

My First Nios II Software

Tutorial



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

TU-01003-2.0

Copyright © 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.





Contents

Chapter 1. My First Nios II Software Design

Software and Hardware Requirements	
Download Hardware Design to Target FPGA	
Nios II SBT for Eclipse Build Flow	
Create the Hello World Example Project	
Build and Run the Program	
Edit and Re-Run the Program	
Debugging the Application	
Why the LED Blinks	
Board Support Package	
Next Steps	

Additional Information

Document Revision History	Info-1
How to Contact Altera	Info-1
Typographic Conventions	Info-1

1. My First Nios II Software Design



This tutorial provides comprehensive information to help you understand how to create a software project for a Nios II processor system in an Altera FPGA and run the software project on your development board.

The Nios[®] II processor core is a soft-core CPU that you download (along with other hardware components that comprise the Nios II system) onto an Altera FPGA. This tutorial introduces you to the basic software development flow for the Nios II processor. In the tutorial, you use a simple, pre-generated Nios II hardware system and create a software program to run on it.

The example Nios II hardware system provides the following necessary components:

- Nios II processor core
- Off-chip DDR memory interface to store and run the software
- USB serial link for communication between the host computer and the target hardware (typically using a USB-Blaster[™] cable)
- LED peripheral I/O (PIO)

Software and Hardware Requirements

This section assumes you have already installed the Quartus[®] II design software, the Nios II Embedded Design Suite, and your development kit software. Figure 1–1 shows an example of the installation directory structure.

🖣 triple_speed_ethernet_design									
<u>File Edit Vi</u> ew F <u>a</u> vorites <u>T</u> ools <u>H</u> elp									
🔇 Back 🔹 🛞 🖌 🏂 Search 🔊 Folders 🛄 🔹									
Address 🛅 C:\MyFirstNiosIISoftwareTutorial\triple_speed_ethern	et_de	sign			💙 🄁 Go				
Folders	х	Name 🔺	Size	Туре	Date Modified 🧧				
🖃 🧼 Local Disk (C:)	~	📾 tserd_3c120.map.summary	1 KB	SUMMARY File	10/22/2009 1:0				
	-	📾 tserd_3c120.pin	90 KB	PIN File	10/22/2009 1:0				
		tserd_3c120.qpf	2 KB	Quartus II Project File	10/22/2009 1:0				
		🐻 tserd_3c120.qsf	57 KB	QSF File	1/15/2010 10:3				
Documents and Settings		🐻 tserd_3c120.qws	1 KB	QWS File	1/15/2010 10:5				
		📾 tserd_3c120.sdc	12 KB	SDC File	10/22/2009 1:0				
		tserd_3c120.sof	3,459 KB	SOF File	10/22/2009 1:0				
		🖬 tserd_3c120.sta.rpt	16,081 KB	RPT File	10/22/2009 1:0				
🗉 🧰 MyDownloads		🖬 tserd_3c120.sta.summary	23 KB	SUMMARY File	10/22/2009 1:0				
MyFirstNiosIISoftwareTutorial		🖬 tserd_3c120.v	9 KB	V File	10/22/2009 1:0				
🖃 🧰 triple speed ethernet design		tserd_3c120_sopc.bsf	14 KB	Quartus II Block Sy	10/22/2009 1:0				
isopc_builder		🖬 tserd_3c120_sopc.html	200 KB	HTML Document	10/22/2009 1:0				
altmemphy-library		📾 tserd_3c120_sopc.ptf	273 KB	PTF File	10/22/2009 1:0				
👝 db		🖬 tserd_3c120_sopc.ptf.8.0	212 KB	0 File	10/22/2009 1:0				
🚞 ddr2 high performance controller-library		🖬 tserd_3c120_sopc.ptf.bak	246 KB	BAK File	10/22/2009 1:0				
E Clipse workspace		tserd_3c120_sopc.ptf.pre_ge	212 KB	PRE_GENERATION	10/22/2009 1:0				
Incremental_db		🖬 tserd_3c120_sopc.qip	1 KB	QIP File	10/22/2009 2:1				
🗉 🧰 ip		🖬 tserd_3c120_sopc.sopc	60 KB	SOPC File	10/22/2009 1:0				
🗉 🧰 software		tserd_3c120_sopc.sopcinfo	792 KB	SOPCINFO File	10/22/2009 1:0				
🗉 🫅 software_examples		🖬 tserd_3c120_sopc.v	1,555 KB	V File	10/22/2009 1:0				
i testbench		🔤 tserd_3c120_sopc_generatio	12 KB	File	10/22/2009 1:0				
triple_speed_ethernet-library		diserd_3c120_sopc_inst.v	4 KB	V File	10/22/2009 1:0				
I C MyProjects		📃 tserd_3c120_sopc_log.txt	6 KB	Text Document	10/22/2009 1:0				
myTools		🔟 uart.v	33 KB	V File	10/22/2009 1:0				
🗉 🧰 Program Files		國 user_dipsw_pio_8in.v	6 KB	V File	10/22/2009 1:0				
🗄 🧰 Program Files (x86)		🗐 version.txt	1 KB	Text Document	10/22/2009 1:0				
	~	<			>				

