

1 Help Contents



Impulse C CoDeveloper™ from Impulse Accelerated Technologies™ Pico Computing Platform Support Package



CONTENTS

[Before You Begin \(Read This First\)](#)

[Pico Platform Support Package Overview](#)

[About Pico FPGA Cards](#)

[Programming for Pico Hardware](#)

[Using the Pico PSP](#)

[Quick Start Tutorial](#)

1.1 Pico Platform Support Package Overview



Welcome to CoDeveloper™ for Pico Computing FPGA Cards™



Welcome to CoDeveloper for Pico Computing FPGA Cards: advanced software technologies enabling FPGA-acceleration of high-performance computing applications. The CoDeveloper tools allow you to create FPGA hardware in the form of synthesizable hardware description language (HDL) files and related hardware and software interface logic. CoDeveloper is capable of generating hardware described by one or more hardware processes written in an ANSI C. These processes communicate with other hardware and software processes using methods that include streaming, message passing, and shared memory.

The complete CoDeveloper environment consists of a set of libraries that allow Impulse C applications to be compiled by a standard C/C++ desktop compiler (for simulation and debugging purposes) as well as cross-compiler and translation tools that allow mixed hardware/software applications to be implemented on selected programmable hardware platforms.

The Platform Support Package described in this document extends the power of CoDeveloper by providing automated software/hardware generation for FPGA accelerators that execute on supported Pico E-14 and E-16 cards. The combination of CoDeveloper and the Pico Platform Support Package (PSP) enables a software application engineer to create FPGA accelerators from Impulse C source code alone. Typically, no user-supplied HDL is required. CoDeveloper for Pico is also highly integrated with Xilinx' ISE synthesis/place-and-route software, making CoDeveloper easy to use for first-time users of FPGAs.

Platform Support Package Highlights

- Supports the E-14 and E-16 cards from Pico Computing
- Supports the Impulse C stream communication mechanism
- Supports IEEE 754 compliant single- and double-precision floating-point arithmetic, through Xilinx CoreGen libraries
- Provides an automated hardware and software flow. When an Impulse C project is exported, all HDL for the FPGA module is produced. In addition, scripts are created that run the synthesis/place-and-route process for the chosen hardware device, as well as a Makefile for building software.

Getting Started

To get started using CoDeveloper for Pico, you should load one of the [tutorial sample projects](#) described in this document. These examples will help you understand how to create and manage an Impulse C project in conjunction with the software tools provided by Xilinx and Pico.

To load a sample project, start the CoDeveloper Application Manager and click the "Sample Projects" tab in the Start Page (View > Start Page menu), then click the name of a sample project to open it in CoDeveloper.

If using the Eclipse-based CoDeveloper Application Manager, load a sample project by opening the Welcome page (Help > Welcome menu) and clicking the "Browse Impulse C Samples" link. Click the name of any of the sample projects listed on the Impulse C Sample Projects page to copy the sample

to your workspace.

Before starting with a CoDeveloper example project, be sure you have reviewed the information in the [Read This First](#) section of this document. Also, please take time to familiarize yourself with the documentation for your Pico card.

See Also

[Help Contents](#)
[Read This First](#)
[About Pico FPGA Cards](#)
[Quick Start Tutorial](#)

1.1.1 Read This First

Release Notes

This is release 3.00 of **CoDeveloper for Pico Platform Support Package**, from Impulse Accelerated Technologies. This PSP is designed for the Pico E-14 and E-16 cards. E-12 cards are supported by the "Pico E-12 LO (VHDL)" and "Pico E-12 EP (VHDL)" PSPs, which are not described in this document.

System Requirements

Applications for the Pico card may be developed on one system and run on another host computer.

The requirements for the development system are:

- x86 PC with 512 MB of RAM (1 GB recommended for place and route)
- Windows 2000 or Windows XP
- Impulse CoDeveloper 3.00 or later
- Xilinx ISE 8.2i or later, licensed and installed
- Optional: Third-party synthesis software supporting Xilinx FPGAs
- Optional: Third-party HDL simulation software

The requirements for the host computer are:

- x86 PC with 512 MB RAM
- Linux-based systems: Fedora Core 5 (other distributions will likely work, but have not been specifically tested)
- Windows-based systems: Windows 2000 or Windows XP
- Pico E-14 or E-16 FPGA card
- Pico driver and PicoUtil software, 4.0.1.0 or later
- GNU g++ compiler, 3.4.2 or later (included with CoDeveloper 3.00 for Windows)

To target any of the supported Pico FPGA cards, the user simply selects the "Pico E-14/E-16 (VHDL)" platform from the drop-down list in the Generate tab of the Project Options window as described in [Programming for Pico Hardware](#). After exporting the hardware files generated by the Impulse C compiler, the user will select the appropriate script to build a hardware bitfile for the chosen card.

