

Introduction

This chapter provides the software developer with a high-level overview of the software development environment for the Nios® II processor. This chapter introduces the Nios II software development environment, the Nios II embedded design suite (EDS) tools available to you, and the process for developing software. This chapter contains the following sections:

- “What’s New” on page 1-1
- “Getting Started” on page 1-2
- “Nios II Software Development Environment” on page 1-3
- “Nios II Programs” on page 1-3
- “Design Flows for Creating Nios II Programs” on page 1-6
- “Additional EDS Support” on page 1-8
- “Third-Party Support” on page 1-9
- “Migrating from the First-Generation Nios Processor” on page 1-9
- “Further Nios II Information” on page 1-9

What’s New

The following features are new in the Nios II EDS v8.0:

- **.sopcinfo** file—Starting in v8.0, the SOPC Builder system file is encoded as a **.sopcinfo** file, rather than as a **.sopc** file. The **.sopcinfo** file is generated when you perform system generation in SOPC Builder. The **.sopcinfo** file is static and describes how your system is generated, based on IP components installed at generation time. Utilities that formerly accepted a **.sopc** file as input, such as **nios2-bsp-create-settings**, accept a **.sopcinfo** file starting in v8.0.

The `--sopc` command line argument supports the **.sopcinfo** file, rather than the **.sopc** file.



If your hardware was generated with SOPB Builder release 7.2 or earlier, regenerate it with SOPC Builder release 8.0 before creating a BSP.

BSP editor—In addition to the command-line Nios II software build tools, another way to create a BSP project is with Nios II BSP editor, a standalone graphical user interface (GUI). The BSP editor provides a graphical front end that drives the software build tools. To launch the Nios II BSP editor, you use the following command:

```
nios2-bsp-editor
```



For details about how to use the BSP editor, launch the tool and refer to the onscreen explanatory text.

- Software examples distributed on the Web—Certain software examples, distributed with the Nios II IDE through v7.2, are distributed on the Web starting in v8.0. The Web-available software examples are as follows:

Example Name	Location
Zip File System	Nios Community Wiki
MicroC/OS-II Message Box	Nios Community Wiki
Hello LED	Nios Community Wiki
Host File System	Nios Community Wiki
Dhrystone	www.altera.com
MicroC/OS-II Tutorial	www.altera.com
Tightly Coupled Memory	www.altera.com
Custom Instruction Tutorial	www.altera.com



The Nios II processor core includes an optional memory management unit (MMU). The Nios II hardware abstraction layer (HAL) does not support the MMU.

Getting Started

Writing software for the Nios II processor is similar to the software development process for any other microcontroller family. The easiest way to start designing effectively is to purchase a development kit from Altera® that includes documentation, a ready-made evaluation board, and all the development tools necessary to write Nios II programs.

The *Nios II Software Developer's Handbook* assumes you have a basic familiarity with embedded processor concepts. You do not need to be familiar with any specific Altera technology or with Altera development tools. Familiarity with Altera hardware development tools can give you a deeper understanding of the reasoning behind the Nios II software development environment. However, software developers can develop and debug applications without further knowledge of Altera technology.

Modifying existing code is perhaps the most common and comfortable way that software designers learn to write programs in a new environment. The Nios II EDS provides many example software designs that you can examine, modify, and use in your own programs. The provided examples range from a simple “Hello world” program, to a working real-time operating system (RTOS) example, to a full transmission control protocol/Internet protocol (TCP/IP) stack running a web server. Each example is documented and ready to compile.

Nios II Software Development Environment

The Nios II EDS provides a consistent software development environment that works for all Nios II processor systems. With a PC, an Altera FPGA, and a Joint Test Action Group (JTAG) download cable (such as an Altera USB-Blaster™ download cable), you can write programs for, and communicate with, any Nios II processor system. The Nios II processor’s JTAG debug module provides a single, consistent method to communicate with the processor using a JTAG download cable. Accessing the processor is the same, regardless of whether a device implements only a Nios II processor system, or whether the Nios II processor is embedded deeply in a complex multiprocessor system. Therefore, you do not need to spend time manually creating interface mechanisms for the embedded processor.