Figure 1–1. Example Installation Directory Structure

The tutorial describes how to use the Nios II tools with different development kits. Table 1–1 describes the target hardware design files and location for development kits the tutorial supports.

Table 1–1. Development Kit Information

Kit	File Type	File / Directory
Nios II Embedded Evaluation Kit,	Design files directory	Download the design example from the Standard Nios II Hardware Design Example page of the Altera website.
Cyclone III Edition	SRAM Object File	cyclonelll_3c25_niosll_standard.sof
(Cyclone III 3C25)	SOPC Information File	cyclonelll_3c25_niosll_standard_sopc.sopcinfo
Cyclone III Development Kit	Design files directory	<nios eds="" ii="" installation<br="">directory>\examples\verilog\niosII_cycloneIII_3c120\triple_speed_ethernet _design directory (1)</nios>
(Cyclone III 3C120)	SRAM Object File	tserd_3c120.sof
	SOPC Information File	tserd_3c120_sopc.sopcinfo

Kit	File Type	File / Directory
Stratix IV GX FPGA Development Kit	Design files directory	<nios eds="" ii="" installation<br="">directory>\examples\verilog\niosII_stratixIV_4sgx230\triple_speed_ethernet _design directory (1)</nios>
(Stratix IV GX	SRAM Object File	tserd_4sgx230.sof
4SGX230)	SOPC Information File	tserd_4sgx230_sopc.sopcinfo
Notes to Table 1–1:		
(1) The default -Nio	s II FDS installation directorys	location is C:\altera\ \nins2eds

Table 1–1. Development Kit Information

(1) The default *<Nios II EDS installation directory>* location is **C:\altera***<version>*\nios2eds

Before starting the tutorial, copy the design files directory described in Table 1–1 for your kit to the location where you plan to run the tutorial. Throughout the tutorial, <your project directory> refers to this directory.

Download Hardware Design to Target FPGA

The software that you build will be executed by a Nios II processor-based system in an FPGA. Therefore, the first step is to configure the FPGA on your development board with the pre-generated Nios II standard hardware system. Download the FPGA configuration file, that is, the SRAM Object File (.sof) that contains the Nios II standard system, to the board by performing the following steps:

- 1. Connect the board to the host computer using the USB download cable.
- 2. Apply power to the board.
- 3. Start the Nios II Software Build Tools (SBT) for Eclipse. On Windows computers, choose All Programs > Altera > Nios II EDS <version> > Nios II <version> Software Build Tools for Eclipse in the Windows Start menu.
- 4. Accept the default workspace. We will be changing this later.
- 5. On the Nios II Tools menu, click Quartus II Programmer.
- 6. Click Hardware Setup in the upper left corner of the Quartus II Programmer window. The Hardware Setup dialog box appears.
- 7. Select USB-Blaster from the Currently selected hardware list, as shown in Figure 1–2.
 - [7 If the download cable does not appear in the list, you must first install drivers for the cable. For information about download cables and drivers, refer to the Download Cables page of the Altera website.

Figure 1–2.	Hardware	Setup	Dialog	Box
-------------	----------	-------	--------	-----

ardware Settings JTAG Settin Select a programming hardware hardware setup applies only to l	 setup to use whe		ces. This programming
Currently selected hardware: Available hardware items:	USB-Blaster [l No Hardware USB-Blaster [l		
Hardware USB-Blaster	Server Local	Port USB-0	Add Hardware

- 8. Click Close. You return to the Quartus II Programmer window.
- 9. Click Auto Detect. The device on your board is detected automatically.
- 10. Click the first entry to highlight it. Refer to Figure 1–3 for the location of the first entry.
- 11. Click Change File.
- 12. Browse to *<your project directory>* and select the **.sof** programming file for your design shown in Table 1–1.
- 13. Click OK.
- 14. Turn on **Program/Configure** the programming file, as shown in Figure 1–3.

