

# 2. Nios II Integrated Development Environment

#### NII52002-7.1.0

## Introduction

This chapter familiarizes you with the main features of the Nios<sup>®</sup> II integrated development environment (IDE). This chapter is only a brief introduction to the look and feel of the Nios II IDE—it is not a user guide. The easiest way to get started using the Nios II IDE is to launch the tool and perform the Nios II software development tutorial, available in the help system.



The figures in this chapter may differ slightly from the actual GUI due to improvements in the software.

For more information on all IDE-related topics, refer to the Nios II IDE help system.

This chapter contains the following sections:

- "The Nios II IDE Workbench" on page 2–1
- "EDS Design Flows and the IDE" on page 2–3
- "Creating a New IDE-Managed Project" on page 2–4
- "Building and Managing Projects" on page 2–5
- "Running and Debugging Programs" on page 2–7
- "Importing User-Managed Projects" on page 2–10
- "Programming Flash" on page 2–19
- "Help System" on page 2–21

## The Nios II IDE Workbench

The term "workbench" refers to the desktop development environment for the Nios II IDE. The workbench is where you edit, compile and debug your programs in the IDE. Figure 2–1 on page 2–2 shows an example of the workbench.

Figure 2–1. The Nios II IDE Workbench

Nios II C/C++ - hello_world.c - N	ios II IDE 📃 🗖 🔀
<u>File E</u> dit Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> rch !	Project Tools <u>R</u> un <u>W</u> indow <u>H</u> elp
<b>□                                   </b>	🖞 • 🞯 • 🗄 🏇 • 🔕 • 🧏 • 🗄 🥭 🔗 🗄 🔂 • 📑 📑 Nios II C/C++
Nios II C/C++ Projects 🗙 🦳 🗆	le hello_world.c \ □ 🗄 o \ 🕱 🖓 🖓
	<pre>#include <stdio.h> I int main() ( printf("Hello from Nios II!\n"); return 0; ) </stdio.h></pre>
i readme.txt ⊞ i i readme.txt i i i i i i i i i i i i i i i i i i i	Problems Console X Properties EX % In all Problems Console X Properties Figure 1 and Console X Properties Console
	nios2-terminal: (Use the IDE stop button or Ctrl-C to termin
	Hello from Nios II!
/hello_world_0/hello_world.	

### Perspectives, Editors, and Views

Each workbench window contains one or more perspectives. Each perspective provides a set of capabilities for accomplishing a specific type of task. For example, Figure 2–1 shows the Nios II C/C++ development perspective.

Most perspectives in the workbench comprise an editor area and one or more views. An editor allows you to open and edit a project resource (i.e., a file, folder, or project). Views support editors, provide alternative presentations, and ways to navigate the information in your workbench. Figure 2–1 shows a C program open in the editor, and the Nios II C/C++ Projects view in the left-hand pane of the workbench. The Nios II C/C++ Projects view displays information about the contents of open Nios II projects.

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the workbench window contain operations that are applicable to the active editor. Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (\*) indicates that an editor has unsaved changes. Views can also provide their own menus and toolbars, which, if present, appear along the top edge of the view. To open the menu for a view, click

the drop-down arrow icon at the right of the view's toolbar or right-click in the view. A view might appear on its own, or stacked with other views in a tabbed notebook.

# **EDS Design** Flows and the IDE

The Nios II IDE is an integral part of both Nios II embedded design suite (EDS) design flows. The main distinction between the two design flows is in the management of the project.

## IDE-Managed Projects and Makefiles

In the Nios II IDE design flow, the IDE manages Nios II C/C++application and system library projects and makefiles that you create with the New Project wizard in Nios II IDE. In IDE-managed projects, the IDE manages the makefiles for you. The best way to modify and build an IDE-managed project is through the IDE. You manage the system library project settings with the System Library page of the Properties dialog box.

You can manually convert an IDE-managed project to a user-managed project. For details, see the Nios II Software Build Tools chapter of the Nios II Software Developer's Handbook.

