

## Introduction

The Nios® II software build tools allow you to create Nios II software projects, with detailed control over the software build process.

This chapter introduces you to project creation with the software build tools at the command line.

## Creating Software Projects from the Command-Line

The Nios II software build tools allow you to construct a wide variety of complex software systems using simple command-line utilities. You can use scripts (or other tools) to combine command utilities in many useful ways.

Like the Nios II integrated development environment (IDE), the Nios II software build tools commands are available on both Windows and Linux operating systems.

The Nios II software build tools are designed to work in the Nios II command shell.



For details about the Nios II command shell, refer to “Altera-Provided Development Tools” in the *Overview* chapter of the *Nios II Software Developer’s Handbook*.

## Outline of the Nios II Software Build Tools

Before you use the software build tools, you should have a basic understanding of what they are.

### The Parts of the Software Build Tools

The Nios II software build tools consist of:

- Command-line utilities
- Command-line scripts
- Tcl commands
- Tcl scripts

These elements work together in a **bash** shell environment to create software projects, as described in the next section.

### What the Build Tools Create

The purpose of the build tools is to create and build Nios II software projects. A Nios II project is simply a makefile with associated source files.

The software build tools can create the following types of projects:

- Nios II application—A program implementing some desired functionality, such as control or signal processing.

- Nios II board support package (BSP)—A library providing access to hardware in the Nios II system, such as UARTs and other I/O devices. A BSP also includes the operating system, and other basic system software packages such as communications protocol stacks. A BSP provides a software runtime environment customized for one processor in an SOPC Builder system.
- User library—A library implementing a collection of reusable functions, such as graphics algorithms.

As a key part of creating a software project, the software build tools create a makefile for you. Nios II projects are sometimes called “user-managed,” because you, the user, are responsible for the content of the project makefile. You use the Nios II software build tools to control what goes in the makefile.

### Makefiles and the Software Build Tools

The makefile is the central component of a Nios II software project, whether the project is created through Nios II IDE or through the command line. The makefile describes all the components of a software project and how they are compiled and linked. With a makefile and a complete set of C/C++ source files, your Nios II software project is fully defined. No special project file is needed.

The Nios II build tools include utilities and scripts to create project makefiles. When you are starting out, it is easiest to use these utilities and scripts to create makefiles for you.



The *Using the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook* provides detailed information about creating makefiles.

As you become experienced with Nios II makefiles, you can modify an application or user library makefile that the build tools generate for you. With further experience, you might choose to create your application or user library makefile manually. Studying the autogenerated application makefiles, and experimenting with the makefile generation tools, can help you understand how to create and modify your own application and user library makefiles.



Altera does not recommend creating or modifying BSP makefiles by hand.

## Getting Started

The best way to learn about the Nios II software build tools is to use them. The following tutorial guides you through the process of creating, building, running, and debugging a “Hello World” program with a minimal number of steps. Later chapters provide more of the underlying details, allowing you to take more control of the process. The goal of this chapter is to show you how simple and straightforward the basic process is.

The Nios II software build tools include a number of scripts that demonstrate how to combine command utilities to obtain the results you need. This tutorial uses the `create-this-app` script as an example.

## What You Need

To complete this tutorial, you must have the following:

- Altera® Quartus® II development software, version 8.0 or later. The software must be installed on a Windows or Linux computer that meets the Quartus II minimum requirements.
- The Altera Nios II Embedded Design Suite (EDS), version 8.0 or later.
- A Nios development board.
- A download cable such as the Altera USB-Blaster™ cable.

## Running the Nios II Command-Line Software Build Tools

You run the Nios II command-line software build tools from the Nios II command shell.



For details about the Nios II command shell, refer to “Altera-Provided Development Tools” in the *Overview* chapter of the *Nios II Software Developer’s Handbook*.

## Creating hello\_world for a Nios Development Board

In this section you create a simple “Hello World” project. To create and build the `hello_world` example for a Nios development board, perform the following steps:

1. Start a Nios II command shell, as described in “Altera-Provided Development Tools” in the *Overview* chapter of the *Nios II Software Developer’s Handbook*.
2. Create a working directory for your hardware and software projects. The following steps refer to this directory as `<projects>`.
3. Change to the `<projects>` directory by typing the following command:

```
cd <projects>←
```

4. Locate a Nios II hardware example for your Nios development board. For example, if you have a Cyclone® II development board, you might select `<Nios II EDS install path>/examples/verilog/niosII_cycloneII_2c35/standard`.