Figure 1-3. Quartus II Programmer

Hardware Setu	p USB-Blaster [USB-0]				Mode: JTA	4G		▼ Pro	igress:	1	00 %	
Enable real-time	ISP to allow background progra	amming (for MAX II dev	ices)									
Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP	Γ
Stop	C:/MyFirstNiosIISoftware	EP3C120F780	0114448F	DEADBEEF	V							
Auto Detect												
Delete												
Add File												
Change File	1											
Save File												
Add Device												
Up												
Down												
												_
pe Messag	ge Configuration succeed	led 1 derrice/	a) configura	ud.								
V INCO:		ed 1 device(ed operation(s)	s) configure	iu ii								
i) Info: i) Info: i) Info:	Successfully performe											

15. Click Start.

The **Progress** meter sweeps to 100% as the Quartus II software configures the FPGA. When configuration is complete, the FPGA is configured with the Nios II system, but it does not yet have a C program in memory to execute.

Nios II SBT for Eclipse Build Flow

The Nios II SBT for Eclipse is an easy-to-use GUI that automates build and makefile management, and integrates a text editor, debugger, the Nios II flash programmer, and the Quartus II Programmer. Software application templates included in the GUI make it easy for new software programmers to get started quickly.

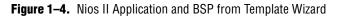
In this section, you use the Nios II SBT for Eclipse to compile a simple C language example software program to run on the Nios II standard system configured in the FPGA on your development board. You create a new software project, build it, and run it on the target hardware. You also edit the project, re-build it, and set up a debugging session.

For a complete tutorial on using the Nios II SBT for Eclipse to develop programs, refer to the software development tutorial in the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook*.

Create the Hello World Example Project

In this section, you create a new Nios II application project from an installed example. To begin, perform the following steps in the Nios II SBT for Eclipse:

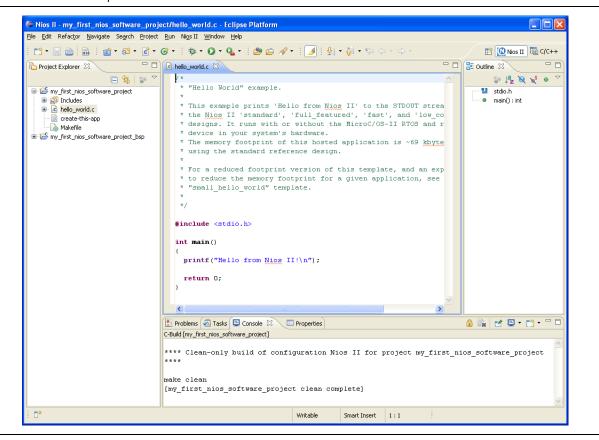
- 1. Return to the Nios II SBT for Eclipse.
 - You can close the Quartus II Programmer or leave it open in the background if you want to reload the processor system onto your development board quickly.
- 2. Create a new workspace in the *<your project directory>* so that the software resides under its hardware project. To do so, perform the following steps:
 - a. On the File menu, point to **Switch Workspace**, and click **Other**. The **Workspace Launcher** dialog box appears.
 - b. Click Browse to display the Select Workspace Directory dialog box, then navigate to *<your project directory>*.
 - c. Click Make New Folder.
 - d. Type eclipse_workspace and press Enter.
 - e. Click OK to exit the Select Workspace Directory dialog box.
 - f. Click **OK** to exit the **Workspace Launcher** dialog box. The Nios II Software Build Tools for Eclipse restarts in the new workspace.
- 3. On the File menu, point to New, and click Nios II Application and BSP from Template. The Nios II Application and BSP from Template wizard appears.
- 4. For SOPC Information File name, browse to *<your project directory>* and open the SOPC Information File (**.sopcinfo**) for your design shown in Table 1–1.
 - Every Nios II software project needs a system description of the corresponding Nios II hardware system. For the Nios II SBT for Eclipse, this system description is contained in a **.sopcinfo** file.
- 5. In the **Project name** box, type my_first_nios_software_project.
- 6. In the **Templates** list, select the **Hello World** project template. Figure 1–4 shows the wizard.



