



My First Nios II Software Tutorial



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>

Document Date:

July 2008

TU-01003-1.4

Copyright © 2008 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 Corporation. 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.



Printed on recycled paper



I.S. EN ISO 9001



Contents

How to Contact Altera	v
Typographic Conventions	v
Introduction	1-1
Software and Hardware Requirements	1-1
Download Hardware Design to Target FPGA	1-4
Nios II IDE Build Flow	1-6
Create the hello_world Example Project	1-7
Build and Run the Program	1-10
Edit and Re-Run the Program	1-12
Why the LED Blinks	1-13
Debugging the Application	1-15
Configure System Library	1-15
Next Steps	1-17



About this Tutorial

This tutorial provides comprehensive information that will help you understand how to create an Altera® FPGA design and run it on your development board.

How to Contact Altera

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

Information Type	Contact (1)
Technical support	www.altera.com/mysupport/
Technical training	www.altera.com/training/custrain@altera.com
Product literature	www.altera.com/literature/
Altera 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.

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: f_{MAX} , lqdesigns directory, d: drive, chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: < <i>file name</i> >, < <i>project name</i> >.pof file.

Visual Cue	Meaning
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: <code>data1</code> , <code>tdi</code> , <code>input</code> . Active-low signals are denoted by suffix <code>n</code> , e.g., <code>resetn</code> . Anything that must be typed exactly as it appears is shown in Courier type. For example: <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword <code>SUBDESIGN</code>), as well as logic function names (e.g., <code>TRI</code>) 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	The caution indicates required information that needs special consideration and understanding and should be read prior to starting or continuing with the procedure or process.
 WARNING	The warning indicates information that should be read prior to starting or continuing the procedure or processes
↵	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Introduction

The Nios® II processor core is a soft-core central processing unit (CPU) that you program (along with other hardware components that comprise the Nios II system) onto an Altera® field programmable gate array (FPGA). This tutorial introduces you to the basic software development flow for the Nios II processor. You will use a simple pre-generated Nios II standard hardware system and create a software program to run on it.

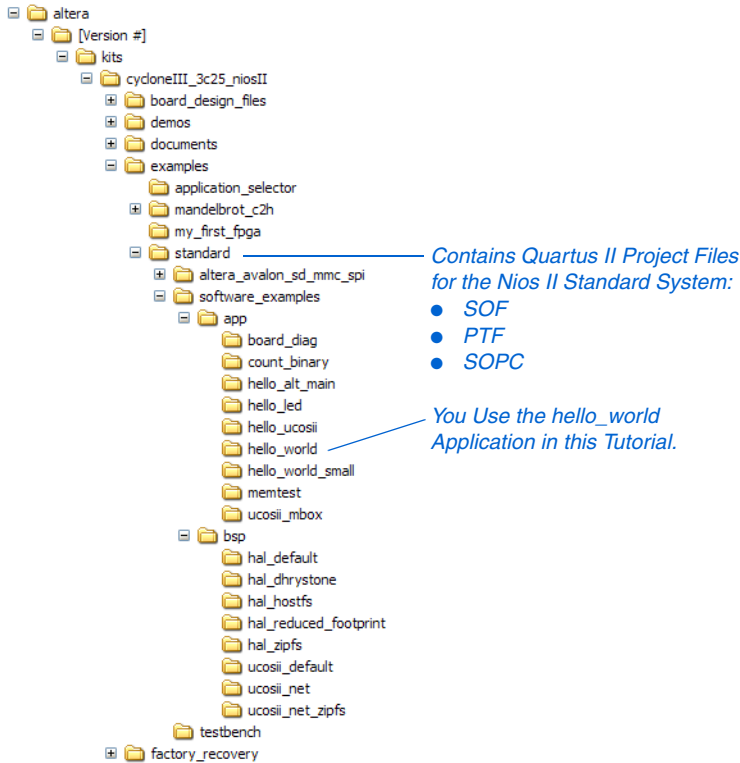
The example Nios II standard hardware system provides the following necessary components:

- Nios II processor core
- Off-chip memory interface to store and run the software
- JTAG link for communication between the host computer and 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 CD-ROM software. [Figure 1-1](#) shows an example of the default installation directories.