Installation Notes

The Pico PSP assumes that the Pico drivers and software have been properly installed. Specifically, the PSP-generated Makefile references software driver source files in the Pico installation directory. This version of the PSP has been tested with version 4.0.1.0 of the Pico software tools.

In order for the Makefile-driven software build to function properly, the PICOBASE environment variable must be set as indicated:

```
PICOBASE = C:\Pico
```

Furthermore, the "bin" subdirectory of PICOBASE (e.g., "C:\Pico\bin") should be added to the PATH, so the Pico-provided hardware generation script can be run from the command line. The Pico software installation normally adds this directory to the PATH.

Similarly, the path to the Xilinx ISE tools (xst, ngdbuild, map, par, bitgen, and data2mem) must be included in the system PATH so these tools can be executed by the Pico build script.

Limitations

This is version 3.00 of the Pico PSP. This version of the PSP has the following limitations:

- Only co_streams are supported; co_memory, co_signal, co_register, and co_semaphore are not
- Only one concurrent software process on the target is supported
- The maximum stream data width supported is 32 bits
- Dual clocks are not supported
- Active-low reset is not supported
- The automatically generated Makefile assumes g++ is the compiler installed on the development system

Check the [Impulse C website](#) periodically for news about Pico PSP updates.

Prerequisites

The tutorials in this Platform Support Package also assume that you have read and understand the introductory sections of the CoDeveloper User's Guide, which is installed with your CoDeveloper product and can be accessed from the CoDeveloper Help menu. In particular, you should take the time to go through the tutorials provided with CoDeveloper so you have a good understanding of the front-end design flow including both desktop simulation and hardware compilation.

Note: The tutorials provided in this document are intended for illustrative purposes only. The steps required to use CoDeveloper with the Pico FPGA cards may differ somewhat from the steps described in these tutorials. Please take the time to familiarize yourself with the Pico documentation prior to beginning.

Getting Started

A sample project suitable for the E-14 or E-16 platform is included with the Platform Support Package. To work with this example, please follow the [tutorial](#) included in this document.

Once you have opened the sample project, you can review the Impulse C source code, launch a desktop simulation or invoke the Impulse C hardware generator to process the sample application and generate output files compatible with a Pico FPGA card. These steps are fully described in the [Quick Start Tutorial](#).

See Also

[Quick Start Tutorial](#)

1.1.2 Contents



Impulse C CoDeveloper™ from Impulse Accelerated Technologies™ Pico Computing Platform Support Package



CONTENTS

[Before You Begin \(Read This First\)](#)

[Pico Platform Support Package Overview](#)

[About Pico FPGA Cards](#)

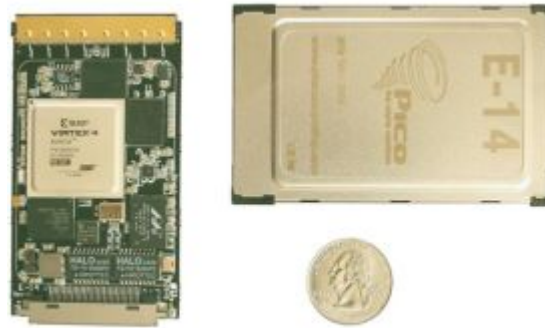
[Programming for Pico Hardware](#)

[Using the Pico PSP](#)

[Quick Start Tutorial](#)

1.2 About Pico FPGA Cards

E-14 Card Overview



Pico's E-14 card offers an extremely powerful embedded system in a Cardbus package boasting a Xilinx Virtex-4 FX-20, FX-40 or FX-60 FPGA. The FPGA also includes an integrated 450 MHz PowerPC processor.

The E-14 also comes equipped with Gigabit Ethernet, 64 MB Flash ROM, 256 MB RAM, and a 32-bit Cardbus interface capable of speeds up to 1 Gbps making this tiny machine capable of accelerating many algorithms in hardware with the flexibility of software.

For more information about the Pico E-14 FPGA Card, see <http://picocomputing.com/products/cards/e14ep.php>.