oject name: my_first_nios_software_project Use default location Project location: C:\MyFirstNiosIISoftwareTutorial\triple_speed_ethernet_design\software\my_first_nios_software_project	
oject name: my_first_nios_software_project Use default location Project location: C:\MyFirstNiosIISoftwareTutorial\triple_speed_ethernet_design\software\my_first_nios_software_project Project template Templatea Template description Hello World prints 'Hello from Nios II' to STDOUT. Blank Project Binark Project Binark Project For details, click Finish to create the project and refer to the readme.txt file in the project directory. Hello World Small Hello World Small	
Use default location Project location: C1/MyFirstNiosIISoftwareTutorial(triple_speed_ethernet_design\software\my_first_nios_software_project Project location: C1/MyFirstNiosIISoftwareTutorial(triple_speed_ethernet_design\software\my_first_nios_software_project Project location: CliphyFirstNiosIISoftwareTutorial(triple_speed_ethernet_design\software\my_first_nios_software_project Project template Blank Project Board Diagnostics Court Binary Hello Freestanding Hello MicroC/OS-II For details, click Finish to create the project and refer to the readme.txt file in the project directory. Hello World Small	
Blank Project Hello World prints 'Hello from Nios II' to STDOUT. Board Diagnostics This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. Hello Freestanding Hello MicroC/OS-II Hello MicroC/OS-II For details, click Finish to create the project and refer to the readme.txt file in the project directory. Hello World Small The BSP for this template is based on the Altera HAL operating system.	
NTP Client For information about how this software example relates to Nios II hardware design examples, Simple Socket Server refer to the Design Examples page of the Nios II documentation available with your installation at: Web Server <installation_directory>/nios2eds/documents/index.htm.</installation_directory>	

- Click Finish. The Nios II SBT for Eclipse creates the my_first_nios_software_project project and returns to the Nios II perspective.
- 8. In the Project Explorer view, expand **my_first_nios_software_project**. Doubleclick **hello_world.c** to view the source code. Figure 1–5 shows the Nios II perspective.

Figure 1–5. Nios II Perspective with Source Code Editor



When you create a new project, the Nios II SBT for Eclipse creates the following new projects in the Project Explorer view:

my_first_nios_software_project is the application project. This project contains the source and header files for your application.

- my_first_nios_software_project_bsp is the board support package (BSP) for your Nios II system hardware. The BSP includes the following details:
 - Component device drivers for your Nios II hardware system
 - newlib C library, which is a richly-featured C library for embedded systems
 - Nios II software packages
 - Nios II hardware abstraction layer (HAL)
 - NicheStack TCP/IP Network Stack, Nios II Edition
 - Nios II host file system
 - Nios II read-only zip file system
 - Micrium's MicroC/OS-II real-time operating system (RTOS)
 - **system.h**, which is a header file that encapsulates your hardware system
 - alt_sys_init.c, which is an initialization file that initializes the devices in the system
 - linker.h, which is a header file that contains information about the linker memory layout.

Build and Run the Program

Perform the following steps to build and run the program:

- To build the program, right-click the my_first_nios_software_project project in the Project Explorer view, and click Build Project. The Build Project dialog box appears and the Nios II SBT for Eclipse begins compiling the project. When compilation completes, the message "Build completed" appears in the Console view. The completion time varies depending on your system.
- 2. To run the program, right-click **my_first_nios_software_project**, point to **Run As**, and click **Nios II Hardware**. The Nios II SBT for Eclipse downloads the program to the FPGA on the target board and executes the code. The message "Hello from Nios II!" displays in the Nios II Console view.
- If the **Run Configurations** dialog box appears, click the **Target Connection** tab and verify the connection to the board. If no connection is shown, click **Refresh Connections**. When a connection appears, click **Run**. For more information about run and debug configurations, refer to the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook*.

Figure 1–6 shows the Nios II Console view at the right side of the Nios II perspective.