Figure 1–1. Default Nios II Embedded Evaluation Kit Installation Directory



As you go through the tutorial, *<installation directory>* represents `\altera\<version number>\kits\<kit name>`. For example, using the example in Figure 1–1, *<installation directory>* is `\altera\<version number>\kits\cycloneIII_3c25_niosII`.

This document describes how to use the Nios II tools with different development kits. Table 1–1 describes the kit-specific information, which is referenced throughout the text.

Table 1–1. Project Directories and Filenames

Kit	Description	
Arria GX Development Kit	Nios II Standard Design	<installation directory>\examples\ArriaGX_PCle_Nios_Standard
	FPGA Programming File	<Nios II standard design>\Nios_Standard_time_limited.sof
	PTF File	<Nios II standard design>\Arria_GX_Standard.ptf
Cyclone III Starter Kit	Nios II Standard Design	<installation directory>\examples\cycloneIII_3c25_start_niosII_standard
	FPGA Programming File	<Nios II standard design>\cycloneIII_3c25_start_niosII_standard.sof
	PTF File	<Nios II standard design>\cycloneIII_3c25_start_niosII_standard.sof
Nios II Embedded Evaluation Kit	Nios II Standard Design	<installation directory>\examples\standard
	FPGA Programming File	<Nios II standard design>\cycloneIII_embedded_evaluation_kit_standard.sof
	PTF File	<Nios II standard design>\cycloneIII_embedded_evaluation_kit_standard.ptf
Cyclone III Development Kit	Nios II Standard Design	<installation directory>\examples\cycloneIII_3c120_dev_niosII_standard
	Project Filename	<Nios II standard design>\cycloneIII_3c120_dev_niosII_standard.sof
	PTF File	<Nios II standard design>\cycloneIII_3c120_dev_niosII_standard.ptf
Stratix III Development Kit	Nios II Standard Design	<installation directory>\examples\stratixIII_3s1150_dev_niosII_standard
	Project Filename	<Nios II standard design>\stratixIII_3s1150_dev_niosII_standard.sof
	PTF File	<Nios II standard design>\stratixIII_3s1150_dev_niosII_standard.ptf

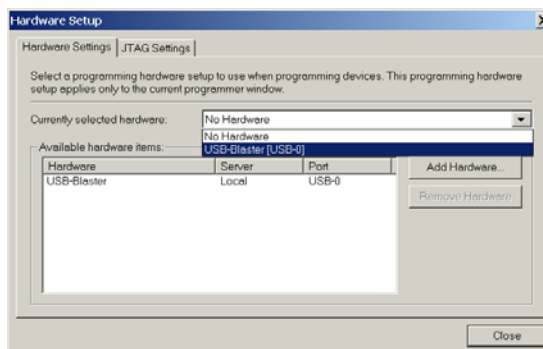
Download Hardware Design to Target FPGA

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 (i.e., 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 via the USB download cable.
2. Apply power to the board.
3. Start the Nios II IDE. On Windows computers, choose **All Programs > Start > Altera Nios II EDS <version> > Nios II IDE <version>** in the Windows Start menu.
4. After the welcome page appears, click **Workbench**.
5. Choose **Tools > Quartus II Programmer**.
6. Click **Auto Detect**. The device on your board (see [Table 1-1 on page 1-3](#)) should be detected automatically.
7. Click the top row to highlight it.
8. Click **Change File**.
9. Browse to the *<Nios II standard design directory>* directory shown in [Table 1-1 on page 1-3](#).
10. Select the programming file *<FPGA programming file>.sof* for your board as shown in [Table 1-1 on page 1-3](#).
11. Click **OK**.
12. Click **Hardware Setup** in the top, left corner of the Quartus II Programmer window. The **Hardware Setup** dialog box appears.
13. Select **USB-Blaster** from the **Currently selected hardware** drop-down list box. See [Figure 1-2](#).

