



Getting Started with the TASKING VX-toolset for Nios II

Software, hardware, documentation and related materials:

Copyright © 2006 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, TASKING, and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.

Table of Contents

Installing the Embedded Software	1-1
1.1 Introduction	1-1
1.2 Installing the TASKING VX-toolset for Nios II	1-1
1.2.1 Installation	1-1
1.2.2 Licensing	1-2
1.2.3 Starting / Closing Nios II IDE	1-2
1.3 How to Use the Documentation	1-4
1.4 Related Publications	1-5
Setting up a Project	2-1
2.1 Introduction	2-1
2.2 Create a Project	2-1
2.3 Manually Add a File to Your Project	2-3
2.4 Editing Files: C/C++ Editor	2-4
2.5 Closing and Opening a Project	2-6
2.6 Setting Project Options	2-6
2.7 Build a Project	2-10
2.8 What's Next?	2-10



1 Installing the Embedded Software

Summary

This chapter guides you through the installation process. It also describes which documentation is available and how you best can use it.

1.1 Introduction

The Nios II integrated development environment (IDE) is the software development graphical user interface (GUI) from Altera for the Nios II processor. The Nios II IDE is based on the popular Eclipse platform. Eclipse is both an environment to develop application *with* and to develop applications *for*. The TASKING VX-toolset for Nios II consists of a set of applications that are available as a plugin to the Nios II IDE. Using the Nios II IDE as develop environment, you can use the TASKING VX-toolset for Nios II to develop embedded applications for the Nios II processor.

In this manual, **TASKING VX-toolset for Nios II** and **TASKING toolset** are used as synonyms.

1.2 Installing the TASKING VX-toolset for Nios II

1.2.1 Installation

1. Make sure that the Quartus II Design Software and the Nios II Embedded Design Suite from Altera is installed on your system. For more information see the *Quartus II Installation & Licensing for PCs* manual from Altera.
2. Insert the TASKING VX-toolset for Nios II CD-ROM into your computer.
3. If the installation program does not start automatically, browse to your CD-ROM drive and run the program **setup.exe**.

The TASKING Setup dialog box appears.

4. Select a product and click on the **Install** button.
5. Follow the instructions that appear on your screen.



You can find your serial number on the invoice, delivery note, or picking slip delivered with the product.

1.2.2 Licensing

TASKING products are protected with license management software (FLEXlm). To use a TASKING product, you must install the license key provided by Altium for the type of license purchased.

1. Run the License Administrator during installation and follow the steps to **Request a license key from Altium by E-mail**.
2. E-mail the license request to your local TASKING sales representative. The license key will be sent to you by E-mail.
3. Follow the steps to **Import a license key received from Altium by E-mail**. The License Administrator creates a license file for you.

For more information read Chapter 2, *Licensing the Quartus II Software* in the *the Quartus II Installation & Licensing for PCs* manual from Altera.

1.2.3 Starting / Closing Nios II IDE

Starting Nios II IDE

To start Nios II:

1. From the Windows **Start** menu, select **Programs -> Altera -> Nios II EDS 6.0 -> Nios II IDE**.
The Workspace Launcher dialog appears.
2. Enter the path to the workspace.
In the remainder of this manual, we assume you use the default.
3. Enable the option **Use this as the default and do not ask again**.
4. Click **OK** to proceed.

Initially, the Nios II IDE opens with a workbench displaying the C/C++ perspective.