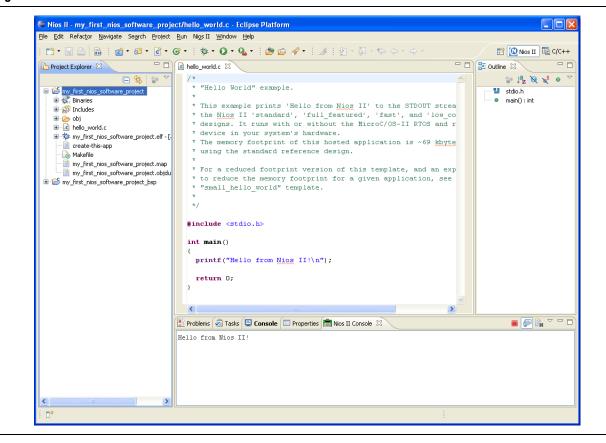


Figure 1–6. Nios II Console View

Now that you have created, compiled, and run your first software program, you can perform additional operations, such as configuring the BSP properties, editing and rebuilding the application, and debugging the source code.

Edit and Re-Run the Program

You can modify the source code in the Nios II SBT for Eclipse, rebuild the project, run the program, and observe your changes executing on the target board. In this section, you add code that makes an LED on your development board blink.

For more information regarding how the LED blinks, refer to "Debugging the Application" on page 1–11.

Perform the following steps to modify and re-run the program:

1. In the **hello_world.c** file, replace the existing code with the following code:

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
    int count = 0;
    int delay;
    printf("Hello from Nios II!\n");
```