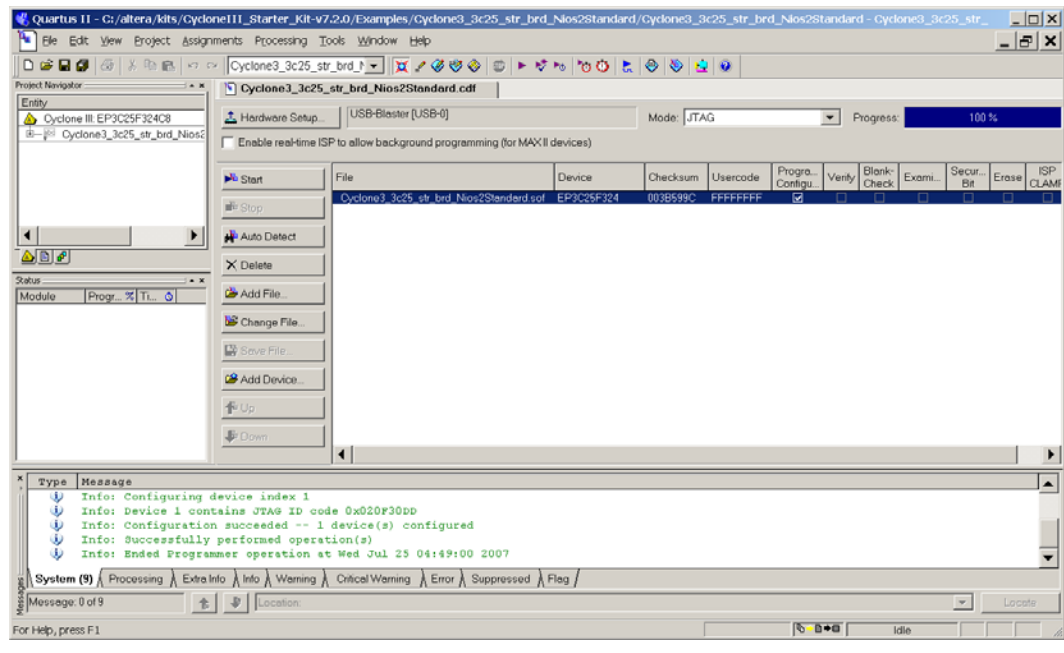


If the appropriate download cable does not appear in the list, you must first install drivers for the cable. Refer to Quartus II Help for information on how to install the driver.

Figure 1–2. Hardware Setup Window

14. Click **Close**.
15. Turn on the **Program/Configure** option for the programming file (see [Figure 1–3](#) for an example).
16. Click **Start**.

Figure 1–3. Quartus II Programmer



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 IDE Build Flow

The Nios II IDE build flow is an easy-to-use graphical user interface (GUI) that automates build and makefile management. The Nios II IDE integrates a text editor, debugger, the Nios II flash programmer, the Quartus II Programmer, and the Nios II C-to-Hardware (C2H) compiler GUI. The included example software application templates make it easy for new software programmers to get started quickly.

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



For a complete tutorial on using the Nios II IDE to develop programs, go to the software development tutorial, which is available in the IDE help.

Create the hello_world Example Project

In this section you will create a new Nios II C/C++ application project from an installed example. To begin, perform the following steps in the Nios II IDE:

1. Return to the Nios II IDE.



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.