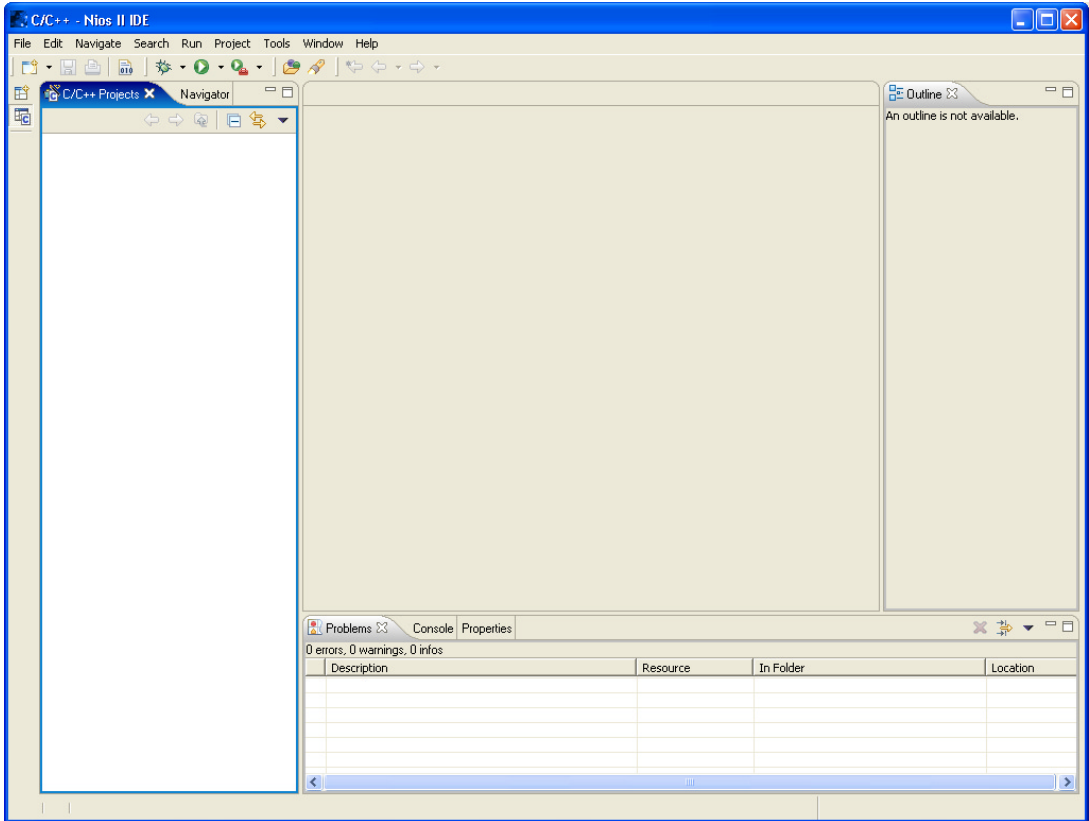


Figure 1-1: Nios II IDE workbench with C/C++ perspective



It is also possible that Nios II IDE opens with the blue **Welcome** view. This view provides some general information and alternative ways to access the on-line documentation. Click the arrow in the upper right corner to reveal the workbench as shown above. To access the Welcome screen again, select **Welcome** from the **Help** menu.

Closing Nios II IDE

To close Nios II IDE:

- From the **File** menu, select **Exit**.



Upon exit, Nios II IDE saves the current workbench layout. The next time you start Nios II IDE, the last saved workbench layout is used.

1.3 How to Use the Documentation

The documentation for the TASKING VX-toolset under the Nios II IDE refers to:

- on-line documentation for Nios II IDE (Eclipse)

and consists of:

- this Getting Started manual
- on-line TASKING documentation:
 - Using the Nios II Embedded Tools (GU0115)
 - Nios II Embedded Tools Reference (TR0130)

It is strongly recommended to read the documentation in this order.

Getting acquainted with Nios II IDE

If you are new to Nios II IDE, start familiarizing with Nios II IDE. Nios II IDE comes with several on-line documents. One document describes how Eclipse is organized as a Workbench, with Perspectives that contain Views; another document explains how to create a sample C/C++ project, build and debug it (CDT documentation). The Nios II IDE Help system contains specific information for the Nios II IDE. Its content overrides any information found in the Eclipse and CDT documentation. To start with this documentation:

1. Start Nios II IDE.
2. From the **Help** menu, select **Help Contents**.
The help screen appears.
3. In the left pane, select **Workbench User Guide** to learn more about working in Eclipse.
4. Continue with **Nios II IDE Help** to learn more about creating and developing a Nios II C/C++ project.



Be aware that this example does not use the TASKING tools, it uses the Nios II GNU tools in Nios II IDE instead.

Getting started with the TASKING VX-toolset for Nios II (this manual)

The next chapter of this manual explains how to setup and work with a TASKING Nios II project. It shows some important features of the TASKING toolset for Nios II.

