The Nios® II Software Build Tools (SBT) allows you to construct a wide variety of complex embedded software systems using a command-line interface. From this interface, you can execute Software Built Tools command utilities, and use scripts (or other tools) to combine the command utilities in many useful ways.

This chapter introduces you to project creation with the SBT at the command line.

This chapter includes the following sections:

- "Advantages of Command-Line Software Development"
- "Outline of the Nios II SBT Command-Line Interface"
- "Getting Started in the SBT Command Line"
- "Software Build Tools Scripting Basics" on page 3–7
- "Running make" on page 3–10

Advantages of Command-Line Software Development

The Nios II SBT command line offers the following advantages over the Nios II SBT for Eclipse™:

- You can invoke the command line tools from custom scripts or other tools that you might already use in your development flow.
- On a command line, you can run several Tcl scripts to control the creation of a board support package (BSP).
- You can use command line tools in a bash script to build several projects at once.

The Nios II SBT command-line interface is designed to work in the Nios II Command Shell.

For details about the Nios II Command Shell, refer to "The Nios II Command Shell" on page 3–2.

Outline of the Nios II SBT Command-Line Interface

The Nios II SBT command-line interface consists of:

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

These elements work together in the Nios II Command Shell to create software projects.

Utilities

The Nios II SBT command-line utilities enable you to create software projects. You can call these utilities from the command line or from a scripting language of your choice (such as perl or bash). On Windows, these utilities have a **.exe** extension. The Nios II SBT resides in the *<Nios II EDS install path>*/**sdk2/bin** directory.

Refer to "Altera-Provided Development Tools" in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook* for a summary of the command-line utilities provided by the Nios II SBT.

Scripts

Nios II SBT scripts implement complex behavior that extends the capabilities provided by the utilities.

Table 3–1 summarizes the scripts provided with the Nios II SBT.

Table 3–1. Nios II SBT Scripts

Command	Summary
nios2-bsp	Creates or updates a BSP
create-this-app *(1)*	Creates a software example and builds it
create-this-bsp *(1)*	Creates a BSP for a specific hardware design example and builds it

Note to Table 3–1:

(1) There are **create-this-app** scripts for each software example and several **create-this-bsp** scripts for each hardware design example. For more details, refer to "Nios II Design Example Scripts" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook*.

Tcl Commands

Tcl commands are a crucial component of the Nios II SBT. Tcl commands allow you to exercise detailed control over BSP generation, as well as to define drivers and software packages.

Tcl Scripts

The SBT provides powerful Tcl scripting capabilities. In a Tcl script, you can query project settings, specify project settings conditionally, and incorporate the software project creation process in a scripted software development flow. The SBT uses Tcl scripting to customize your BSP according to your hardware and the settings you select. You can also write custom Tcl scripts for detailed control over the BSP.

The Nios II Command Shell

The Nios II Command Shell is a bash command-line environment initialized with the correct settings to run Nios II command-line tools. The Command Shell supports the GCC toolchain.

For general information about GCC toolchains, refer to "Altera-Provided Development Tools" in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*

Starting the Nios II Command Shell

To open the Nios II Command Shell, perform the following steps, depending on your environment:

■ In the Windows operating system, on the Start menu, point to **Programs** > **Altera** > **Nios II EDS** *<version>*, and click **Nios II** *<version>* **Command Shell**:.

■ In the Linux operating system, in a command shell, change directories to *<Nios II EDS install path>*, and type the command `nios2_command_shell.sh`.

Auto-Executing a Command in the Nios II Command Shell

In certain situations, you might need to run a command or a script automatically after the Nios II Command Shell is initialized. When you start the Nios II Command Shell environment, to automatically execute a command perform one of the following steps, depending on your environment:

■ In the Windows operating system, execute the following command:

   `"<Nios II EDS install path>/Nios II Command Shell.bat" <command>`↵

■ In the Linux operating system, execute the following command:

   `<Nios II EDS install path>/nios2_command_shell.sh <command>`↵

For example, in Windows, to run an automated build, you might execute the following command:

`"<Nios II EDS install path>/Nios II Command Shell.bat" custom_build.sh`↵

The Nios II Command Shell startup script (`Nios II Command Shell.bat` or `nios2_command_shell.sh`) makes no special assumptions about its initial environment. You can use the Nios II Command Shell with auto-execution from any environment that accepts commands native to your host operating system. For example, in Linux you can use **crontab** to schedule a job to run in the Nios II Command Shell at a later time.

Getting Started in the SBT Command Line

Using the Nios II SBT on the command line is the best way to learn about it. 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 that the basic process is simple and straightforward.

The Nios II SBT includes a number of scripts that demonstrate how to combine command utilities to obtain the results you need. This tutorial uses a **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.

■ An Altera development board.

■ A download cable such as the Altera USB-Blaster™ cable.

You run the Nios II SBT commands from the Nios II Command Shell.

For details about the Nios II Command Shell, refer to "The Nios II Command Shell".