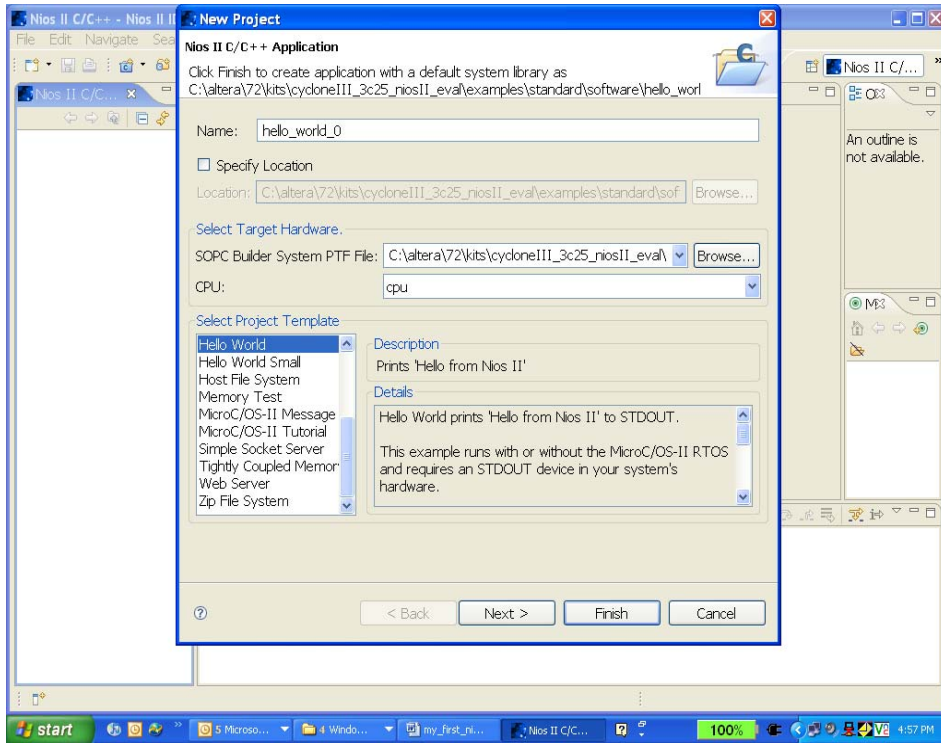
2. Choose **File > New > Nios II C/C++ Application** to open the **New Project** wizard.
3. In the **New Project** wizard, make sure you have:

- Selected the Hello World project template.
- Given the project a name (the default is **Hello_World_0**).
- Selected the target hardware system PTF file (you can browse to the location for the PTF file for your development board as shown in [Table 1-1 on page 1-3](#)).

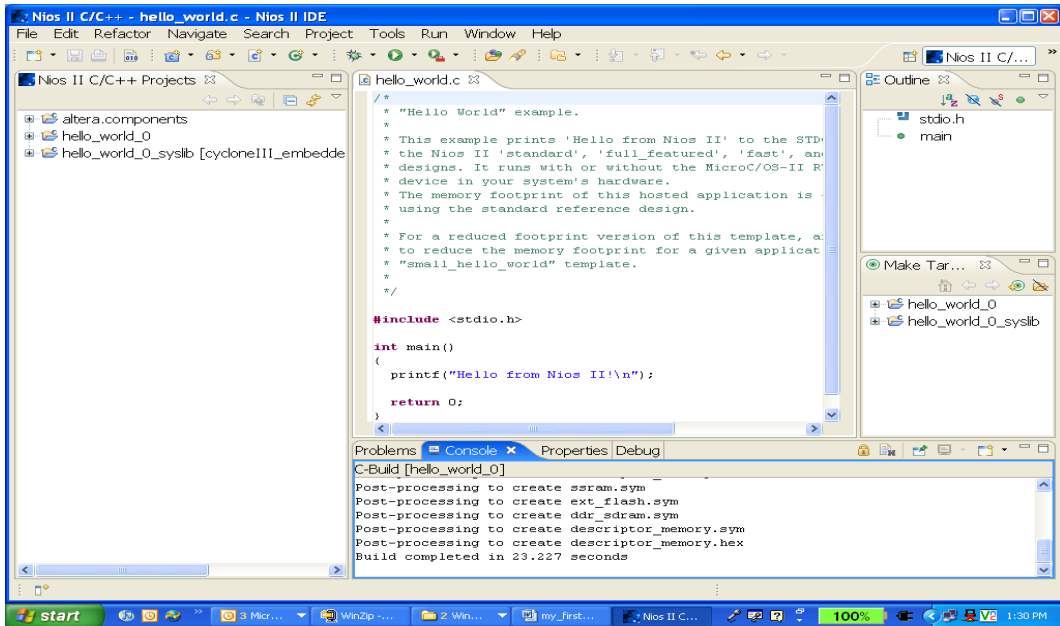


Every Nios II software project needs a system description of the corresponding Nios II hardware system. For the Nios II IDE, this system description is contained in a PTF file. See [Figure 1-4 on page 1-8](#) for an example.

Figure 1–4. Nios II IDE New Project Wizard



4. Click **Finish**. The Nios II IDE creates the **hello_world_0** project and returns to the Nios II C/C++ project perspective. See [Figure 1–5](#).