On-line TASKING Nios II Documentation

Once you are introduced to Nios II IDE and the TASKING toolset, you can start creating your own projects. The on-line documentation for the TASKING toolset covers the TASKING Nios II C/Assembler language, as well as detailed description of the various tools and options. Accessing the documentation for the TASKING toolset is similar to accessing the on-line documentation for Nios II IDE:

1. Start Nios II IDE.
2. From the **Help** menu, select one of the **TASKING Nios II** documents.

1.4 Related Publications

C Standards

- ISO/IEC 9899:1999(E), Programming languages – C [ISO/IEC]
More information on the standards can be found at <http://www.ansi.org>

TASKING Tools

- Using the Nios II Embedded Tools
[Altium, GU0115]
- Nios II Embedded Tools Reference
[Altium, TR0130]

Nios II

- Nios II Processor Reference Handbook
[2006, Altera Corporation, NII5V1–6.0]
- Nios II Software Developer's Handbook
[2006, Altera Corporation, NII5V2–6.0]



2 Setting up a Project

Summary

This tutorial shows how to create an embedded software project with the TASKING toolset for Nios II. It lets you create your own project with a simple hello-world example.

2.1 Introduction

By now you should be familiar with the Nios II IDE workbench, perspectives and views. If you are not, please read the Nios II IDE / Eclipse documentation as described in Section 1.3, [How to Use the Documentation](#), in Chapter *Installing the Embedded Software*.

2.2 Create a Project

Set the C/C++ perspective

Before creating a TASKING Nios II project, it is necessary to have the C/C++ perspective on the workbench.

1. Start Nios II IDE.

Nios II IDE starts with the last saved workbench layout.

2. To open the C/C++ perspective: from the **Window** menu, select **Open Perspective » Other... » C/C++**.

The name of the perspective is displayed in the title bar of the workbench window.



If you attempt to create a TASKING Nios II project while the C/C++ perspective is *not* active, Nios II IDE will ask you to activate the C/C++ perspective after you finish the **New Project** wizard.

Create a TASKING Nios II project with the New Project wizard

1. From the **File** menu select **New » Project...**

The New Project wizard appears.

2. Expand the **Altera Nios II** entry and select **TASKING Nios II C Application**. Click **Next** to continue.

The New Project wizard appears.

3. Enter a name for your project, for example `hello_world_0`.

4. Specify a location for the TASKING Nios II C application project.

In the **Location** box you will see the location where the new project will be stored. To change the default location, you can uncheck the **Use Default Location** check box and browse for an alternative location. However, use the default location for now.

5. Specify the **SOPC Builder System** file (.ptf) that describes your target hardware.
6. Select a specific CPU for the project in the **CPU** list. If your SOPC Builder system has only one CPU, the Nios II IDE sets the CPU automatically.
7. Select a project template in the **Select Project Template** list, for example `Hello World`.

As you click the project templates, information about each template appears in **Description** and **Details** boxes. Each template is a collection of software files and project settings that serve as a base for the new project. Use the **Blank Project** template to avoid copying any files into the new project.

8. If you want the Nios II IDE to create a default associated system library for you, skip to step 11.
9. Click **Next** to continue.
10. Specify how you want to associate the C application project to a system library by doing one of the following:
 - Click **Create a new system library named: `projectname_syslib`**. This option creates a default HAL system library. For now, select this option.
 - Click **Select or create a system library**, then select an existing system library in the **Available System Library Projects** list. Alternatively, you can click **New System Library Project...** to create a new system library project.
11. Click **Finish** to finish the wizard and to create the project.

The Nios II IDE creates new directories for your C application project and system library project. The IDE copies the template source files into the project directories. The two new projects are displayed in the left-hand pane of the IDE. `hello_world_0` is your TASKING Nios II C application project and `hello_world_0_system` is a system library that defines the target hardware.

The left-hand pane in the IDE has two views. The **C/C++ Projects** view shows the structure of your projects, complete with all files that are used in the project.

The **Navigator** view lets you navigate through the physical project folder and project files on your hard disk. The navigator view does not show the include files needed for your project, because these files are stored in the general `... \include` folder, not in your project folder.



In the standard Eclipse documentation about the workbench is described how you can move and organize views on your workbench.

2.3 Manually Add a File to Your Project

We will recreate the project as described in Section 2.2, *Create a Project*; however, this time without the automatic 'Hello world' example C source file. Instead, the example below illustrates how you can manually add a file to your project.