E-16 Card Overview



Pico's E-16 is an extremely powerful co-processor designed for high performance computing in a ExpressCard/34 package with a Xilinx Virtex-5 LX-50.

The E-16 utilizes the PLX PEX8311 PCIe Bridge Chip, 32 MB PSRAM, and a x1 lane of PCIe interface capable of speeds up to 250 MB/s, making this tiny machine capable of accelerating many algorithms in hardware with the flexibility of software.

For more information about the Pico E-16 FPGA Card, see <http://picocomputing.com/products/cards/e16.php>.

See Also

[Programming for Pico Hardware](#)
[Using the Pico PSP](#)
[Quick Start Tutorial](#)

1.3 Programming for Pico Hardware

Note: The following discussions assume that you have a good understanding of programming using the Impulse C API functions for multiprocess communication. If you are not familiar with the concept of data streaming, please read the tutorials provided with your CoDeveloper installation.

Using Streams

The Pico PSP supports stream communication methods. Currently, streams between a hardware and a software process are limited to a maximum width of 32 bits. A maximum of 16 streams per process are supported by the Impulse C host software library.

Hardware/Software Communication Performance Notes

- A stream between a software process running on the host and a hardware process running on an E-14 (CardBus interface) yields about ??? MB/sec of bandwidth.
- A stream between a software process running on the host and a hardware process running on an E-16 (PCIe x1 interface) yields about 150 MB/sec of bandwidth (half-duplex)

Using Floating-Point Arithmetic

The Pico PSP includes support for single- and double-precision IEEE 754 compliant floating-point arithmetic, through instantiation of Xilinx CoreGen components. Floating-point hardware components are inferred automatically from arithmetic operations on the float or double ANSI C types.

To use floating-point in your application, select the "Include floating-point library" and, if applicable, the "Allow double-precision types and operators" options (in CoDeveloper: Project > Options menu, Generate tab).