Figure 1–5. Nios II IDE C++ Project Perspective for `hello_world_0`

When you create a new project, the Nios II IDE creates two new projects in the **Nios II C/C++ Projects** tab:

- **hello_world_0** is your C/C++ application project. This project contains the source and header files for your application.
- **hello_world_0_syslib** is a system library that encapsulates the details of the Nios II system hardware.



When you build the system library for the first time the Nios II IDE automatically generates files useful for software development, including:

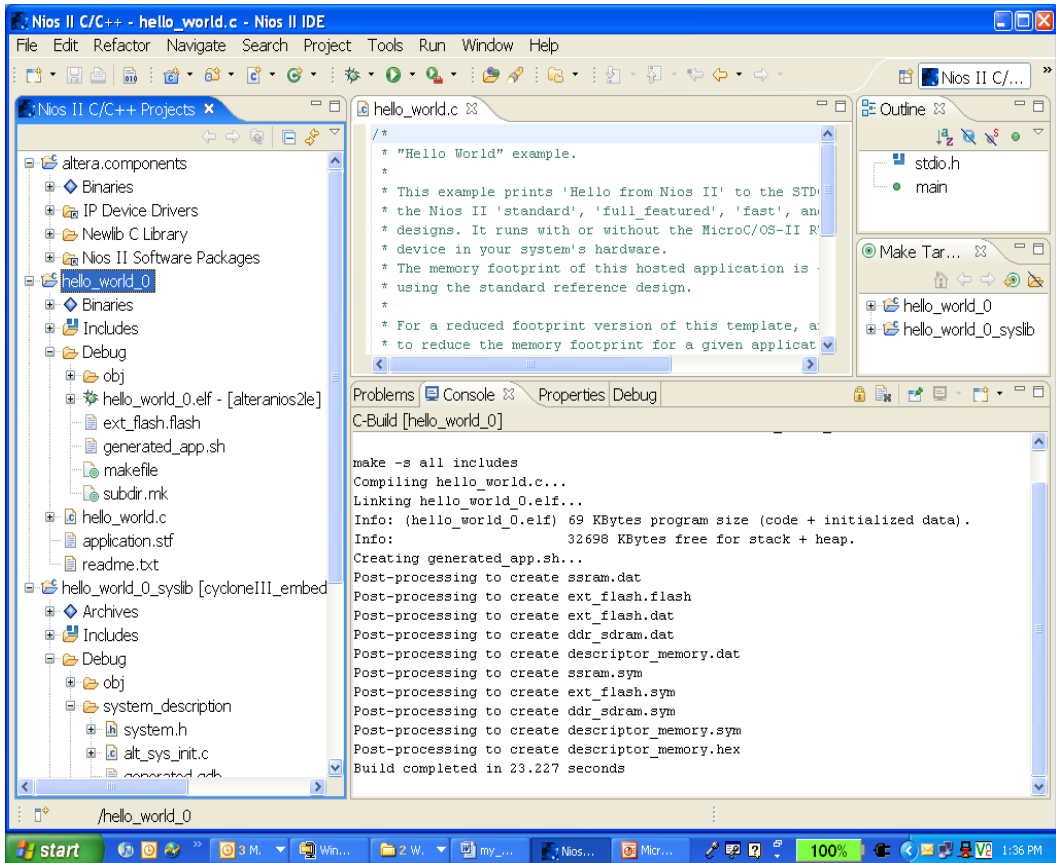
- IP device drivers, including SOPC component device drivers for the Nios II hardware system
- NewLib C library, which is a richly featured C library for the Nios II processor
- Nios II software packages
 - Nios II hardware abstraction layer (HAL)
 - NicheStack TCP/IP Network Stack, Nios II Edition
 - Nios II host file system

- Nios II read-only zip file system
- Micrium's μ C/OS-II realtime 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
- **Hello_world_0.elf**, which is an executable and linked format file for the application located in **hello_world_0** folder under **Debug**.