The Nios II EDS provides two distinct design flows and includes many proprietary and open-source tools (such as the GNU C/C++ tool chain) for creating Nios II programs. The Nios II EDS automates board support package (BSP) creation for Nios II-based systems, eliminating the need to spend time manually creating BSPs. Altera BSPs contain the Altera hardware abstraction layer (HAL), an optional RTOS, and device drivers. The BSP provides a C/C++ runtime environment, insulating you from the hardware in your embedded system.

Nios II Programs

Each Nios II program you develop in either Nios II EDS design flow consists of an application project, optional library projects, and a BSP project. You build your Nios II program into an Executable And Linked Format File (.elf) which runs on a Nios II processor. While terminology sometimes differs between the two design flows, the Nios II programs you develop are conceptually equal.

The following sections describe the project types that comprise a Nios II program:

Application Project

A Nios II C/C++ application project consists of a collection of source code combined into one executable (`.elf`) file. A typical characteristic of an application is that one of the source files contains function `main()`. An application includes code that calls functions in libraries and BSPs.

Library Project

A library project is a collection of source code contained within a single library archive (`.a`) file. Libraries often contain reusable, general purpose functions that multiple application projects can share. A collection of common arithmetical functions is one example. A library does not contain a function `main()`.

BSP Project

A Nios II BSP project is a specialized library containing system-specific support code. A BSP provides a software runtime environment customized for one processor in an SOPC Builder system. The Nios II EDS provides tools to modify settings that control the behavior of the BSP.



The Nios II integrated development environment (IDE) and the Nios II IDE design flow documentation use the term "system library" when referring to a BSP.

A BSP contains the following elements:

- Hardware Abstraction Layer (HAL)
- Newlib C Standard Library
- Device Drivers
- Optional Software Packages
- Optional Real-Time Operating System (RTOS)

Hardware Abstraction Layer (HAL)

The HAL provides a non-threaded, UNIX-like, C/C++ runtime environment. The HAL provides generic I/O devices, allowing you to write programs that access hardware using the newlib C standard library routines, such as `printf()`. The HAL minimizes (or eliminates) the need to access hardware registers directly to control and communicate with peripherals.



For complete details about the HAL, refer to the *The Hardware Abstraction Layer* section and the *HAL API Reference* chapter of the *Nios II Software Developer's Handbook*.

Newlib C Standard Library

Newlib is an open source implementation of the C standard library intended for use on embedded systems. It is a collection of common routines such as `printf()`, `malloc()`, and `open()`.

Device Drivers

Each device driver manages a hardware component. By default, the HAL instantiates a device driver for each component in your SOPC Builder system that needs a device driver. In the Nios II software development environment, a device driver has the following properties:

- A device driver is associated with a specific SOPC Builder component.
- A device driver might have settings that impact its compilation. These settings become part of the BSP settings.

Optional Software Packages

A software package is source code that you can optionally add to a BSP project to provide additional functionality. The NicheStack® TCP/IP - Nios II Edition is an example of a software package.



The Nios II IDE and the Nios II IDE design flow documentation use the term "software component" when referring to a software package.

In the Nios II software development environment, a software package typically has the following properties:

- A software package is not associated with specific hardware.
- A software package might have settings that impact its compilation. These settings become part of the BSP settings.



In the Nios II software development environment, a software package is distinct from a library project. A software package is part of the BSP project, not a separate library project.

Optional Real-Time Operating System (RTOS)

The Nios II EDS includes an implementation of the third-party MicroC/OS-II RTOS that you can optionally include in your BSP. MicroC/OS-II is built on the HAL, and implements a simple, well-documented RTOS scheduler. You can modify settings that become part of the BSP settings. Other operating systems are available from third-party vendors.

Design Flows for Creating Nios II Programs

The Nios II EDS provides two distinct design flows for creating Nios II programs. You can work entirely within the Nios II integrated development environment (IDE), or you can use the Nios II software build tools in command line and scripted environments and then import your work into the IDE for debugging.

The two design flows are not interchangeable. Source code for your applications, libraries, and drivers works in either flow, but the makefiles in the two flows are different and not compatible. Once you have committed to using one design flow, you cannot switch to using the other design flow for that project without starting over.