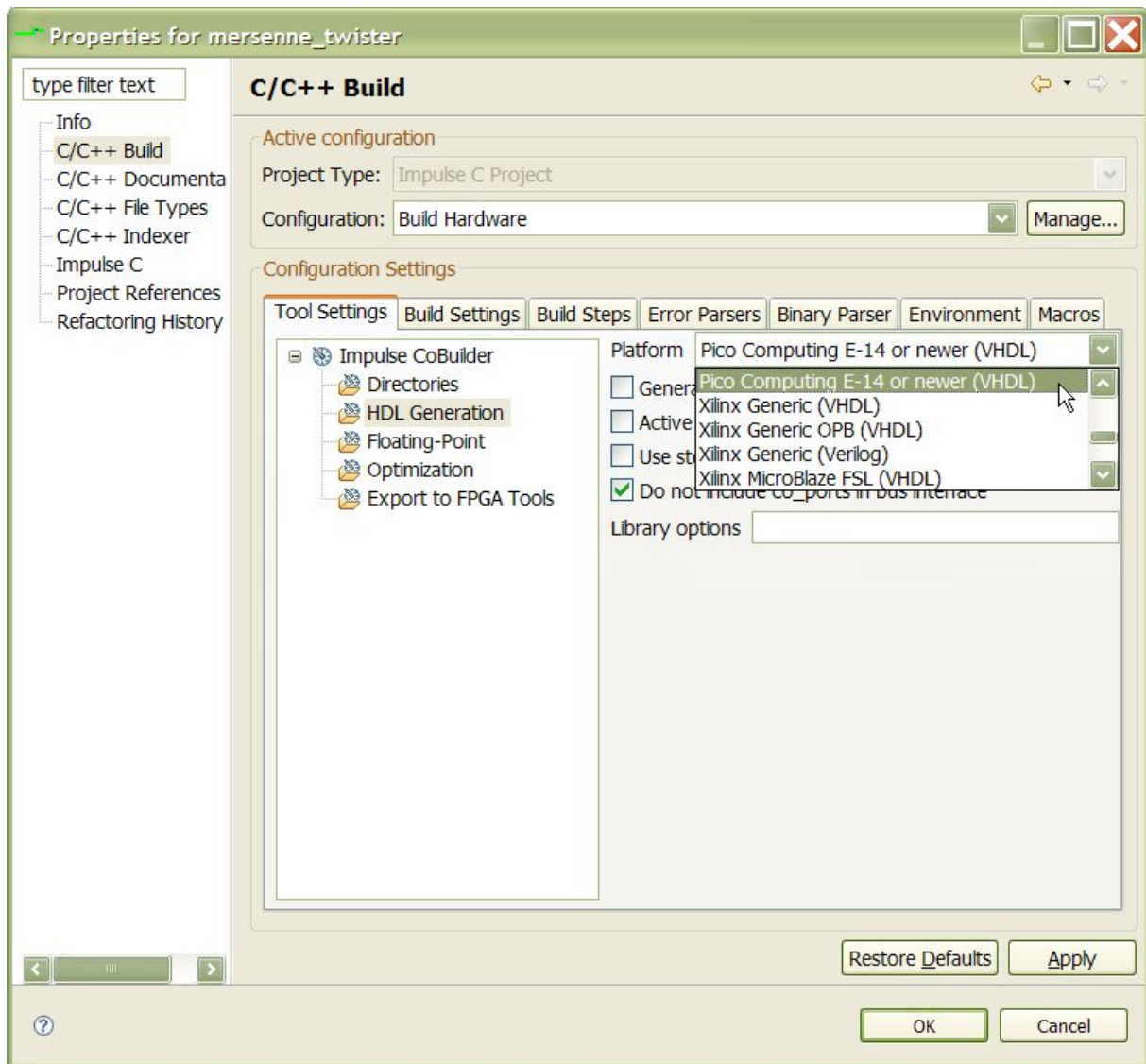
See Also

[Using the Pico PSP](#)

1.4 Using the Pico PSP

Targeting an Application to a Pico FPGA Card

Assuming an Impulse C project has been created and opened, the Pico PSP ("Pico E-14/E-16 (VHDL)") can be targeted by selecting it in the project's Properties for the Build Hardware configuration, as shown below:



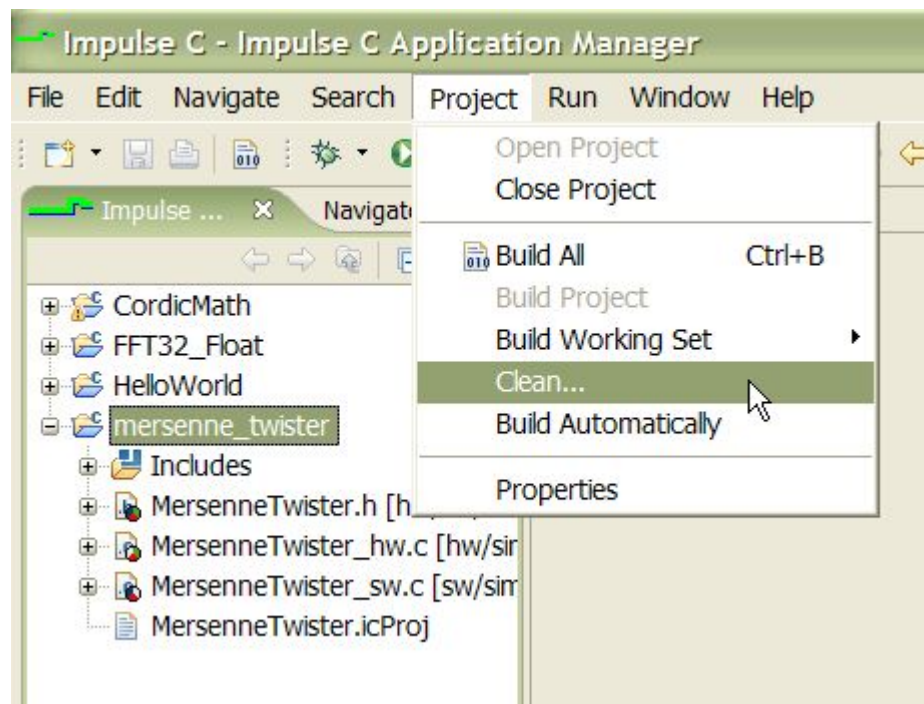
Several Build Hardware configuration options are not supported by the current PSP, including the "Generate dual clocks" option, the "Active-low reset" option, and the "Use std_logic types for VHDL interfaces" option.