## User-Managed Projects and Makefiles

In the Nios II software build tools design flow, you manage Nios II application, library, and BSP projects and makefiles, giving you total control. Typically, you create user-managed projects outside of the IDE and then import them into the IDE for debugging.

The best way to create a user-managed project, and modify the settings, is with the Nios II software build tools through a command shell or scripting tool. You can create the makefile by hand, or you can use the Nios II software build tools to create it.

When you import a user-managed C/C++ application project into the IDE for debugging, the Nios II IDE does not manage the project or makefile. The IDE does not support management of or changes to a BSP project after import. You must manage the BSP with the software build tools, outside of the IDE.

IP

User-managed projects and IDE-managed projects are not interchangeable.

## Creating a New IDE-Managed Project

The Nios II IDE provides a **New Project** wizard that guides you through the steps to create new IDE-managed projects. To start the **New Project** wizard for Nios II C/C++ application projects, on the File menu in the Nios II C/C++ perspective, point to **New**, and then click **Nios II C/C++ Application**, as shown in Figure 2–2.

Figure 2–2. Starting the Nios II C/C++ Application New Project Wizard

<mark>ile E</mark> dit Refac <u>t</u> or <u>N</u> a	wigate Se <u>a</u> rch <u>P</u> roject	Tools <u>R</u> un <u>W</u> indow <u>H</u> elp	
New	Alt+Shift+N	Project	
Open File <u>.</u>		Nios II C/C++ Application	
⊆lose	Ctrl+W	Nios II System Library	
Close All	Ctrl+Shift+W	Nios II User-Makefile C/C++ Application	
<u>S</u> ave	Ctrl+S	🕈 Nios II C/C++ Library	
Save <u>A</u> s		😂 Source Folder	
📄 Sav <u>e</u> All	Ctrl+Shift+S	C Source File	
Revert		h Header File	
Mo <u>v</u> e		G Class	
Rena <u>m</u> e	F2	<b>□</b> ↑ Other	
Refresh	F5		_

The Nios II C/C++ application **New Project** wizard prompts you to specify:

- 1. A name for your new Nios II project.
- 2. The target hardware.
- 3. A template for the new project.

Project templates are ready-made, working designs that serve as examples to show you how to structure your own Nios II projects. It is often easier to start with a working "Hello World" project, than to start a blank project from scratch.

Figure 2–3 shows the Nios II C/C++ application **New Project** wizard, with the template for a Dhrystone benchmark design selected.

iloszeds (examples (ver	unditunger_chroniert_scolorgi ingendisor (waterbinitikscolie_o
Name: dhrystone	<u>_</u> 0
Specify Location	
Location: C:\altera	\61\nios2eds\examples\verilog\niosII_cycloneII_2c35
Select Target Hardwa	2/2
SOPC Builder System:	xamples)verilog/niosII_cycloneII_2c35)standard/std_2C35.ptf 👽 Browse
CPU:	(cpu
Select Project Temple Blank Project Board Diagnostics Count Binary Custom Instruction T Drygstone Hello Freestanding Hello MicroC/OS-II Hello World Small Host File System MicroC/OS-II Messag	Image: Second
Memory Test MicroC/OS-II Messag	ge Box

Figure 2–3. The Nios II C/C++ Application New Project Wizard

After you click **Finish**, the Nios II IDE creates the new project. The IDE also creates a system library project, **\*\_syslib** (for example, **dhrystone\_0\_syslib** for Figure 2–3 on page 2–5). These projects show up in the Nios II C/C++ Projects view of the workbench.

# Building and Managing Projects

Right-clicking on any resource (a file, folder, or project) opens a contextsensitive menu containing commands that you can perform on the resource. Right-clicking is usually the quickest way to find the command you need, though commands are also available in menus and toolbars.

To compile a Nios II project, right-click the project in the Nios II C/C++ Projects view, and click **Build Project**. Figure 2–4 on page 2–6 shows the context-sensitive menu for the project dhrystone\_0, with the **Build Project** option chosen. When building, the Nios II IDE first builds the system library project (and any other project dependencies), and then compiles the main project. Any warnings or errors are displayed in the Tasks view.