```
while(1)
{
    IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count & 0x01);
    delay = 0;
    while(delay < 2000000)
    {
        delay++;
    }
        count++;
    }
    return 0;
}
2. Save the file.</pre>
```

- 3. Right-click **my_first_nios_software_project** in the Project Explorer view, point to **Run As**, and click **Nios II Hardware**.
 - You do not need to build the project manually; the Nios II SBT for Eclipse automatically rebuilds the program before downloading it to the FPGA.
- 4. Observe the LED blinking on your development board.

Debugging the Application

To debug your application, perform the following steps:

- 1. Right-click **my_first_nios_software_project** in the Project Explorer view, point to **Debug As**, and click **Nios II Hardware**.
 - If the **Debug Configurations** dialog box appears, click the **Target Connection** tab and verify the connection to the board. If no connection is shown, click **Refresh Connections**. When a connection appears, click **Debug**. For more information about run and debug configurations, refer to the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook*.
- 2. If the **Confirm Perspective Switch** message box appears, click **Yes**. The context switches to the Nios II Debug perspective.

After a moment, the main() function appears in the editor. A blue arrow next to the first line of code indicates that execution stopped at that line.

3. On the Run menu, click **Resume** to resume execution.

When debugging a project in the Nios II SBT for Eclipse, you can pause, stop, or single step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

- To return to the Nios II perspective from the Nios II Debug perspective, click the two arrows (>>) in the upper right corner of the GUI, and click **Nios II**.
- For more information about debugging software projects in the Nios II SBT for Eclipse, refer to the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook*.

Why the LED Blinks

The Nios II system description header file, **system.h**, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios II hardware system. The **system.h** file resides in the **my_first_nios_software_project_bsp** directory. Open the **system.h** file and locate the

LED_PIO_BASE macro.

LED_PIO_BASE contains the base address of the PIO peripheral controlling the LEDs. The Nios II processor controls the PIO ports (and therefore the LED) by reading and writing to the register map. For the PIO, there are four registers: data, direction, interruptmask, and edgecapture. To turn the LED on and off, the application writes to the PIO data register.

The PIO core has an associated software file **altera_avalon_pio_regs.h**. This file defines the core's register map, providing symbolic constants to access the low-level hardware. The **altera_avalon_pio_regs.h** file resides in the **my_first_nios_software_project_bsp** directory in the **drivers\inc** subdirectory.

When you include the **altera_avalon_pio_regs.h** file, several useful functions that manipulate the PIO core registers are available to your program. In particular, the function IOWR_ALTERA_AVALON_PIO_DATA(base, data) can write to the PIO data register, turning the LED on and off.

The PIO is just one of many SOPC Builder peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to *Volume 5: Embedded Peripherals* of the *Quartus II Handbook*.

When developing your own designs, you can use the software functions and resources that are provided with the Nios II HAL.

 Refer to the *Nios II Software Developer's Handbook* for extensive documentation on developing your own Nios II processor-based software applications.

Board Support Package

This section explores configuring your BSP.

For BSP properties changes to take affect, you must regenerate your BSP before rerunning your program by either clicking **Generate** in the BSP Editor or by rightclicking the **my_first_nios_software_project_bsp** project in the Project Explorer view, pointing to **Nios II**, and clicking **Generate BSP**.

To access the most common BSP properties, perform the following steps:

- 1. In the Nios II SBT for Eclipse, right-click my_first_nios_software_project_bsp and click Properties. The Properties for my_first_nios_software_project_bsp dialog box appears.
- 2. Click **Nios II BSP Properties**. The most-common settings related to how the program interacts with the underlying hardware appear. Figure 1–7 shows the Nios II BSP Properties dialog box.

Figure 1–7. Nios II BSP Properties

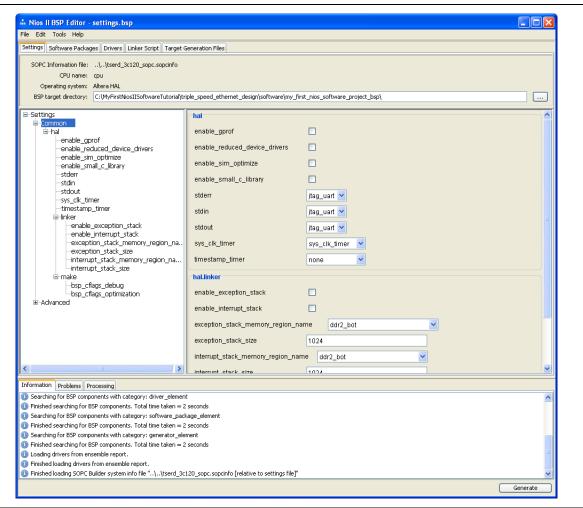
e filter text	Nios II BSP Proper	ties	← → ⇒ → →
Resource			
Builders	Flags		
C/C++ Build	Defined symbols:	none	
C/C++ General	Undefined symbols:	none	
Project References	Assembler flags:	-Wa,-gdwarf2	
Run/Debug Settings	Warning flags:	-wall	
Task Repository			
	User flags:	none	
	Optimization level: Reduced device d ModelSim only, no Support C++ GPROF support Small C library	ivers	BSP Editor
			Apply

An extensive set of build properties and options are available using the BSP Editor. To access all the BSP properties, perform the following steps:

 In the lower right corner of the Properties for my_first_nios_software_project_bsp dialog box, click BSP Editor. The Nios II BSP Editor appears. Figure 1–8 shows the Nios II BSP Editor.

You can also access the Nios II BSP Editor from the Nios II menu.

Figure 1-8. Nios II BSP Editor



- 2. Click the **Settings** tab to see the available settings. For example, you can set the interfaces to use for stdio, stdin, and stderr.
- 3. Click the **Software Packages** tab to see the software packages, such as files systems, graphics libraries, network stacks, available in your BSP.
- 4. Click the **Drivers** tab to see the available drivers for the Altera-provided intellectual property (IP) in your system.
- 5. Click the Linker Script tab to see the linker section memory assignments. The linker section assignments determine what memory is used to store the compiled executable program when the my_first_nios_software_project program runs. Figure 1–9 shows the Linker Script tab of the Nios II BSP Editor.

Figure 1-9. Linker Script Tab of the Nios II BSP Editor

ddr2_bot_BEFORE_EXCEPTION 0x0000000 descriptor_memory 0x0800200 descriptor_memory 0x0800200 exceptions 0x0000000 heap 0x0000000 exceptions ddr2 heap ddr2 icodata icodata icodata	Memor - 0x17FFFFFF ddr2_b - 0x1000011F ddr2_b - 0x0003FFF descri - 0x003FFFFFF ext_f1 - 0x03FFFFFF ext_f1 - 0x0000001F ext f1 bot bot bot bot	bot iptor_memory lash lash lash dari ddri ddri ddri ddri ddri ddri ddri	Size (bytes) Size (bytes)	Offset (bytes) 288 0 32 0	Restore Defaults
ddr2_bot 0x100012 ddr2_bot_DEFORE_EXCEPTION 0x1000002 ddr2_bot_DEFORE_EXCEPTION 0x0000002 ext_flash 0x0000002 ext_flash 0x0000000 ext_flash 0x0000000 ext_flash 0x0000000 ext_flash 0x0000000 ext_flash 0x0000000 0x00000000 0x00000000 0x00000000	- 0x17FFFFF ddr2_b - 0x100001FF ddr2_b - 0x00003FFF descri - 0x003FFFFFF ext_f1 - 0x0000001F ext f1 - 0x0000001F ext f1 bot bot bot bot	bot bot ijtor_memory lash lash dash dati dati dati dati dati dati	nory Device Name 2_bot flash 2_bot 2_bot	288 0 0 32	Remove Restore Defaults Memory Usage Memory Map Add Remove
ddr2_bot_BEFORE_EXCEPTION 0x0000000 descriptor_memory 0x0800200 descriptor_memory 0x0800200 exceptions 0x0000000 heap 0x0000000 exceptions ddr2 heap ddr2 icodata icodata icodata	 c.x1000011F ddr2_b dx08003FF descrit dx0807FFFF ext fl c.x087FFFF ext fl c.x00000001F ext fl ext fl bot bot bot 	bot iptor_memory lash lash lash dari ddri ddri ddri ddri ddri ddri ddri	288 8192 6710883 32 nory Device Name 2 bot flash 2 bot 2 bot 2 bot	0 0 32	Restore Defaults Memory Usage Memory Map Add Remove
descriptor_memory 0x06002000 ext_flash 0x00000020 reset 0x00000000 linker 0x000000000 linker 0x00000000000000000000000000000000000	- CNOBODSFFF descri - OxOBFFFFF ext_fl - OXODODOOLF ext fl Region Name bot bot bot	iptor_memory lash lash dari dari dari dari dari dari dari dari	nory Device Name 2 bot flash 2 bot 2 bot	0 32	Add Remove
ext_flash 0x0000002 reset 0x0000000 Linker Section Name Linker .bss ddr2 .entry reset .exceptions ddr2 .rodata ddr2 .rodat	Region Name bot bot bot	lash lash Men ddr2 ddr2 ddr2 ddr2 ddr2	nory Device Name 2_bot flash 2_bot 2_bot 2_bot	32	Memory Usage Memory Map Add Remove
Linker Section Name A Linker bss ddr2 exceptions ddr2 exceptions ddr2 rodata ddr2 rodata ddr2 stack ddr2 stack ddr2 stack ddr2 ferayed out entries are automatically created at generate ti Information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 24	Region Name bot bot bot bot	lash Men ddr2 ddr2 ddr2 ddr2 ddr2	nory Devke Name 2_bot flash 2_bot 2_bot		Add
Linker Section Name Linker .bss ddr2	Region Name bot bot bot bot bot	Men ddr2 ddr2 ddr2 ddr2 ddr2	nory Device Name 2 bot flash 2 bot 2 bot	0	Memory Map Add Remove
bas ddr2, entry reset exceptions ddr2 heap ddr2 rodata ddr2 stack ddr2 text ddr2 text ddr2 back ddr2 text ddr2 frayed out entries are automatically created at generate ti information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 2 d	bot bot bot bot bot	ddr2 ext_ ddr2 ddr2 ddr2	2_bot flash 2_bot 2_bot		Add Remove
bas ddr2, entry reset exceptions ddr2 heap ddr2 rodata ddr2 stack ddr2 text ddr2 text ddr2 back ddr2 text ddr2 frayed out entries are automatically created at generate ti information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 2 d	bot bot bot bot bot	ddr2 ext_ ddr2 ddr2 ddr2	2_bot flash 2_bot 2_bot		Remove
entry rest exceptions ddr2 .heap ddr2 .rodata	bot bot bot bot	ext_ ddr2 ddr2 ddr2	flash 2_bot 2_bot		
.exceptions ddr2 .heap ddr2 .rodata ddr2 .rwdata ddr2 .rwdata ddr2 .stack ddr2 .stack ddr2 .text ddr2 .text ddr2 .formetion Problems Processing	bot bot bot bot	ddr2 ddr2 ddr2	2_bot 2_bot		Restore Defaults
.heap ddr2 .rodata ddr2 .rodata ddr2 .rodata ddr2 .stack .stack .stack <	bot bot bot	ddr2 ddr2	2_bot		Restore Defaults
.rodata ddr2 .rodata ddr2 .rodata ddr2 .stack ddr2 .text ddr2 Grayed out entries are automatically created at generate ti Information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 24	bot bot	ddr2			
.rwdata ddr2 .stack ddr2 .text ddr2 Grayed out entries are automatically created at generate ti Information Problems Processing ③ Searching for BSP components with category: driver_element ④ Finished searching for BSP components. Total time taken = 2 ±	bot		2_bot		