Build and Run the Program

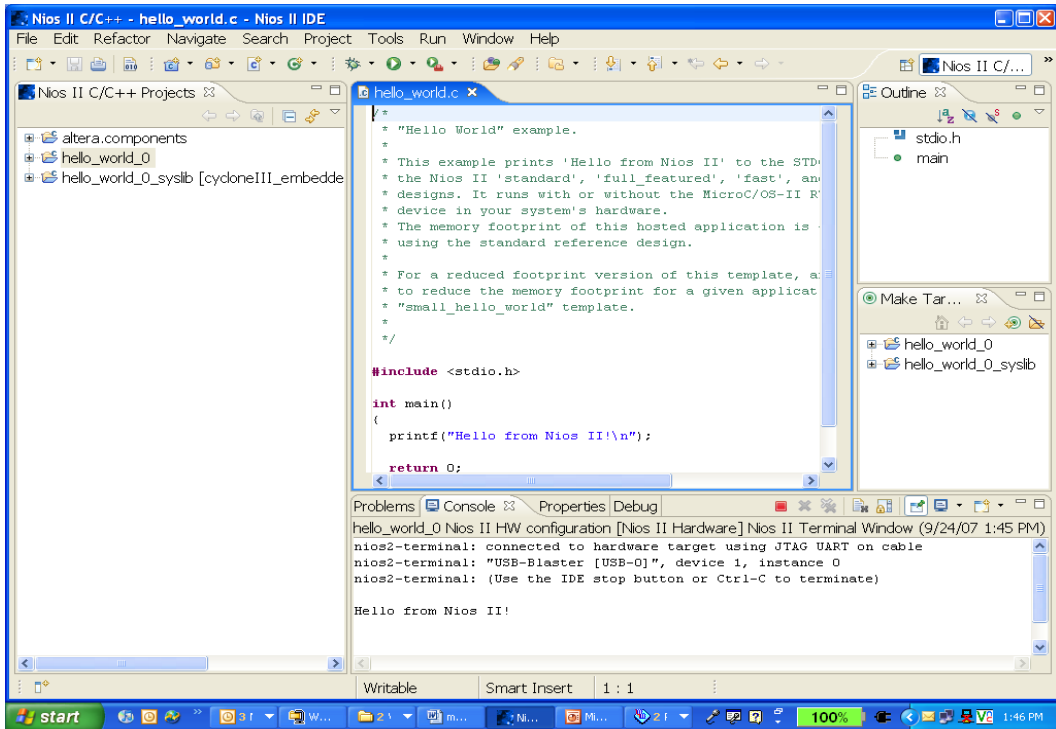
In this section you will build and run the program to execute the compiled code.

To build the program, right-click the **hello_world_0** project in the **Nios II C/C++ Projects** tab and choose **Build Project**. The **Build Project** dialog box appears and the IDE begins compiling the project. When compilation completes, the message "Build completed" appears in the **Console** tab. The completion time varies depending on your system. See [Figure 1-6](#) for an example.

Figure 1–6. Nios II IDE `hello_world_0` Build Completed

Right-click the `hello_world_0` project, choose **Run As**, and choose **Nios II Hardware**. The IDE downloads the program to the FPGA on the target board and begins execution. When the target hardware begins executing the program, the message “Hello from Nios II!” displays in the Nios II IDE Console tab. See Figure 1–7 for an example.

Figure 1–7. Hello_World_0 Program Output



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

Edit and Re-Run the Program

You can modify the `hello_world_0.c` program file in the IDE, build it, and re-run the program to observe your changes executing on the target board. In this section you will add code that makes LED1 blink.



For more information on how LED1 blinks refer to [“Why the LED Blinks”](#) on page 1–13.