Figure 2–4. Building a Project Using the Context-Sensitive (Right-Click) Menu

Right-clicking a project in the Nios II C/C++ Projects view also allows you to access the following important options for managing the project:

- Properties—Manage the dependencies on target hardware and other projects
- System Library Properties—Manage hardware-specific settings, such as communication devices and memory partitioning
- **Build Project**—i.e., make

- **Run As**—Run the program on hardware or under simulation
- **Debug As**—Debug the program on hardware or under simulation

## Running and Debugging Programs

Run and debug operations are available by right-clicking the Nios II project. The Nios II IDE allows you to run or debug the project either on a target board, under the Nios II instruction set simulator (ISS), or under ModelSim<sup>®</sup>. For example, to run the program on a target board, right-click the project in the Nios II C/C++ Projects view, point to Run As, and then click **Nios II Hardware**. See Figure 2–5. Character I/O to stdout and stderr are displayed in the Console view.

#### Figure 2–5. Running a Program on Target Hardware



Starting a debug session is similar to starting a run session. For example, to debug the program on the ISS, right-click the project in the Nios II C/C++ Projects view, point to **Debug As**, and then click **Nios II Instruction Set Simulator**. See Figure 2–6 on page 2–8.





Figure 2–7 on page 2–9 shows a debug session in progress for the dhrystone\_0 project.

Figure 2–7. Debugging dhrystone\_0 on the ISS

💽 Debug - dhry21a.c - Nios II IDE		
Eile Edit Refactor Navigate Search Project Tools <u>R</u> un <u>Wi</u> ndow <u>H</u> elp		
i 📬 • 🖫 👜   🗟 i 🕸 • 🔕 • i 🙋 🖋 i 🙆 • i 🖢 - 🖓 - 🖘 🗇 •	⇔ -	😭 🐝 Debug 💙
🕸 Debug 🛛 🥂 🍇 🤣 🕪 🗉 🔳 💐 🚴 🐼 🖧 🧮 😿 🖬 🍸 🗖	😂 Variables 🗙 Breakpoints Registers Mode	ules 🐁 📲 📄 🦨 🗶 🗶 🔽 🗖
🖃 💹 dhrystone_0 Nios II ISS configuration [Nios II Instruction Set Simulator]	Name	Value
😑 🧽 Nios II Elf Debugger (11/15/06 2:49 PM) (Suspended)	(X)= Int_1_Loc	0
□ 🔐 Thread [0] (Suspended)	(X)= Int_2_Loc	10
3 main() at\dhry21a.c:223 UxU4UUU388	(×)= Int_3_Loc	0
2 aic_main() at (cygorive(c)aitera(o1(nios2eos(components)aitera_nai(niAL)src(aic_ 1 ctart() at (cygorive)c)aitera(o1)in(nios2 in)aitera_nios2(H0L)src(crt0 Sr356 0x(	(X)= Ch_Index	Tdeek 4
I	(X)= Enum_Loc	Ident_1
db (11/15/06 2:49 PM)		~
	<	>
		<u>^</u>
		×
🖻 dhry21a.c 🛛		Outline Disassembly 🛛 🗌
<pre>/* Arr_2_Glob [8][7] would have an undefined value.</pre>	*/ 🔼	0x0400037c <main+312>: mo</main+312>
/* Warning: With 16-Bit processors and Number_Of_Run	s > 32000, */	0x04000380 <main+316>: ad</main+316>
/* overflow may occur for this array element.	*/	0x04000384 <main+320>: ca.</main+320>
		printf ("2. The performation
printi ("\n"); printf ("Derustone Benchmark, Version 2.1 (Lenguage, C)\n"		UXU4UUU388 <main+324>: mov</main+324>
printf ("briyscole benchmark, version 2.1 (Language: C)(h-		0x04000390 <main+328>: au</main+328>
printf ("If you are running this on the Nios II ISS, pleas	e be aware of the following	printf ("since the Nics
printf ("1. This may take several minutes to run.\n");	e at anale of one fortowing	0x04000394 <main+336>; mov</main+336>
printf ("2. The performance on the ISS will be optimistic of	compared to running on hard	0x04000398 <main+340>: ad</main+340>
printf ("since the Nios II ISS is not cycle accurate.\n");		0x0400039c <main+344>: ca.</main+344>
<pre>printf ("\n");</pre>		<pre>printf ("\n");</pre>
/*	✓	0x040003a0 <main+348>: mov</main+348>
	>	< >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Console 🕅 Tasks Memory	8	🗶 🔆 🕞 🚮 🛃 🛃 - 🗂 - 🗆 🗋
dhrystone_0 Nios II ISS configuration [Nios II Instruction Set Simulator] Nios II Instruction Set Simulator	(11/15/06 2:49 PM)	
		<u> </u>
Dhrystone Benchmark, Version 2.1 (Language: C)		
If you are running this on the Nice II ISS please he aware of	the following:	
1. This may take several minutes to run.	che forfowing.	
K		
: U'		