This example uses the Verilog HDL standard hardware example design. You can select the language you prefer (Verilog HDL or VHDL), and any type of example design except `small`.

5. Copy the hardware example to your `<projects>` working directory, using a command such as the following:

```
cp -R $SOPC_KIT_NIOS2/examples/verilog/niosII_cycloneII_2c35/standard .←
```



`SOPC_KIT_NIOS2` is a predefined environment variable representing `<Nios II EDS install path>`.

6. Ensure that the working directory and all subdirectories are writable by typing the following command:

```
chmod -R +w .r
```

7. The `<projects>` directory contains a subdirectory named `software_examples/app/hello_world`. The following steps refer to this directory as `<application>`.

8. Change to the `<application>` directory by typing the following command:

```
cd <application>↵
```

9. Type the following command to create and build the application:

```
./create-this-app↵
```

The `create-this-app` script copies the application source code to the `<application>` directory, runs `nios2-app-generate-makefile` to create a makefile (named **Makefile**), and then runs `make` to create an executable and linking format (**.elf**) file. The `create-this-app` script finds a compatible BSP by looking in `<projects>/software_examples/bsp`. In the case of `hello_world`, it selects the `hal_default` BSP.

To create the example BSP, `create-this-app` calls the `create-this-bsp` script in the BSP directory.

## Running `hello_world` on a Nios Development Board

To run the `hello_world` example on a Nios development board, perform the following steps:


1. Start a Nios II command shell, as described in “Altera-Provided Development Tools” in the *Overview* chapter of the *Nios II Software Developer’s Handbook*.
2. Download the SRAM object (**.sof**) file for the Quartus II project to the Nios development board. This step configures the FPGA on the development board with your project’s associated SOPC Builder system.

The **.sof** file resides in `<projects>`, along with your Quartus II project (**.qpf**) file. You download it by changing to the `<projects>` directory, then running `nios2-configure-sof` by typing the following commands:

```
cd <projects>↵  
nios2-configure-sof↵
```

The board is configured, and ready to run the project’s executable code.

The `nios2-configure-sof` utility runs the Quartus II Programmer to download the **.sof** file. You can also run the `quartus_pgm` command directly.

 For more information about programming the hardware, refer to the *Nios II Hardware Development Tutorial*.

3. Start another command shell. If practical, make both command shells visible on your desktop.
4. In the second command shell, run the Nios II terminal application to connect to the Nios development board through the JTAG UART port by typing the following command:

```
nios2-terminal↵
```

5. Return to the original command shell, and ensure that `<projects>/software_examples/app/hello_world` is the current working directory.

6. Download and run the `hello_world` executable program as follows:

```
nios2-download -g hello_world.elf
```

The following output appears in the second command shell:

```
Hello from Nios II!
```

## Debugging `hello_world`

An integrated development environment is the most powerful environment for debugging a software project. You debug a command-line project by importing it to the Nios II IDE. After you import the project, the IDE uses your makefiles to build the project. This two-step process combines the advantages of the command-line software build tools development flow with the convenience of a GUI debugger.

This section discusses the process of importing and debugging the `hello_world` application.

When debugging a project in the Nios II IDE, you can also pause, stop, and single-step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

## Next Steps

In the previous section, you create, build, run, and debug a sample program. The next section, “[Creating a Script](#)”, shows you how to create your own project script.



For detailed information about the using Nios II software build tools, refer to the *Using the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*.