The Nios II IDE Design Flow

In the Nios II IDE design flow, you create, modify, build, run, and debug Nios II programs with the Nios II IDE graphical user interface (GUI). The IDE creates and manages your project makefiles for you. This design flow is best if you only need limited control over the build process and the project settings, and do not require customized scripting.

The Nios II IDE is based on the popular Eclipse IDE framework and the Eclipse C/C++ development toolkit (CDT) plug-ins. The Nios II IDE runs other tools behind the scenes, shields you from the details of low-level tools, and presents a unified development environment.

With wizards to assist in creating and configuring projects, the Nios II IDE is easy to use, and is especially helpful for Nios II beginners. The Nios II IDE is available on both Windows and Linux operating systems.



For details about the Nios II IDE, refer to the *Nios II Integrated Development Environment* chapter of the *Nios II Software Developer's Handbook*.

The Nios II Software Build Tools Design Flow

In the Nios II software build tools design flow, you create, modify, build, and run Nios II programs with commands typed at a command line or embedded in a script. This design flow is best if you need fine control over the build process and the project settings, or if you require customized scripting.

The Nios II software build tools are utilities and scripts that provide similar functionality to the **New Project** wizard and the **System Library** properties page in the Nios II IDE. The Nios II software build tools design flow allows you to integrate your Nios II software development with other parts of your development flow. Using scripting, your software development flow is fully repeatable and archivable.

At debug time, you import your Nios II software build tools projects into the IDE as Nios II IDE projects for debugging. You can further edit, rebuild, run, and debug your imported application project in the IDE. You can also import library and BSP projects to be available for source code viewing in the debugger, however, imported library and BSP projects are not directly buildable in the IDE.

Like the Nios II IDE, the Nios II software build tools are available on both Windows and Linux operating systems. The Nios II software build tools are the basis for Altera's future Nios II development.



For further information about the Nios II software build tools, refer to the *Introduction to the Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*

Design Flow Tools

This section introduces the tools you use to create Nios II programs for each design flow. The tables are organized to help you determine the level of control you need. You can use the tables as a quick-reference guide to remind you of the differences between the design flows.

Table 1-1 shows the tools that offer a highly-automated level of control for tasks in each Nios II design flow. At this level of control, you create an entire example Nios II program (consisting of an application project and BSP project) or just an example BSP project from Altera-provided software examples using default settings. Use the highly-automated tools as a starting point for your development or when you do not need customization.

Task	Nios II IDE Design Flow	Nios II Software Build Tools Design Flow
Building an example Nios II program	File > New > Nios II C/C++ Application	create-this-app script
Building an example BSP	File > New > Nios II System Library	create-this-bsp script
Debugging	Run > Debug As > Nios II Hardware	<ul style="list-style-type: none"> ● File > Import > Altera Nios II > Existing Nios II software build tools project or folder into workspace ● Run > Debug As > Nios II Hardware

Table 1–2 shows the tools that offer an intermediate level of control for tasks in each Nios II design flow. At this level of control, you create a Nios II program from your custom code or from Altera-provided software examples. Use the intermediate tools as a starting point for your development when you need more control than the default settings provide.

Task	Nios II IDE Design Flow	Nios II Software Build Tools Design Flow
Building an application	File > New > Nios II C/C++ Application	nios2-app-generate-makefile utility
Building a library	File > New > Nios II C/C++ Library	nios2-lib-generate-makefile utility
Building a BSP	<ul style="list-style-type: none"> • File > New > Nios II System Library • Project > Properties > System Library 	nios2-bsp script
Debugging	Run > Debug As > Nios II Hardware	<ul style="list-style-type: none"> • File > Import > Altera Nios II > Existing Nios II software build tools project or folder into workspace • Run > Debug As > Nios II Hardware

Table 1–3 shows the tools that offer an advanced level of control for tasks related to BSPs in each Nios II design flow. At this level of control, you create a Nios II BSP with sophisticated, scriptable control. Use the advanced tools when you need total control over the BSP build process and the BSP project settings.