Launching the debugger changes the workbench perspective to the debug perspective. You can easily switch between the debug perspective and the Nios II C/C++ development perspective by clicking on the **Open Perspective** icon at the upper right corner of the workbench window.

After you start a debug session, the debugger loads the program, sets a breakpoint at main(), and begins executing the program. You use the usual controls to step through the code: Step Into, Step Over, Resume, Terminate, etc. To set a breakpoint, double click in the left-hand margin of the code view, or right-click in the margin and then click Add Breakpoint.

The Nios II IDE offers many views that allow you to examine the status of the processor while debugging: Variables, Expressions, Registers, Memory, etc. Figure 2–8 on page 2–10 shows the Registers view.

Figure 2–8. The Registers View While Debugging

											×
Run	<u>Window</u> <u>H</u> elp	_									
<b>9</b> :	<u>N</u> ew Window New Editor	• 🔿 -		 				E	🕈 🏇 Debu		»
lwarel	Open Perspective	N 3. 3.	n 🗟   🖹 🖬 🔨	Variables	Breakpoints	NN Reg	gisters 🗙	Modules	<b>∰ €</b>		
ended)	Show <u>V</u> iew	© <sub>⊜</sub> <u>B</u> reakpoints	Alt+Shift+Q, B		1ain			Value			
	Customize Perspective	📃 <u>C</u> onsole	Alt+Shift+Q, C	0101	ll zero			0×0			
248	Save Perspective As	🏇 <u>D</u> ebug		1	li at			0xdeadbee	f		
os2eds	Reset Perspective	🚮 Disassembly		1	11 r2			0x0			
ios2_ip	⊆lose Perspective	👰 Error Log		1	11 r3			0×1			
utput (	Close All Perspectives	Expressions		1	?!? r4			Π×Π		>	-
	Navigation 🕨	Memory									^
	🔁 Working Sets	Cutline									$\sim$
	Preferences	IIIII <u>R</u> egisters	N	 <						>	
L		ā Signals	~				Outline	🗓 Disassemb	ly 23 yl	- [	
		🚈 <u>T</u> asks				^	}			^	
		🔵 <u>T</u> race					#endi	f			
		(×)= <u>V</u> ariables	Alt+Shift+Q, V				Nex	t Ptr G	lob = (H	lec Po:	
alloc	( <b>sizeof</b> (Rec_Type)	<u>O</u> ther	Alt+Shift+Q, Q				♦ 0x040	00248 <	main+4>:	mov	
(siz	<pre>eof (Rec_Type));</pre>						0x040	00274 <	main+48>	ca.	
	= Next_Ptr_Glob;						0x040 Ptr	0027c < Glob =	(Rec_Pc	inter;	

# Importing User-Managed Projects

In the Nios II software build tools design flow, you import user-managed projects, created with the Nios II software build tools, into the IDE for debugging. This section discusses that process.

When you create a C/C++ application (and its associated BSP) with the Nios II software build tools, the application is ready to import into the IDE as a user-managed project. No additional preparation is necessary.