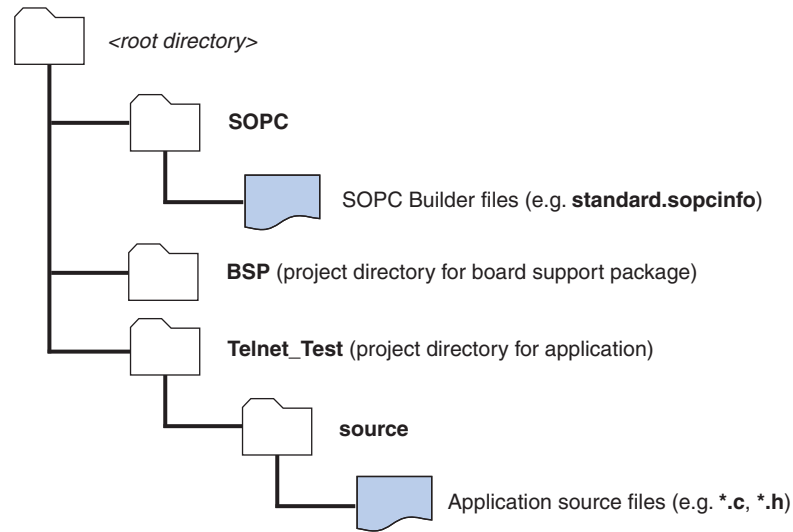
## Creating a Script

In simple cases, you can create a complete Nios II software application by running Nios II software build tools utilities from the command line. More commonly, you might need to create some simple scripts. You can base your scripts on example scripts supplied with the Nios II EDS, or develop them independently.

## Scripting Basics

This section provides an example to teach you how you can create a software application using a script.

Suppose you want to build a software application for a Nios II system that features the `lan91c111` component and supports the NicheStack<sup>®</sup> TCP/IP stack. Furthermore, suppose that you have organized the hardware design files and the software source files as shown in [Figure 3-1](#).

**Figure 3-1.** Simple Software Project Directory Structure

One easy method for creating a BSP is to use the `nios2-bsp` script. The script in [Example 3-1](#) creates a BSP and then builds it.

**Example 3-1.** `nios2-bsp`

```
nios2-bsp ucosii . ../SOPC/ --cmd enable_sw_package altera_iniche \  
--set altera_iniche.iniche_default_if lan91c111  
make
```

[Table 3-1](#) shows the meaning of each argument to the `nios2-bsp` script in [Example 3-1](#).

**Table 3-1.** `nios2-bsp` Example Arguments

Argument	Purpose
<code>ucosii</code>	Sets the BSP type to MicroC/OS-II
<code>.</code>	Specifies the directory in which the BSP is to be created
<code>../SOPC/</code>	Points to the location of the SOPC Builder system
<code>--cmd enable_sw_package altera_iniche</code>	Adds the NicheStack TCP/IP stack software package to the BSP
<code>--set altera_iniche.iniche_default_if lan91c111</code>	Specifies the default hardware interface for the NicheStack TCP/IP stack

You use `nios2-app-generate-makefile` to create application projects. The script in [Example 3-2](#) creates an application project and builds it.

**Example 3-2.** nios2-app-generate-makefile

```
nios2-app-generate-makefile --bsp-dir ../BSP \
--elf-name telnet-test.elf --src-dir source/
make
```

Table 3-2 shows the meaning of each argument in Example 3-2.

**Table 3-2.** nios2-app-generate-makefile Example Arguments

Argument	Purpose
--bsp-dir ../BSP	Specifies the location of the BSP on which this application is based
--elf-name telnet-test.elf	Specifies the name of the executable file
--src-dir source/	Tells nios2-app-generate-makefile where to find the C source files

These simple scripts are all you need to create and build your application.

## Nios II Scripting Examples

The Nios II EDS includes many hardware and software examples based on the Nios II processor. These include hardware designs that you can download to Nios development boards, and software examples that run on these designs. The examples can be very helpful as you start the development of your custom design. They provide a stable starting point for exploring design options. Also, they demonstrate many commonly used features of the Nios II EDS.

The Nios II software examples include scripts to create and build the software projects using the Nios II software build tools. These scripts do everything necessary to create a BSP and an application project for each software example. You can copy and modify these scripts to create your custom software design.

