builder system, you will need to get a license from El Camino. However, if your particular application has no need to access the SD Card then you do not need to include the SD Card core in your system. Simply uncheck this core or delete it and re-generate the system. You should now be able to rebuild the hardware system without error.

When your hardware and software image is ready you can add your design to be loadable by the application selector by following the steps listed in the FAQ: "How do I add my own design so the Application Selector can find and run it?"

Where can I get the SD-Card Controller IP License?

If your design requires access to the on-board SD Card then you can request an evaluation license or purchase the SD-Card Controller IP, drivers and FAT File system from El Camino.

El Camino GmbH
Landshuter Str. 1
D-84048 Mainburg
Germany

Tel. +49 - 8751 - 8787 - 0
Fax +49 - 8751 - 842876
Web: www.elcamino.de
E-mail: info@elca.de

How do I add pictures so the Picture Viewer Application can find them?

1. Connect the SD-Card reader provided in the Nios II Embedded Evaluation Kit to your PC via a USB port.
2. Remove the SD-Card and place in the SD-Card Reader.
3. Add any .JPEG or .BMP file to the **"images"** folder on the SD Card.
4. Re-insert the SD-Card in the Nios II Embedded Evaluation Kit board.

The next time you run the Picture Viewer application, these new files will be found.

How do I add my own design so the Application Selector can find and run it?

The Nios II Embedded Evaluation Kit provides an elegant way to add designs such that a user can scroll through and select the design of choice using the application selector. Details of the application selector can be found in the 'readme.txt' located in the application_selector folder under examples in the Nios II Embedded Evaluation Kit directory.

To convert your own Nios II design into an application which is loadable by the Application Selector utility you will need the following

1. A hardware image (a Cyclone III 3C25 .SOF file)
2. A software image which runs on that hardware (a Nios II .ELF file).
3. An SD Card reader

For a step by step instructions refer to section in Appendix B of this document entitled "Creating Your Own Loadable Applications"

Where do I go to get more designs for the Nios II Embedded Evaluation Kit?

Be sure to check for the latest demos available for download to your Nios II Embedded Evaluation kit by visiting the web site: www.altera.com/nios2eval.

You will be able to download the designs to your local drive and add them to your SD Card by placing them in the SD Card folder entitled **Altera_EEK_Appliations**. The application selector should automatically detect these new designs and load them on to your kit.

How do I open a design example in the Nios II IDE?

Several example applications have been provided to you in source code form so that you can use them to learn how to develop your own software applications. The example applications have been provided in the Nios II Software Build flow format, i.e. in the form of application (APP) and board support package (BSP).

If you would like to open these Nios II Software Build flow projects to the Nios II IDE for debugging purposes, then you will import the app and bsp projects into the Nios II IDE.

For step by step instructions on importing a Nios II Software Build flow project in to the Nios II IDE, refer to the software tutorial "*My First Nios II Software Application*" in the *documents/tutorials/software_tutorials* folder of the Nios II Embedded Evaluation kit directory.

How do I restore the factory image?

To restore the factory image, perform the following steps:

1. Using the Quartus II Programmer, configure the FPGA with the SOF file:

   ```
   altera/72/kits/cycloneIII_3c25_niosII_eval/
   examples/application_selector/cycloneIII_embedded_
   evaluation_kit_applicatoin_selector.sof
   ```

2. Open a Nios II Command Shell and change to the directory:

   ```
   altera/72/kits/cycloneIII_3c25_niosII_eval/factory
   _recovery/flash_contents
   ```

3. In the Nios II Command Shell, program the factory image into flash with the command:

   ```
   nios2-flash-programmer ––base=0x04000000

   restore_cycloneIII_3c25.flash
   ```

If you get the error message: No CFI table found at address <address> Leaving target processor paused

Check that either the address is correct (hex four million - i.e. 6 zeroes after the 4) or that you have two "-" characters before "base".

4. You should now be able to reset the board to start the Application Selector.

How do I re-build the factory image?

To re-build the factory image refer to Appendix C, Re-building the Factory Image.

Additional Information

Revision History

The table below displays the revision history for the chapters in this user guide.

Chapter	Date	Version	Changes Made
All	November 2007	1.0.0	• First publication.

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Information Type	Contact *Note (1)*
Technical support	www.altera.com/mysupport/
Technical training	www.altera.com/training/
Technical training services	custrain@altera.com
Product literature	www.altera.com/literature
Product literature services	literature@altera.com
FTP site	ftp.altera.com

Note to table:
(1) You can also contact your local Altera sales office or sales representative.

Further Information

For other related information, refer to the following websites:

- For the Cyclone III handbook:

  www.altera.com/literature/lit-cyc3.jsp

- For the Cyclone III reference designs:

  www.altera.com/endmarkets/refdesigns/device/cyclone3/cyclone3-index.jsp

- For eStore if you want to purchase devices:

  www.altera.com/buy/devices/buy-devices.html

- For Cyclone III Orcad symbols:

www.altera.com/support/software/download/pcb/pcbpcb_index.html

- For Nios® II 32-bit embedded processor solutions:

  www.altera.com/technology/embedded/emb-index.html

Typographic Conventions

This document uses the typographic conventions shown below.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: **Save As** dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
"Subheading Title"	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: "Typographic Conventions."
`Courier type`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix `n`, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ● •	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.

Visual Cue	Meaning
	A warning calls attention to a condition or possible situation that can cause injury to the user.
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.