The IDE imports four kinds of Nios II software build tools projects:

- User-managed C/C++ application project
- User-managed board support package (BSP) project
- User-managed library project
- C/C++ source project (a directory tree containing supporting source code)

Table 2–1. IDE Capabilities for Imported	d Projects			
Type of project	Source editable?	Buildable?	Debuggable?	Settings Manageable?
User-managed C/C++ application project	Yes	Yes	Yes	No
User-managed BSP project	Yes	No(1)	(2)	No
User-managed library project	Yes	No(1)	(2)	No
C/C++ source project	Yes	No	(2)	No

The IDE treats each type of imported project as listed in Table 2–1.

#### Notes to Table 2–1:

- (1) When the IDE builds a C/C++ application project, it also builds the associated BSP, and any associated libraries. It is not necessary to import BSPs and libraries to build them as part of a C/C++ application in the IDE.
- (2) When the IDE debugs a C/C++ application, it can step into any associated BSP, library, or supporting C/C++ source code that you have imported.

The IDE imports each type of project through the **Import** wizard. The **Import** wizard determines the kind of project you are importing, and configures it appropriately.

You can continue to develop project code in your user-managed project after importing the project into the IDE. You can edit source files and rebuild the project, using either the IDE tool chain or the software build tools. However, you must manage BSP settings with the software build tools.



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

### **Road Map**

Importing and debugging a project typically involves several of the following tasks. You do not need to perform these tasks in this order, and you can repeat or omit some tasks, depending on your needs.

- Import a user-managed C/C++ application
- Import a supporting project
- Debug a user-managed C/C++ application
- Edit user-managed C/C++ application code

The following sections describe these tasks.

### Import a User-Managed C/C++ Application

To import a user-managed C/C++ application, perform the following steps:

- 1. Launch the IDE.
- 2. On the File menu, click **Import**. The **Import** dialog box appears.

Figure 2–9. Launching the Import Wizard

Import	n an	×
Select		r <sup>2</sup> 1
Select an import	source:	
type filter text		
<ul> <li>Altera</li> <li>Attera</li> <li>Ext.</li> <li>C/C++</li> <li>C</li> <li>CV5</li> <li>CV5</li> <li>Team</li> </ul>	sting Altera Nios II Project into Workspace sting Command-Line Project or Folder Into	Workspace
0	Sack Next >	Finish Cancel

- 3. Expand the Altera Nios II folder, and select Existing Nios II software build tools project or folder into workspace, as shown in Figure 2–9.
- 4. Click **Next**. The **Import** wizard appears, as shown in Figure 2–10 on page 2–13.

e workspace.			
roject name:			
Project contents: 📗			Browse
ease select the direct	tory which contains existing :	source code.	
nis directory should c hich you want to mał	ontain a project Makefile, or ke visible to the workspace.	be the root directory of a se	et of C/C++ files
his directory should c hich you want to mał	ontain a project Makefile, or ke visible to the workspace.	be the root directory of a se	et of C/C++ files
nis directory should c hich you want to mał	ontain a project Makefile, or ke visible to the workspace.	be the root directory of a se	at of C/C++ files

Figure 2–10. User-Managed Project Import Wizard

- 5. Click **Browse** and locate the directory containing the C/C++ application project to import.
- 6. Click **OK**. The wizard fills in the project name and path, as shown in Figure 2–11. The project name is the directory name. You can override the project name by typing a new name in the **Project name** box.

#### Figure 2–11. Importing a User-Managed C/C++ Application

nport Project From	h File System
:\altera\71\nios2eds\ex	amples\verilog\niosII_stratix_1s10\standard\software_examples\
-	
Project name:   hello_led	
Project contents: C:\al	ltera\71\nios2eds\examples\verilog\niosII_stratix_1s10\stanc Browse
lease select the director	y which contains existing source code.
his directory should cont	tain a project Makefile, or be the root directory of a set of C/C++ files
which you want to make v	visible to the workspace.
	1 1 1

You might see a warning saying "There is already a **.project** file at: *<path>*". This warning indicates that the directory already contains an IDE project. Either it is an IDE-managed project, or it is a user-managed project that is already imported into the IDE.