Developing a Project for a Pico FPGA Card

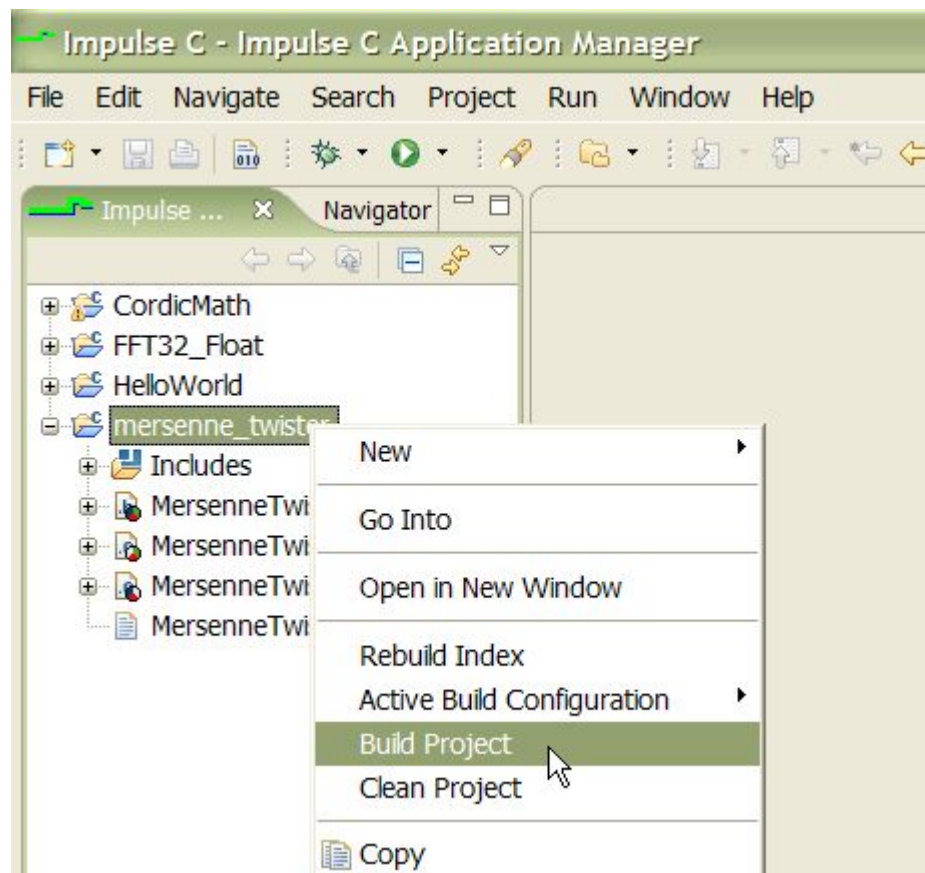
Please note the [System Requirements](#) before proceeding. Errors may occur when building hardware or software if the development system is not properly configured.

The Pico PSP is designed to help automate the task of producing an FPGA accelerator for the E-14 or E-16, reducing the development time otherwise required. The following sequence of steps is recommended when creating applications using Impulse C and the Pico PSP. The following assumes the project has been created and the Impulse C application has been written. With this in mind, the recommended compilation sequence is:

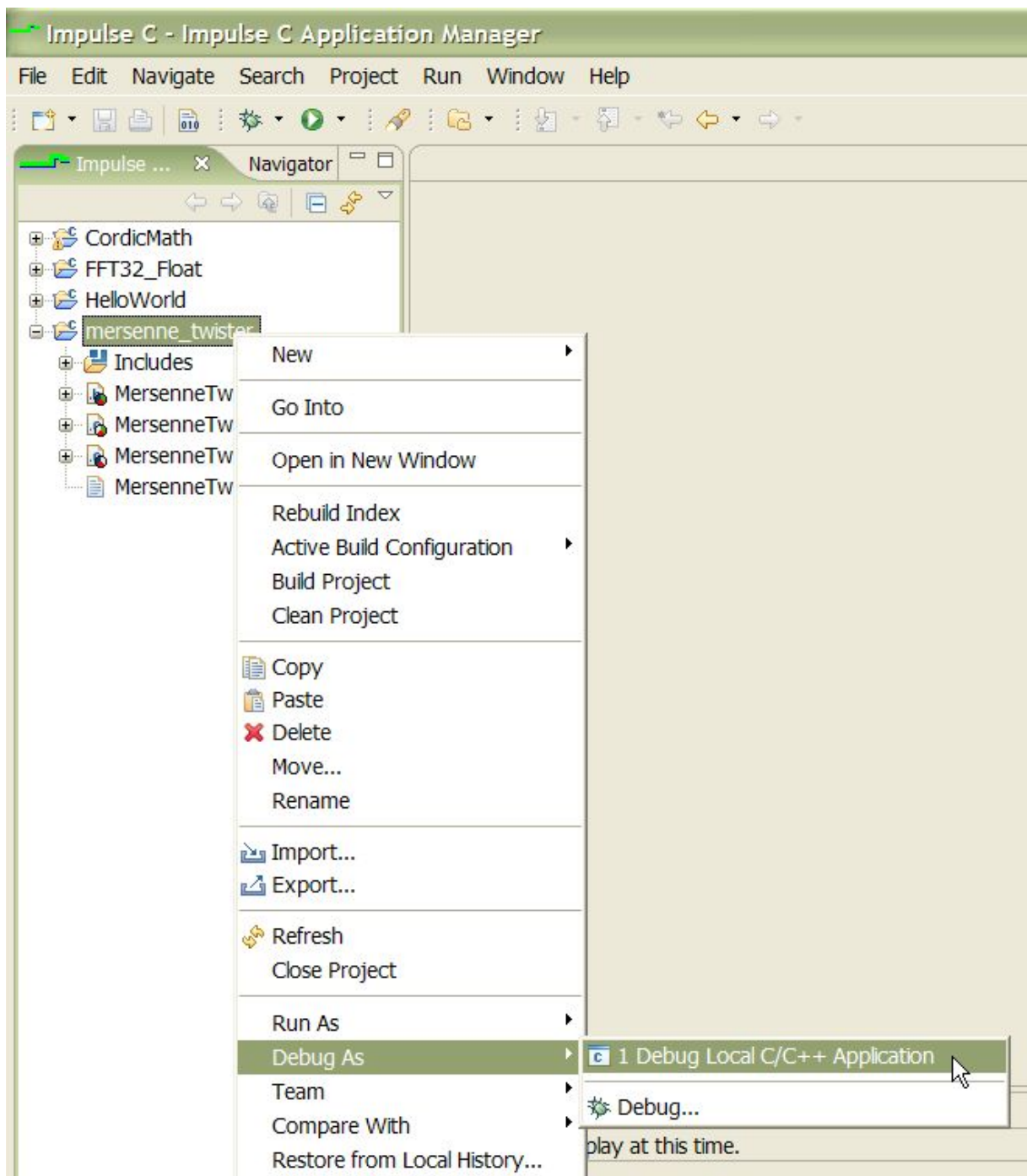
1. Remove any previous compilation results by cleaning the project. Select Clean from the Project menu to do this:



2. Compile the entire project to run on the CPU in a desktop simulation. To do this, right-click the project in the Impulse C Project view and select Active Build Configuration > DesktopSimulationDebug from the menu, then right-click again and select Build Project:



3. Execute and debug the software simulation executable. The simulation executable can be run for debugging from within the CoDeveloper GUI by right-clicking the project and selecting Debug As > Local C/C++ Application, as shown below:



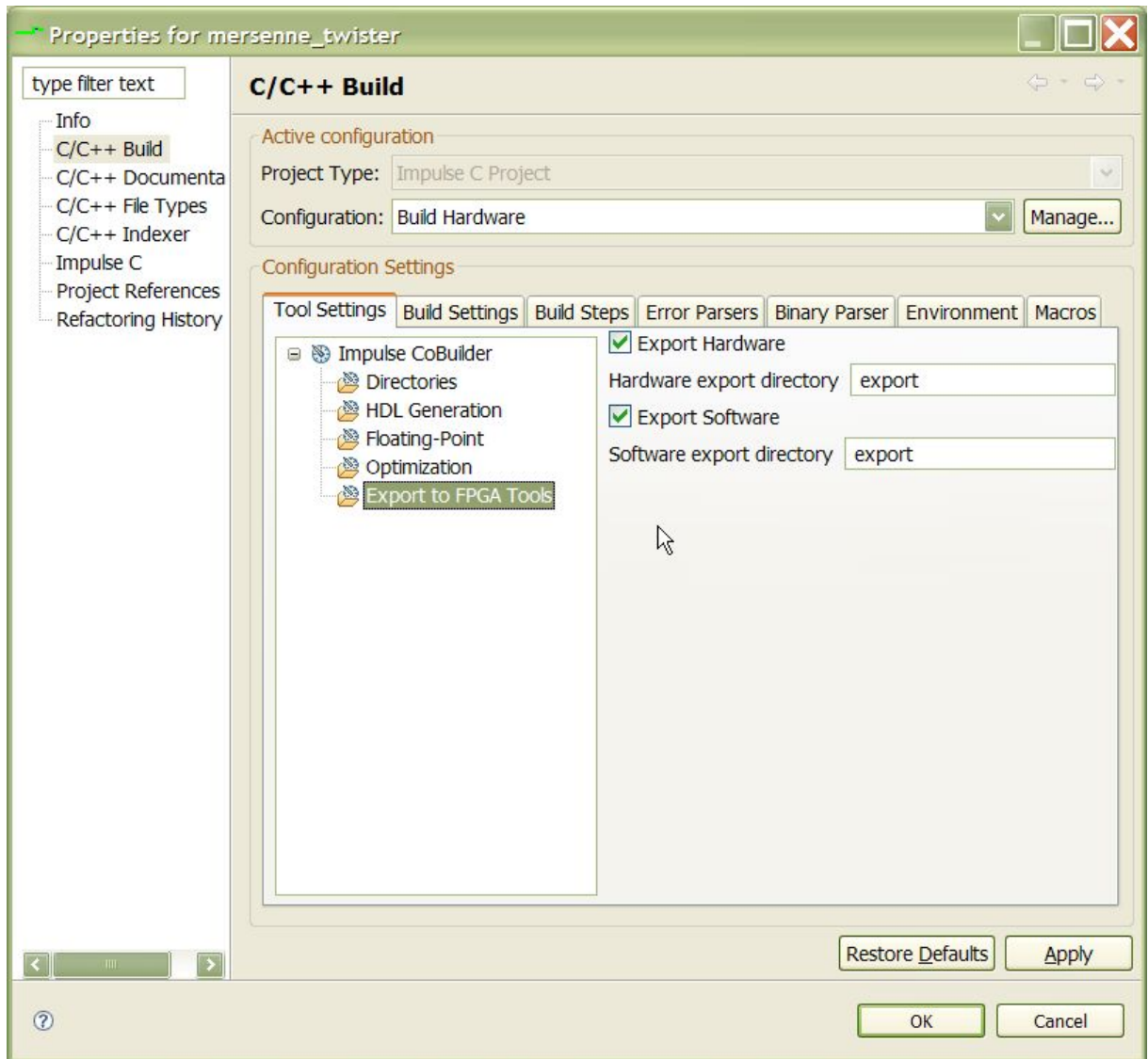
Alternatively, the software simulation executable can be launched from the command line (e.g., Windows Command Prompt or bash). After the application has been debugged and runs correctly in software simulation on the desktop, go to the next step.

4. Generate HDL. To do this, right-click the project and select Active Build Configuration > BuildHardware from the menu, then right-click again and select Build Project.

In this step, hardware processes described in Impulse C are compiled to VHDL. After this step has been performed, the Stage Master Explorer tool can be used to graphically view the results

of the compilation. (Please refer to the CoDeveloper documentation for more information about Stage Master Explorer.)