Recreate your project without 'Hello world' template

First repeat steps 1 through 6 of *Create a TASKING Nios II project with the New Project wizard* in Section 2.2, *Create a Project*. Continue with:

7. In the **Select Project Template** list, select **Blank Project**.
8. Click **Finish** to finish the wizard and to create the project.

Add a new file to your project

To add a new, empty file:

1. In the C/C++ Projects view, right-click on the name of the project, `hello_world_0`, and select **New » File**.

The New File dialog appears.

2. Specify a source folder and a name for the new file. By default, the new file will be stored in the project folder (in this case: `hello_world_0`). If your projects contains multiple folders, you can browse for an alternative source the folder to store the new file in.

- In the **Enter or select parent folder** field, make sure it refers to `hello_world_0`.
- In the **File name** field, type the name of the new file, for example `hello_world.c`. Note that you must specify the extension `.c`!

3. Click **Finish** to continue.

The new file `hello_world.c` is created and ready for editing in the editor view.

Add an existing file to your project (import)

Import a file

Instead of creating a new file, it is also possible to import an existing file into your project or to create a file directly in the `hello_world_0` folder. To demonstrate this, follow the steps below. Do not close Nios II IDE.

- First create a C source file (`existing.c`) with a standard editor outside Nios II IDE. (As content you can, for example, use a single line containing comments only).
- You can store the file anywhere on your hard disk, but *not* in your project folder. (For example in `c:\`)

In Nios II IDE, follow the following steps to import the existing file:

1. In the C/C++ view, right-click on the project `hello_world_0` and select **Import...**

The Import wizard appears.

2. Select **File System**. Click **Next** to continue.
3. In the **From directory** field, type the path to the directory where you saved `existing.c` (For example `c:\`) and click in the empty white box below.

The left box shows the file structure of the directory the right box shows the files located in that directory, similar to the windows explorer.

4. In the right box, select check the file `existing.c`.
5. Click **Finish** to finish the wizard and import the file into your project.

The file `existing.c` is copied from its location at `c:\` into your project folder and added to your project. It is now visible as C source file in the C/C++ view. Changes you make to *this* file, will not affect the original file stored in `c:\`. Also, removing this file from your project will remove the file also from your project folder, but the original file remains untouched.

Create a file in the project folder

Instead of importing a file, you can store the file `existing.c` directly in the `hello_world_0` folder. To add the file to your project:

- In the C/++ view, right-click on `hello_world_0` and select **Refresh**.

The file `existing.c` should now be visible as part of your project.

Remove a file from your project

As we do not need this file for the remainder of this tutorial, we can safely remove it again from the project:

- In the C/C++ view, right-click on the file `existing.c` and select **Delete**.

The file `existing.c` is no longer part of your project and has been removed from your project folder.



Be aware that when you remove a file from your project, it always will be removed from its location in the project folder on your hard disk too!

2.4 Editing Files: C/C++ Editor

Editing a file

Enter the following simple C source in your new source document `hello_world.c`:

```
#include <stdio.h>

void main( void )
{
    printf( "Hello world" );
}
```

Note the following:

- The tab label of the editor view shows an asterisk in front of the file name (*hello_world.c) to indicate that the file has been modified.
- The C/C++ editor view uses syntax coloring.
- The Outline view shows the structure of the file. You can use this view to navigate through (larger) source files easily. Alternatively you can expand the structure of the file in the C/C++ project view.
- Right-clicking in the editor view presents you with a list of menu commands.
- To receive more help about the editor view, make sure it is active and press **F1**.

Saving and closing a file

To save the file:

1. From the **File** menu select **Save** (Ctrl+S)

To close the file:

2. From the **File** menu select **Close** (Ctrl+F4)

Nios II IDE will ask you to save the files that have been modified since the last save.



Notice also the menu commands **Save All** and **Close All** which you can use when you are working with multiple files.

Opening a file in the C/C++ editor

There are several ways to open an existing file. An easy way to open the C source file `hello_world.c` directly in the C/C++ editor is:

- In the C/C++ view, double-click on the file name.

Nios II IDE recognizes the file as a C source file and opens the file in the C/C++ editor.



If you want to open a C source file in a standard text editor (instead of the C/C++ editor), proceed as follows:

- In the C/C++ view, right-click on the file `hello_world.c` and select **Open With » Text Editor**.