If the project is not already in your workspace, you can import it, but be aware that the **Import** wizard does not convert it from one type to another.

If the project is already in your workspace, do not re-import it.

7. Click **Finish**. The wizard imports the project, creating a new C/C++ application project in the workspace.

At this point, the IDE can build, debug, and run the complete program, including the BSP and any libraries, by using the user-managed makefiles in your imported C/C++ application project. The IDE can display and step through application source code, exactly as if the project were IDE-managed. However, the IDE does not have direct information about where BSP or library code resides. If you need to view, debug or step through BSP or library source code, you need to import the BSP or library. The process of importing supporting projects, such as BSPs and libraries, is described in the next section.

## Import a Supporting Project

While debugging a C/C++ application, you might need to view, debug or step through source code in a supporting project, such as a BSP or library. To make supporting project source code visible in the IDE debug perspective, you need to import the supporting project.

If you do not need BSP or library source code visible in the debugger, you can skip this task, and go directly to "Debug a User-Managed C/C++ Application" on page 2-17.

To debug or step through the code in the GNU newlib library, import the **nios2-gnutools** folder as a C/C++ source project, as described in "Importing a C/C++ Source Project" on page 2–16.

If you have several C/C++ applications based on one BSP or library, import the BSP or library once, and then import each application that is based on the BSP or library. Each application's makefile contains the information needed to find and build any associated BSP or libraries.

#### Importing a User-Managed BSP

To import a user-managed BSP project, perform the following steps:

- 1. On the File menu, click **Import**. The **Import** dialog box appears.
- 2. Expand the Altera Nios II folder, and select Existing Nios II software build tools project or folder into workspace.
- 3. Click Next. The Import wizard appears.
- 4. Click **Browse** and locate the directory containing the BSP project to import.
- 5. Click **OK**. The wizard fills in the project name and path. The project name is the directory name. You can override the project name by typing a new name in the **Project name** box.
  - You might see a warning saying "There is already a **.project** file at: *<path>*". This warning indicates that the directory already contains an IDE project. Either it is an IDE-managed project, or it is a user-managed project that is already imported into the IDE.

If the project is not already in your workspace, you can import it, but be aware that the **Import** wizard does not convert it from one type to another.

If the project is already in your workspace, do not re-import it.

- 6. Click **Finish**. The wizard imports the project, creating a new BSP project in the workspace.
- After import, a user-managed BSP looks the same as a usermanaged C/C++ application. However, you cannot directly build or run a user-managed BSP in the IDE.

#### Importing a User-Managed Library

To import a user-managed library, perform the following steps:

- 1. On the File menu, click **Import**. The **Import** dialog box appears.
- 2. Expand the Altera Nios II folder, and select Existing Nios II software build tools project or folder into workspace.
- 3. Click Next. The Import wizard appears.

- 4. Click **Browse** and locate the directory containing the library project to import.
- 5. Click **OK**. The wizard fills in the project name and path. The project name is the directory name. You can override the project name by typing a new name in the **Project name** box.
  - You might see a warning saying "There is already a **.project** file at: *<path>*". This warning indicates that the directory already contains an IDE project. Either it is an IDE-managed project, or it is a user-managed project that is already imported into the IDE.

If the project is not already in your workspace, you can import it, but be aware that the **Import** wizard does not convert it from one type to another.

If the project is already in your workspace, do not re-import it.

- 6. Click **Finish**. The wizard imports the project, creating a new library project in the workspace.
- After import, a user-managed library looks the same as a usermanaged C/C++ application. However, you cannot directly build or run a user-managed library in the IDE.

Importing a C/C++ Source Project

To import a C/C++ Source Project, such as newlib, perform the following steps:

- 1. On the File menu, click **Import**. The **Import** dialog box appears.
- 2. Expand the Altera Nios II folder, and select Existing Nios II software build tools project or folder into workspace.
- 3. Click Next. The Import wizard appears.
- 4. Click **Browse** and locate the directory containing the C/C++ source project to import.
- 5. Click **OK**. The wizard fills in the project name and path. The project name is the directory name. You can override the project name by typing a new name in the **Project name** box.