Creating hello_world for an Altera Development Board

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

1. Start the Nios II Command Shell, as described in "The Nios II Command Shell".

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 Altera development board. For example, if you have a Stratix® IV GX FPGA Development Kit, you might select *<Nios II EDS install path>***/examples/verilog/niosII_stratixIV_4sgx230/ triple_speed_ethernet_design**.

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

`cp -R /altera/100/nios2eds/examples/verilog/niosII_stratixIV_4sgx230/triple_speed_ethernet_design .`↵

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 File (**.elf**). 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 an Altera Development Board

To run the `hello_world` example on an Altera development board, perform the following steps:

1. Start the Nios II Command Shell.

2. Download the SRAM Object File (**.sof**) for the Quartus II project to the Altera 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 File (**.qpf**). You download it 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 Altera 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 SBT for Eclipse. After you import the project, Eclipse uses your makefiles to build the project. This two-step process combines the advantages of the SBT command line development flow with the convenience of a GUI debugger.

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

Import the hello_world Application

To import the **hello_world** application, perform the following steps:

1. Launch the Nios II SBT for Eclipse.

2. On the File menu, click **Import**. The **Import** dialog box appears.

3. Expand the **Nios II Project** folder, and select **Import Nios II project**.

4. Click **Next**. The **File Import** wizard appears.

5. Click **Browse** and navigate to the *<application>* directory, containing the **hello_world** application project.

6. Click **OK**. The wizard fills in the project path.

7. Type the project name `hello_world` in the **Project name** box.

8. Click **Finish**. The wizard imports the application project.

☞ If you want to view the BSP source files while debugging, you also need to import the BSP project into the Nios II SBT for Eclipse.

For a description of importing BSPs into Eclipse, refer to "Importing a Command-Line Project" in the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook.*

Download Executable Code and Start the Debugger

To debug the software project, perform the following steps:

1. Right-click the `hello_world` project, point to **Debug As**, and click **Nios II Hardware**.

2. If the **Confirm Perspective Switch** dialog box appears, click **Yes**.

   After a moment, you see the `main()` function in the editor. There is a blue arrow next to the first line of code, indicating that execution is stopped on this line.

   When targeting Nios II hardware, the **Debug As** command does the following tasks:

   ■ Creates a default debug configuration for the target board.

   ■ Establishes communication with the target board

   ■ Optionally verifies that the expected SOPC Builder system is configured in the FPGA.

   ■ Downloads the **.elf** file to memory on the target board.

   a. Sets a breakpoint at `main()`.

   ■ Instructs the Nios II processor to begin executing the code.

3. In the Run menu, click **Resume** to resume execution. You can also resume execution by pressing **F8**.

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

👣 For more detailed information about debugging projects in the Nios II SBT for Eclipse, refer to "Importing a Command-Line Project" and "Getting Started with Eclipse" in the *Getting Started with the Graphical User Interface* chapter of the *Nios II Software Developer's Handbook.*

Software Build Tools Scripting Basics

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

In this section, assume that you want to build a software application for a Nios II system that features the **lan91c111** component and supports the NicheStack® TCP/IP stack. Furthermore, assume 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
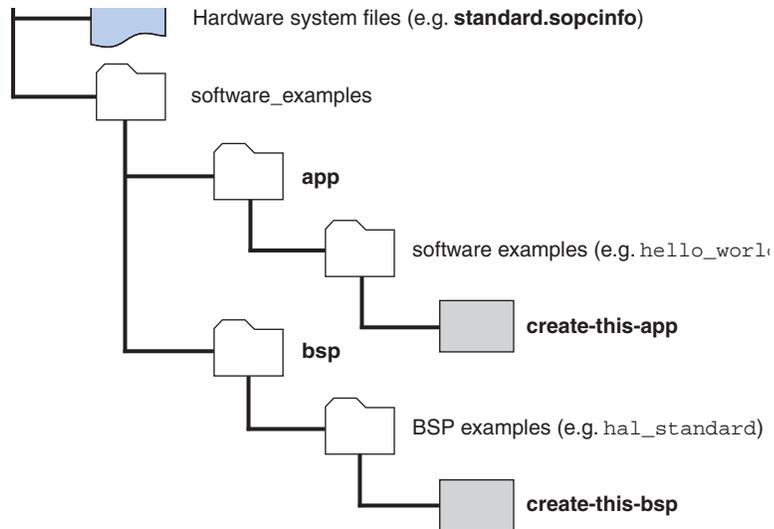
Creating a BSP with a Script

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 3–2 shows the meaning of each argument to the **nios2-bsp** script in Example 3–1.

For additional information about the **nios2-bsp** command, refer to "Nios II Software Build Tools Utilities" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook*.

Table 3–2. nios2-bsp Example Arguments

Argument	Purpose	Further Information
`ucosii`	Sets the operating system to MicroC/OS-II	"Settings Managed by the Software Build Tools" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook*
`.`	Specifies the directory in which the BSP is to be created	—
`../SOPC/`	Points to the location of the hardware project	—
`--cmd enable_sw_package altera_iniche`	Adds the NicheStack TCP/IP stack software package to the BSP	"Settings Managed by the Software Build Tools" and "Software Build Tools Tcl Commands" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook*
`--set altera_iniche.iniche_default_if lan91c111`	Specifies the default hardware interface for the NicheStack TCP/IP Stack - Nios II Edition	"Settings Managed by the Software Build Tools" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook*

Figure 3–2 shows the flow to create a BSP using the **nios2-bsp** script. The **nios2-bsp** script uses the **.sopcinfo** file to create the BSP files. You can override default settings chosen by **nios2-bsp** by supplying command-line arguments, Tcl scripts, or both.