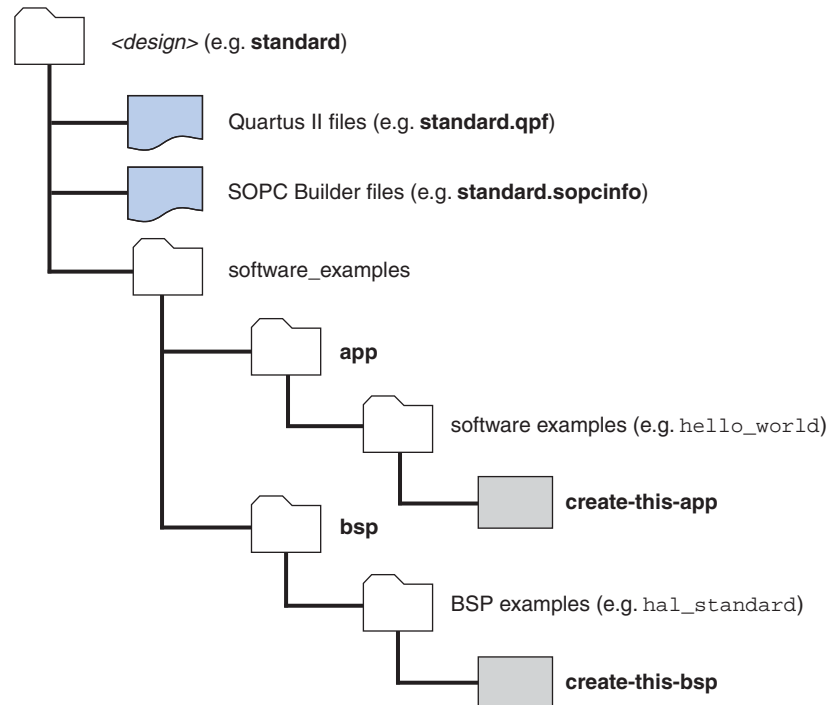
The hardware examples for each Nios II development board reside in the following location:


*<Nios II EDS install path>/examples/<language>/<board>*

*<language>* is either **vhdl** or **verilog** and *<board>* is the name of the development board. For example, the standard Verilog HDL example design for the Nios II 1S40 development board resides in the following location:

*<Nios II EDS install path>/examples/verilog/niosII\_stratix\_1s40/standard*

Figure 3-2 shows the directory structure under each hardware example design. There are multiple software examples and BSP examples, each with its own directory. Each software example directory contains a `create-this-app` script and each BSP example directory contains a `create-this-bsp` script. These scripts create software projects, as demonstrated in “Creating hello\_world for a Nios Development Board” on page 3-3.

**Figure 3-2.** Software Example Design Directory Structure

 For more detail about the software example scripts, refer to the *Using the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*.

## Using the Nios II BSP Editor

You can also create or modify a Nios II BSP project with the Nios II BSP editor, a standalone GUI. The BSP editor provides a GUI for editing BSPs. This section briefly describes the BSP editor.

You can create a BSP with the Nios II BSP editor. To start the BSP editor, type the following command in the Nios II command shell:

```
nios2-bsp-editor↵
```

The BSP editor enables you to edit settings, linker regions, and section mappings, and to select software packages and device drivers.

To create a default BSP, on the File menu, click **New BSP**. After the BSP is created, you can modify the settings using the GUI.

The capabilities of the BSP editor constitute a large subset of the capabilities of the `nios2-bsp-create-settings`, `nios2-bsp-update-settings`, and `nios2-bsp-generate-files` utilities. Any project created in the BSP editor can also be created using the command-line utilities.

The BSP editor prompts you to select the following items:

- A location for your BSP
- An SOPC Builder system (`.sopcinfo`) file

You can also select an operating system, and with a multiprocessor Nios II system you can select a CPU name. Click **Next** to create the BSP settings file.

After the BSP settings file is created, the BSP editor allows you to modify default settings.



For details about how to use the BSP editor, start the tool and refer to the onscreen explanatory text. For more information about creating BSPs, refer to the *Introduction to the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*.

## Referenced Documents

This chapter references the following documents:

- *Overview* chapter of the *Nios II Software Developer's Handbook*
- *Introduction to the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*
- *Using the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*
- *Nios II Hardware Development Tutorial*

## Document Revision History

Table 3-3 shows the revision history for this document.

**Table 3-3.** Document Revision History

Date & Document Version	Changes Made	Summary of Changes
March 2009 v9.0.0	<ul style="list-style-type: none"> <li>■ Described BSP editor.</li> <li>■ Reorganized and updated information and terminology to clarify role of Nios II software build tools.</li> <li>■ Corrected minor typographical errors.</li> </ul>	BSP Editor
May 2008 v8.0.0	No change from previous release.	
October 2007 v7.2.0	Repurpose this chapter as a “getting started” guide. Move descriptive and reference material to separate chapters.	Additional “getting started” material. Descriptive and reference material in separate chapters.
May 2007 v7.1.0	Initial Release.	—