You might see a warning saying "There is already a **.project** file at: *<path>*". This warning indicates that the directory already contains an IDE project. Either it is an IDE-managed project, or it is a user-managed project that is already imported into the IDE.

If the project is not already in your workspace, you can import it, but be aware that the **Import** wizard does not convert it from one type to another.

If the project is already in your workspace, do not re-import it.

- 6. Click **Finish**. The wizard imports the project, creating a new C/C++ source project in the workspace.
- After import, user-managed C/C++ source code looks the same as a user-managed C/C++ application. However, you cannot directly build or run user-managed C/C++ source code in the IDE.

## Debug a User-Managed C/C++ Application

To debug an imported user-managed C/C++ application project, perform the following steps:

1. In the Nios II C/C++ Projects view, right click the project name, point to **Debug As**, and click **Nios II Hardware**.

The debug configuration shows the message: "Specify an SOPC Builder system PTF file", as in Figure 2–12 on page 2–18. The debugger needs information about the target system in order to establish communications.



Debug		×
Create, manage, and run conf	igurations	the second secon
Specify an SOPC Builder system PTF	file.	
Sype filter text      C (/C++ Attach to Local App      C (/C++ Local Application      C (/C++ Local Application      C (/C++ Local Application      Nios II Hardware      Nios II Hardware      Nios II Instruction Set Simu      Nios II Multiprocessor Colle	Name:       hello_led Nios II HW configuration         Image:       Target Connection       Image:       Image:         Project:       Image:       Image:       Image:         hello_led       Image:       Image:       Image:         Nios II ELF Executable:       Image:       Image:       Image:         Target Hardware       Image:       Image:       Image:         SOPC Builder System PTF File:       Image:       Image:         CPU:       Image:       Image:         Additional nios2-download arguments:       Image:       Image:	Help Browse Search Browse
× >	Validate Nios II system ID before software download	Revert
0	Debug	Close

- 2. Click Browse at the right of the SOPC Builder System PTF File box.
- 3. Locate the SOPC Builder System File (**.ptf**) on which the application's BSP is based. For example, if you are using a Nios II software example, the SOPC Builder System File is three levels up in the directory tree from the software project.

After you select the file, the message disappears, as shown in Figure 2–13 on page 2–19.



Debug		×
Create, manage, and run conf	figurations	T.
C/C++ Attach to Local App         C/C++ Local Application         C/C++ Local Application         C/C++ Destmortem debug;         Nios II Hardware         Initian Hello, led Nios II HW coil         Nios II Instruction Set Simu	Name: hello_led Nios II HW configuration	Help Browse
II Multiprocessor Colle	Nics II ELF Executable: hello_led.elf Target Hardware SOPC Builder System PTF File: C:\altera\71\nios2eds\examples\verilog\niosII_stral CPU: Additional nios2-download arguments:instance=0	Browse Load JDI File
	Validate Nios II system ID before software download	Revert
	Apply	Revert
0	Debug	Close

4. Click Apply.

Your software application is ready to run or debug exactly as you would run or debug an IDE-managed project. For details about running and debugging applications in the Nios II IDE, see "Running and Debugging Programs" on page 2–7.

## Edit User-Managed C/C++ Application Code

You can edit the code in an imported user-managed project with the editor exactly the same way you edit the code in an IDE-managed project.

## Programming Flash

Many Nios II processor systems use external flash memory to store one or more of the following items:

Program code

- Program data
- FPGA configuration data
- File systems

The Nios II IDE provides a Flash Programmer utility to help you manage and program the contents of flash memory. Figure 2–14 on page 2–21 shows the Flash Programmer.

- To program a user-managed C/C++ application to flash memory, you must first specify an SOPC Builder System File, as follows:
- 1. Click Browse at the right of the SOPC Builder System PTF File box.
- 2. Locate the SOPC Builder System File on which the application's BSP is based. For example, if you are using a Nios II software example, the SOPC Builder System File is three levels up in the directory tree from the software project.