Figure 3–2. nios2-bsp Command Flow
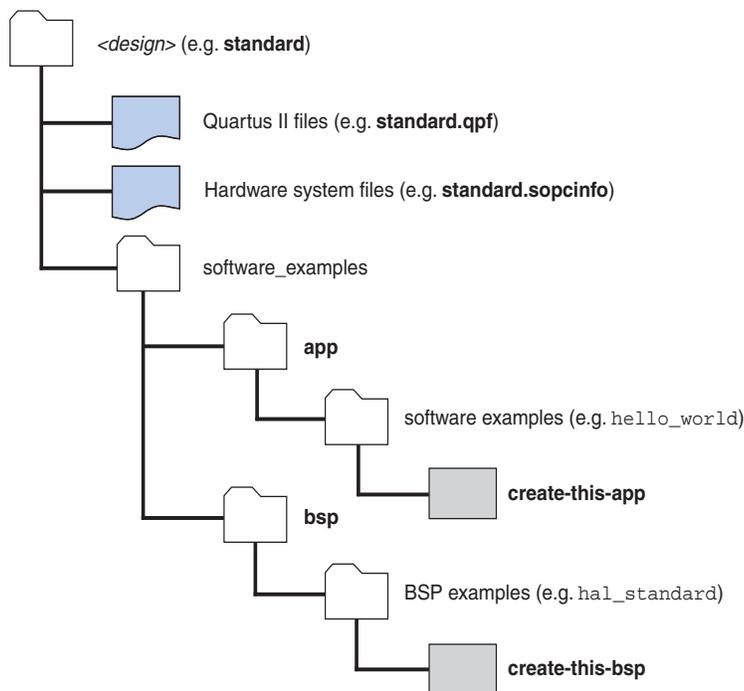
Creating an Application Project with a Script

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 3–3 shows the meaning of each argument in Example 3–2.

Table 3–3. 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

> For further information about each command argument in Table 3–3, refer to "Nios II Software Build Tools Utilities" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook.* For more detail about the software example scripts, refer to "Nios II Design Example Scripts" in the *Nios II Software Build Tools Reference* chapter of the *Nios II Software Developer's Handbook.*

Running make

nios2-bsp places all BSP files in the BSP directory, specified on the command line with argument `--bsp-dir`. After running **nios2-bsp**, you run **make**, which compiles the source code. The result of compilation is the BSP library file, also in the BSP directory. The BSP is ready to be linked with your application.

You can specify multiple targets on a **make** command line. For example, the following command removes existing object files in the current project directory, builds the project, downloads the project to a board, and runs it:

```
make clean download-elf↵
```

You can modify an application or user library makefile with the **nios2-lib-update-makefile** and **nios2-app-update-makefile** utilities. With these utilities, you can execute the following tasks:

- Add source files to a project

- Remove source files from a project

- Add compiler options to a project's make rules

- Modify or remove compiler options in a project's make rules

Creating Memory Initialization Files

To create memory initialization files for a Nios II system, you can use the Nios II Command Shell. Change to the software application folder, and type:

```
make mem_init_generate↵
```

This command creates the memory initialization and simulation files for all memory devices. It also generates a Quartus II IP File (**.qip**). The **.qip** file tells the Quartus II software where to find the initialization files. Add the **.qip** file to your Quartus II project.

Document Revision History

Table 3–4 shows the revision history for this document.

Table 3–4. Document Revision History

Date	Version	Changes
January 2014	13.1.0	■ Updated GCC4 toolchain from 4.1.2 to GCC 4.7.3. ■ Removed references to the Nios II IDE. ■ Removed references to GCC 3. ■ Removed the "Using the Nios II C2H Compiler" section.
May 2011	11.0.0	■ Can auto-execute a Command in the Nios II Command Shell ■ The GCC 3 toolchain is an optional feature
February 2011	10.1.0	■ Do not mix versions of GCC. ■ Removed "Referenced Documents" section.
July 2010	10.0.0	■ Introduction of GCC 4. ■ Discuss usage of GCC 3 and GCC 4 command shells.
November 2009	9.1.0	■ Repurpose and retitle this chapter as an introduction to Nios II Software Build Tools command-line usage. ■ Information about the BSP Editor moved to the *Getting Started with the Graphical User Interface* chapter.
March 2009	9.0.0	■ Describe BSP Editor. ■ Reorganize and update information and terminology to clarify role of Nios II Software Build Tools. ■ Correct minor typographical errors.
May 2008	8.1.0	Maintenance release.
October 2007	7.2.0	Repurpose this chapter as a "getting started" guide. Move descriptive and reference material to separate chapters.
May 2007	7.1.0	Initial Release.