Task	Nios II IDE Design Flow	Nios II Software Build Tools Design Flow
Building a BSP	Scriptable control of a system library project is not supported.	<ul style="list-style-type: none"> • nios2-bsp-create-settings utility • nios2-bsp-generate-files utility • Tcl scriptable
Updating a BSP	Scriptable control of a system library project is not supported.	<ul style="list-style-type: none"> • nios2-bsp-update-settings utility • nios2-bsp-generate-files utility • Tcl scriptable
Querying a BSP	Not supported.	<ul style="list-style-type: none"> • nios2-bsp-query-settings utility • Tcl scriptable
Customizing newlib	Not supported.	nios2-bsp script using CUSTOM_NEWLIB_FLAGS setting

Additional EDS Support

In addition to the Nios II IDE and Nios II software build tools, the Nios II EDS includes the following items:

- GNU Tool Chain
- Instruction Set Simulator
- Example Designs

GNU Tool Chain

The Nios II compiler tool chain is based on the standard GNU gcc compiler, assembler, linker, and make facilities.



For more information on GNU, see www.gnu.org.

Instruction Set Simulator

The Nios II instruction set simulator (ISS) allows you to begin developing programs before the target hardware platform is ready. The Nios II IDE allows you to run programs on the ISS as easily as running on a real hardware target.

Example Designs

The Nios_II EDS includes software examples and hardware designs to demonstrate all prominent features of the Nios II processor and the development environment.

Third-Party Support



Several third-party vendors support the Nios II processor, providing products such as design services, operating systems, stacks, other software libraries, and development tools.

For the most up-to-date information on third-party support for the Nios II processor, visit the Nios II processor home page at www.altera.com/nios2.

Migrating from the First-Generation Nios Processor

If you are a user of the first-generation Nios processor, Altera recommends that you migrate to the Nios II processor for future designs. The straightforward migration process is discussed in *AN 350: Upgrading Nios Processor Systems to the Nios II Processor*.

Further Nios II Information

This handbook is one part of the complete Nios II processor documentation suite. Consult the following references for further Nios II information:

- The *Nios II Processor Reference Handbook* defines the processor hardware architecture and features, including the instruction set architecture.
- The *Quartus® II Handbook, Volume 5: Embedded Peripherals* provides a reference for the peripherals distributed with the Nios II processor. This handbook describes the hardware structure and Nios II software drivers for each peripheral.
- The Nios II IDE provides tutorials and complete reference for using the features of the GUI. The help system is available within the Nios II IDE.
- The Altera Knowledge Database is an Internet resource that offers solutions to frequently asked questions via an easy-to-use search engine. Go to answers.altera.com/altera/index.jsp.
- Altera application notes and tutorials offer step-by-step instructions on using the Nios II processor for a specific application or purpose. These documents are available on the Literature: Nios II Processor page at www.altera.com/literature/lit-nio2.jsp.

Referenced Documents

This chapter references the following documents:

- *The Hardware Abstraction Layer* section of the *Nios II Software Developer's Handbook*
- *HAL API Reference* chapter of the *Nios II Software Developer's Handbook*
- *Nios II Integrated Development Environment* 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*
- *AN 350: Upgrading Nios Processor Systems to the Nios II Processor*
- *Nios II Processor Reference Handbook*
- *Quartus II Handbook, Volume 5: Embedded Peripherals*

Document Revision History

Table 1–4 shows the revision history for this document.

Date & Document Version	Changes Made	Summary of Changes
May 2008 v8.0.0	Added “What’s New” on page 1–1.	<ul style="list-style-type: none"> ● .sopcinfo file ● Example designs removed from EDS ● MMU added to Nios II core
October 2007 v7.2.0	No change from previous release.	
May 2007 v7.1.0	<ul style="list-style-type: none"> ● Revised entire chapter to introduce Nios II EDS design flows, Nios II programs, Nios II software build tools, and Nios II BSPs. ● Added table of contents to Introduction section. ● Added Referenced Documents section. 	Nios II software build tools
March 2007 v7.0.0	No change from previous release.	
November 2006 v6.1.0	No change from previous release.	
May 2006 v6.0.0	No change from previous release.	
October 2005 v5.1.0	No change from previous release.	
May 2005 v5.0.0	No change from previous release.	
May 2004 v1.0	Initial Release.	