This procedure is identical to specifying the SOPC Builder System File before debugging a user-managed C/C++ application, as described in "Debug a User-Managed C/C++ Application" on page 2–17.

Figure 2–14. The Nios II IDE Flash Programmer

■ × □ ⇒ ·	Name: dhrystone_0 programmer	
pe filter text	Man L Target Connection	
Flash Programmer drystone 0 programmer	Target Board: Nios Development Board, Cyclone II (EP2C35)	Hel
flash programmer	Program software project into flash memory	
	Project:	
	dhrystone_0	Browse
	Nios II ELF Executable:	
	Release/dhrystone_0.elf	<u>S</u> earch
	Citatteralis 11 nins2eddayamplestyerion1nins11_cyclone11_2c35i.v	Browce
		browsen
	CDL CDL	
	CPU: CPU: CPU: CPU CPU: CPU	
	CPU:     Qu     Additional nios2-flash-programmer arguments:      Program FPGA configuration data into hardware-image region of flash memory	
	CPU:     CPU:     Additional nios2-flash-programmer arguments:     Program FPGA configuration data into hardware-image region of flash memory      FPGA Configuration (SOF):     C1{aktra}61/nios2eds{examples}verilog/nios11_cyclone11_2c35{standard,sof	Browse
	CPU:       cpu       v         Additional nios2-flash-programmer arguments:       v       Program FPGA configuration data into hardware-image region of flash memory)         FPGA Configuration (SOF):       C1(altera)61/nios2eds(examples)verilog(nios11_cycloneII_2c35)standard(standard.sof         Hardware Image:       Custom       Memory:       ext_flash       v       Offset:       0x0	Browse
	Additional nios2-flash-programmer arguments:      Additional nios2-flash-programmer arguments:      Program FPGA configuration data into hardware-image region of flash memory)      FPGA Configuration (SOF): C1{aktra}61/nios2eds/examples/verilog/nios11_cycloneII_2c35/standard/standard.sof      Hardware Image: Custom      Memory: ext_flash      Offset: Dx0      Program a file into flash memory	Browse
	CPU:       cpu       v         Additional nios2-flash-programmer arguments:	Browse
	CPU:       cpu       v         Additional nios2-flash-programmer arguments:	Browse
	CPU:       cpu       v         Additional nios2-flash-programmer arguments:	Browse
	CPU:       cpu       v         Additional nios2-flash-programmer arguments:	Browse

## **Help System**

The Nios II IDE help system provides documentation on all IDE topics. To launch the help system, click **Help Contents** on the Help menu. You can also press F1 on Windows (Shift-F1 on Linux) at any time for contextsensitive help. The Nios II IDE help system contains hands-on tutorials that guide you step-by-step through the process of creating, building, and debugging Nios II projects. Figure 2–15 on page 2–22 shows the Nios II IDE help system displaying a tutorial.





# Referenced Documents

This chapter references the following document:

Nios II Software Build Tools chapter of the Nios II Software Developer's Handbook

# DocumentTable 2–2 shows the revision history for this document.Revision History

Table 2–2. Document Revision History		
Date & Document Version	Changes Made	Summary of Changes
May 2007 v7.1.0	<ul> <li>Added instructions for importing user-managed projects</li> <li>Changed chapter title.</li> <li>Added table of contents to Introduction section.</li> <li>Added Referenced Documents section.</li> </ul>	Nios II software build tools
March 2007 v7.0.0	No change from previous release.	
November 2006 v6.1.0	Describes updated look and feel, including Nios II C/C++ perspective and Nios II C/C++ Projects views, renamed project types.	Updated look and feel based on Eclipse 3.2.
May 2006 v6.0.0	No change from previous release.	
October 2005 v5.1.0	Updated for the Nios II IDE version 5.1.	
May 2005 v5.0.0	No change from previous release.	
September 2004 v1.1	Updated screen shots.	
May 2004 v1.0	Initial Release.	