The file opens in a text editor. Notice that syntax coloring is not available now and that the Outline view does not show the file structure because the file is not being edited as a C source file.

You can even open the file in an application (editor) outside Nios II IDE:

- In the C/C++ view, right-click on the file `hello_world.c` and select **Open With » System Editor**.

The file opens in the application that is associated with the file extension `.c` in windows.

2.5 Closing and Opening a Project

Closing a project

Like files, you can close a complete project. To do so:

1. In the C/C++ view, right-click on the project `hello_world_0` and select **Close Project**.

If there are unsaved files, the Save Resources dialog appears in which you can choose which modified files need to be saved before closing the project.

2. Select the files you want to be saved and click **OK** to continue.

Any selected unsaved files are saved first, then the project closes. In the C/C++ view the project `hello_world_0` is now visible as a closed map.

Opening a project

To reopen the project again:

- In the C/C++ view, right-click on the project `hello_world_0` and select **Open Project**.

The project is open for modifications again. You may need to expand the project structure to view its contents.

2.6 Setting Project Options

Now you are familiar with opening and editing (files in) your project, we will have a look at the options you can set for building your project.

First make sure the project `hello_world_0` is open.

To access the options for your project:

1. From the **Project** menu select **Properties**.

The Properties for `hello_world_0` dialog appears.

2. If not selected, select **C/C++ Build** to access the TASKING toolset for Nios II options.

The following screen should now appear.

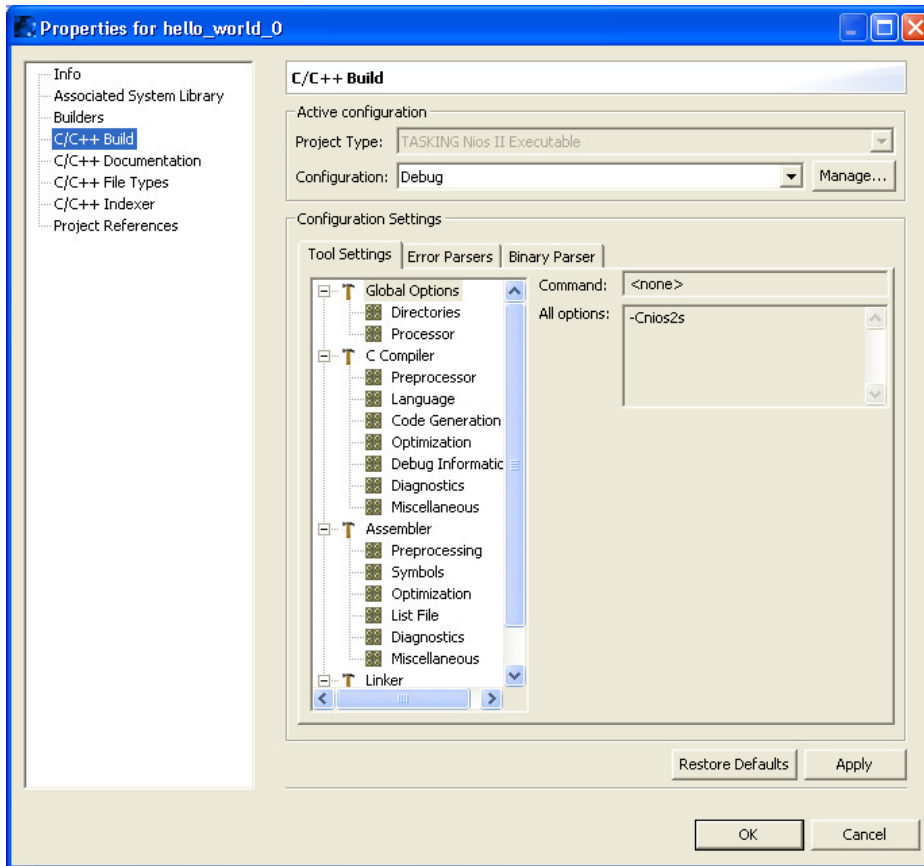


Figure 2-1: Properties for <Project> dialog

On the **Tool Settings** tab, the options are grouped in Global Options, C compiler, Assembler and Linker. The **Project Type** field shows which type of file will be built. Below you can choose a Configuration.