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

1. In the **hello_world_0.c** file, add the text shown in blue in the example below :

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
    printf("Hello from Nios II!\n");
    int count = 0;
    int delay;
    while(1)
    {
        IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count & 0x01);
        delay = 0;
        while(delay < 2000000)
        {
            delay++;
        }
        count++;
    }
    return 0;
}
```

2. Save the project.
3. Recompile the file by right-clicking **hello_world_0** in the **Nios II C/C++ Projects** tab and choosing **Run > Run As > Nios II Hardware**.

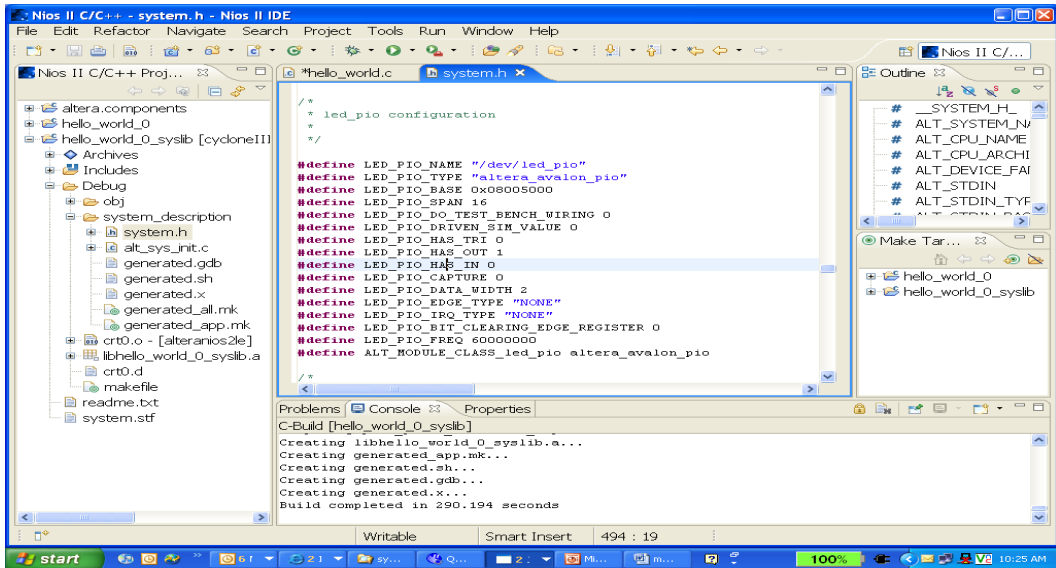


You do not need to build the project manually; the Nios II IDE automatically re-builds the program before downloading it to the FPGA.

4. Orient your development board so that you can observe LED 1 blinking.

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 is located in the in the **hello_world_0_syslib\Debug\system_description** directory as shown in [Figure 1-8](#).

Figure 1–8. *system.h* Location

If you look at the `system.h` file for the Nios II project example used in this tutorial, you will notice the `led_pio` function. This function controls the LED. The Nios II processor controls the PIO ports (and thereby 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 is located in `altera\<version number>\ip\sopc_builder_ip\altera_avalon_pio`.

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 peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to *Quartus II Version <version> Handbook Volume 5: Embedded Peripherals*.

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.

Debugging the Application

Before you can debug a project in the Nios II IDE, you must create a debug configuration that specifies how to run the software. To set up a debug configuration, perform the following steps:

1. To debug your application, right-click the application (**hello_world_0** by default) and choose **Debug as Nios II Hardware**.
2. Click **Apply**.
3. Click **Debug**.
4. If the **Confirm Perspective Switch** message box appears, click **Yes**.

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.

5. Choose **Run > Resume** to resume execution.

When debugging a project in the Nios II IDE, 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 C/C++ project perspective from the debug perspective, click the two arrows `>>` in the top right corner of the GUI.



For more information about debugging software projects in the Nios II IDE, refer to the "Nios II Integrated Development Environment" section in the *Nios II Software Developer's Handbook* or the Nios II IDE help.

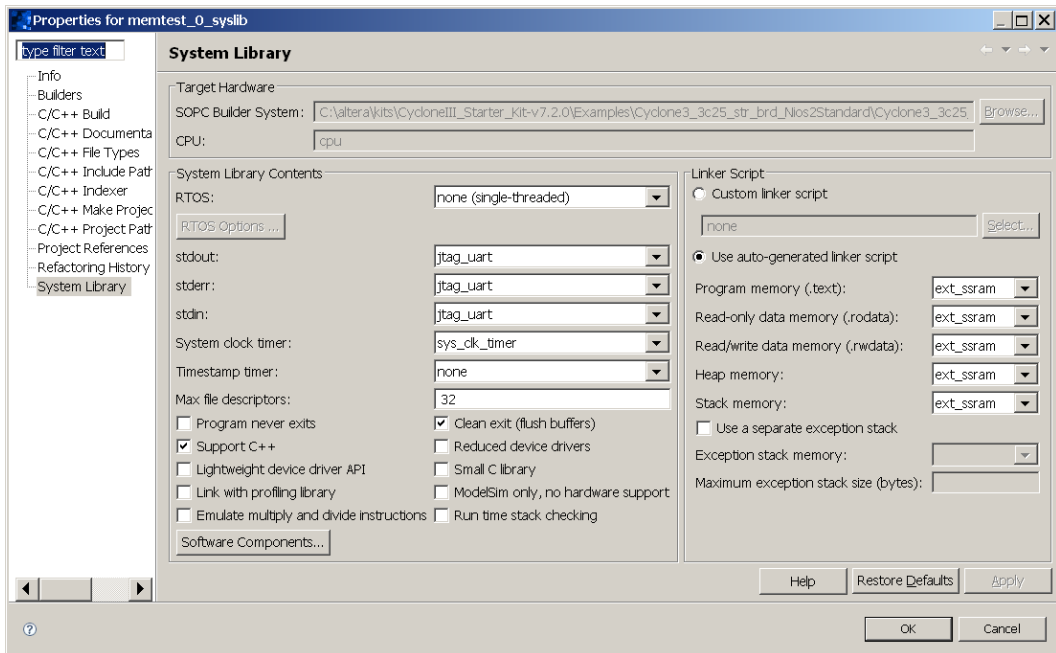
Configure System Library

A commonly asked question is how to change the target memory in which to run application code. In this section you will explore your software program settings using the **System Properties** dialog box. Perform the following steps:

1. In the Nios II IDE, right-click **hello_world_0** and choose **System Library Properties**. The **Properties for hello_world_0_syslib** dialog box opens.

2. Click **System Library**. The **System Library** page contains settings related to how the program interacts with the underlying hardware. The settings have names that correspond to the targeted Nios II hardware.
3. In the **Linker Script** box, observe which memory has been assigned for **Program memory (.text)**, **Read-only data memory (.rodata)**, **Read/write data memory (.rwdata)**, **Heap memory**, and **Stack memory**, see [Figure 1-9](#). These settings determine which memory is used to store the compiled executable program when the example `hello_world_0` program runs. You can also specify which interface you want to use for `stdio`, `stdin`, and `stderr`. You can also add and configure an RTOS for your application and configure build options to support C++, reduced device drivers, etc.
4. Choose `ext_ssram` for all of the memory options in the **Linker Script** box. See [Figure 1-9](#) for an example.

Figure 1-9. Configuring System Library Properties



5. Click **OK** to close the **Properties for memtest_0_syslib** dialog box and return to the IDE workbench.



If you make changes to the system properties you must rebuild your project. To rebuild, right-click the **hello_world_0** project in the **Nios II C/C++ Projects** tab and choose **Build Project**.

Next Steps

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

- *Developing Software for Nios II*—These short, online software tutorials walk you through the basics of developing software for the Nios II processor. You can access these tutorials from the Training link on the Embedded Processing web page at www.altera.com/embedded.
- *Nios II Software Developer's Handbook*—This handbook provides a complete reference on developing software for the Nios II processor.
- *Software Development Tutorial*—This tutorial teaches how to use the Nios II IDE to develop, run, and debug new Nios II C/C++ application projects. This tutorial is available in the Nios II IDE help.
- *Nios II IDE Help*—The Nios II IDE help provides complete reference on features of the IDE. To open the help, click **Help > Help Contents** and click the **Nios II IDE Help** book in the **Contents** pane.
- *Nios II Processor Reference Handbook*—This handbook provides a complete reference for the Nios II processor hardware.
- *Quartus II <version> Handbook Volume 5: Embedded Peripherals*—This volume contains details on the peripherals provided with the Nios II Embedded Design Suite.
- *Quartus II <version> Handbook Volume 4: SOPC Builder*—This volume provides a complete reference on using SOPC Builder, including building memory subsystems and creating custom components.

For a complete list of all documents available for the Nios II processor, visit the Nios II literature page at www.altera.com/nios2.