Grayed out entries are automatically created at generate ti Information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 2 d		ddr2	ddr2_bot		
Grayed out entries are automatically created at generate ti Information Problems Processing Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 2 :			2_bot		
Grayed out entries are automatically created at generate ti Information Problems Processing © Searching for BSP components with category: driver_element I Finished searching for BSP components. Total time taken = 2 ±			2_bot 2 bot		
 Searching for BSP components with category: driver_element Finished searching for BSP components. Total time taken = 2 s 					
	ne. They are not editable or p	persisted in the BSP s	ettings file.		
Finished searching for BSP components. Total time taken = 2 s					
					-
Searching for BSP components with category: software_packa	ge_element				
I Finished searching for BSP components. Total time taken = 2 s	econds				
③ Searching for BSP components with category: generator_elem	ent				
Finished searching for BSP components. Total time taken = 2 :	econds				
Loading drivers from ensemble report.					
Finished loading drivers from ensemble report.					
Finished loading SOPC Builder system info file "\\tserd_3c1:	.0_sopc.sopcinfo [relative to setti	tings file]"			~
		-			Generate

- 6. Click the **Target Generation File** tab to see the files that get added to your BSP at build time.
- 7. If you have made any changes to your BSP, click Generate to update your BSP.
- 8. On the File menu, click **Exit**.
- 9. Click **OK** to close the **Properties for my_first_nios_software_project_bsp** dialog box.

Next Steps

The following documents provide next steps to further your understanding of the Nios II processor:

 Developing Software for Nios II Processors—These short, online software tutorials walk you through the basics of developing software for the Nios II processor. Access these tutorials on the Embedded Training Resources page of the Altera website.

- Nios II Software Developer's Handbook This handbook provides a complete reference on developing software for the Nios II processor.
- The "Getting Started" section of the Getting Started with the Graphical User Interface chapter of the Nios II Software Developer's Handbook—This tutorial teaches in detail how to use the Nios II SBT for Eclipse to develop, run, and debug new Nios II application projects.
- *Nios II Processor Reference Handbook*—This handbook provides a complete reference for the Nios II processor hardware.
- Volume 4: SOPC Builder of the Quartus II Handbook—This volume provides a complete reference on using SOPC Builder, including building memory subsystems and creating custom components.
- *Volume 5: Embedded Peripherals* of the *Quartus II Handbook*—This volume contains details on the peripherals provided with the Nios II Embedded Design Suite.

For a complete list of documents available for the Nios II processor, refer to the Literature: Nios II Processor page of the Altera website.



Document Revision History

The following table shows the revision history for this document.

Date	Version	Changes
January 2010	2.0	Revised for Nios II Software Build Tools for Eclipse.
July 2008	1.4	Removed command line flow to simplify tutorial.
May 2007	1.3	Minor fixes.
May 2007	1.2	Minor fixes.
May 2007	1.1	Minor fixes.
May 2007	1.0	Initial release.

How to Contact Altera

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

Contact <italic> (Note 1)</italic>	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product literature	Website	www.altera.com/literature
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, Save As dialog box. For GUI elements, capitalization matches the GUI.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, \qdesigns directory, d: drive, and chiptrip.gdf file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, AN 519: Stratix IV Design Guidelines.

Visual Cue	Meaning	
Italic type	Indicates variables. For example, $n + 1$.	
	Variable names are enclosed in angle brackets (< >). For example, <i><file name=""></file></i> and <i><project name="">.pof</project></i> file.	
Initial Capital Letters	Indicates keyboard keys and menu names. For example, the Delete key and the Options menu.	
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."	
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, data1, tdi, and input. The suffix n denotes an active-low signal. For example, resetn.	
	Indicates command line commands and anything that must be typed exactly as it appears. For example, c:\qdesigns\tutorial\chiptrip.gdf.	
	Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword SUBDESIGN), and logic function names (for example, TRI).	
1., 2., 3., and a., b., c., and so on	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.	
	Bullets indicate a list of items when the sequence of the items is not important.	
IP	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 your work.	
WARNING	A warning calls attention to a condition or possible situation that can cause you injury.	
4	The angled arrow instructs you to press the Enter key.	
	The feet direct you to more information about a particular topic.	