Note that the linker and/or archiver options are for the GNU tools. The TASKING linker and archiver are not used. Also note that the compiler and assembler options are for the TASKING C compiler and assembler. If you create hand-coded assembly for the TASKING assembler, use the extension `.asm` or `.s.c`.



For a detailed description of all TASKING toolset options refer to the manual *Nios II Embedded Tools Reference* (TR0130).

Selecting a predefined configuration

A configuration is a predefined set of options. You can choose between the **Debug** and the **Release** configuration. Both have their own setting. Check this for your self:

1. Select the **Release** configuration.
*The option **Generate symbolic debug information** is set to **None**.*
3. Select the **Debug** configuration.
*The option **Generate symbolic debug information** is set to **Default (-g)***

Setting options and restoring defaults

You can use one of the available configurations as a starting point for setting your options. For now, choose the Debug configuration.

1. Change the option **Default (-g)** to **Full (-Wc-ga)**
At this point you can change as many options as you like.
2. Click **Apply** to apply the new setting(s) to your project.
The dialog does not close, but the new options are saved to the Debug configuration.

To restore to the default Debug configuration options:

3. Click the **Restore Defaults** button.
The option settings are changed to the default settings of the chosen configuration.
4. Click **Apply** to apply the default settings to your project.



If you change options without applying them and you try to change the configuration, you are asked whether to apply the changes first.

Creating your own configuration

Because of the amount of possible options, it may be very convenient to create your own configuration.

1. Click on the **Manage...** button.
The Manage dialog appears.
2. Click on the **New...** button.
The Create configuration dialog appears.
3. Type a **Name** (Myconfig) and optional a **Description** for your configuration.
*In the **Copy settings from** box, you can choose the initial option settings for your configuration:*

4. Select **Existing configuration** and choose the **Debug** configuration.

The existing Debug configuration is the same as the default Debug configuration because we applied the default settings in the previous example.

5. Click **OK**.

The Manage dialog shows the new configuration.

6. Click **OK**.

Your new configuration has become the active configuration. From now on, all changes in options settings you apply, are saved in the `Myconfig` configuration.



Important: the **Restore Defaults** button is still associated with the default Debug configuration! Remember that the new configuration `Myconfig` is based on the Debug configuration.

Creating your own defaults

The previous example showed how to create your own configuration to store settings. However, it was impossible to return to your own defaults, only the original **Debug** and/or original **Release** defaults were available. Below it is described how you can create your own defaults:

If you have found a satisfying combination of option settings, you can create a configuration named `Mydefaults`.

- First change the option settings to your own needs.
- Repeat steps 1 through 6 of *Creating your own configuration* but in step 3, type the name `Mydefaults`.

Normally, any settings you change from here, are saved to `Mydefaults`, thus losing your original defaults. To prevent this:

- Repeat steps 1 through 3 of *Creating your own configuration*, to create a second new configuration and name it `Myworkoptions`. The name suggests that this will be the configuration for experimentally changing option settings.

4. Select **Copy settings from Existing configuration** and choose the **Mydefaults** configuration.

Your new 'working' configuration is now the same as the configuration named Mydefaults.

5. Click **OK**.

The Manage dialog shows the new configuration.

6. Click **OK**.

Now you can work with the `Myworkoptions` configuration. If you want to return to your defaults, you can either make the `Mydefaults` configuration active, or create a new configuration using the `Mydefaults` configuration to copy the settings from.

2.7 Build a Project

When you build a TASKING Nios II project in the Nios II IDE, the TASKING Nios II compiler and assembler and GNU linker are used to compile and link all the source code and the libraries associated with the project.

To build a project:

- In the C/C++ Projects view, right-click on the name of the project, `hello_world_0` (not `hello_world_0_system`), and select **Rebuild Project**.

This first builds the `hello_world_0_syslib` project and then the `hello_world_0` project.

From the **Project** menu, the following "Build" commands are available:

Build All...	Builds all projects in the active workspace.
Build Project...	Builds the active project.
Build Working Set »	Opens a wizard in which you can create a customized set of files that will be build.
Clean...	Removes all intermediate files that are created during a build. As a consequence, the next build cannot rely on existing results from previous builds (thus simulating a rebuild).

2.8 What's Next?

Now you have created and built a TASKING Nios II project you can run and debug the project in Nios II IDE. This is described in detail in the Nios II IDE Help, section *Running and Debugging Projects*.