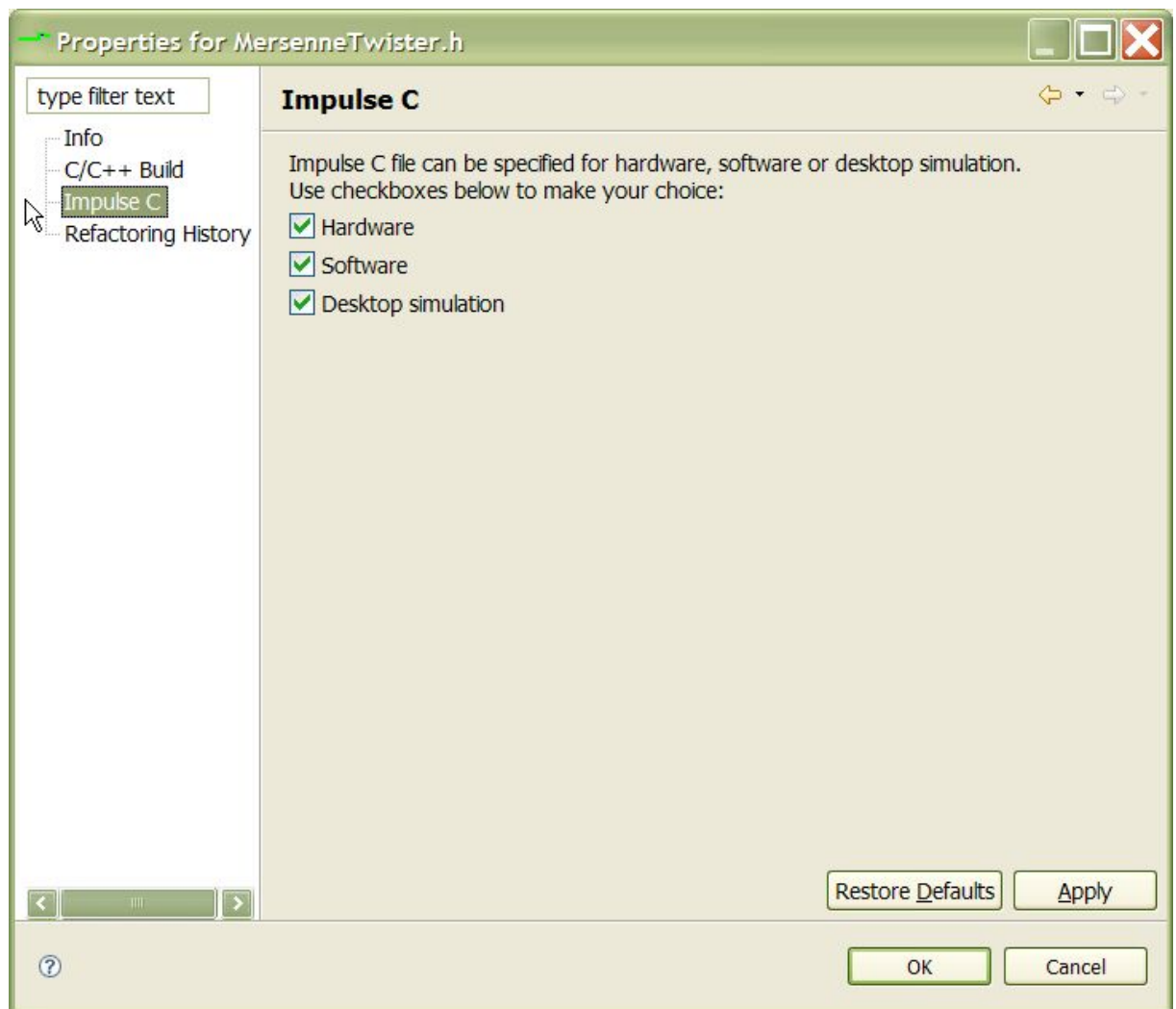
As part of the hardware generation process, the generated files will also be exported by default. The option to export files can be toggled in the Properties dialog, as shown below. While it is possible to import the HDL files produced by the HDL generation step into an ISE project and integrate the hardware interfaces manually, the Pico PSP can automate this task for you through the export options:



The Export Hardware option creates project files for use with ISE and the Pico tools that integrate the Impulse C user logic with the Pico card's CardBus or PCIe interface. The project files also specify preset constraints provided by Pico for each card model. These generated files will appear in the "Hardware export directory".

The Export Software option creates the "Software export directory" under the project. Within this subdirectory can be found all of the software source code for the project, along with an automatically generated Makefile. The Makefile is rudimentary, however, and may need some manual editing for complex software projects. Make sure source and include file used to build

the host application program is marked as an Impulse C "Software" file in the Properties dialog (accessible by right-clicking the file and selecting Properties > Impulse C):



7. Execute synthesis and place and route of the FPGA. Producing a binary programming file (bitfile) that can be downloaded to the FPGA is as simple as invoking a Pico script (**build-pico-fw.bat**, installed in %PICOBASE%\bin), passing it the name of the generated project file appropriate to your chosen Pico card model. For example, to generate a bitfile for the E-14 FX60:

```
C:\MyPicoProject\export>build-pico-fw e14fx60.fwproj
```

The synthesis and place and route process can take anywhere from 15 minutes to several hours to complete, depending on the utilization of the FPGA on the Pico card and the capabilities of your development machine.

8. Compile Impulse C target software. First, copy the entire software export subdirectory (e.g., "export/software") to the system with a Pico card. On the target system, go to the software subdirectory just created and from a command line shell type `make`. This step assumes that the g++ compiler and associated tools are installed on the target system.
9. Download FPGA programming file. Using the PicoUtil software, upload the generated bitfile to

the Pico card. See the Pico documentation for details.

10. Execute the application. From the development system's target software directory, execute the application on the command line.

See Also

[Quick Start Tutorial](#)

1.5 Quick Start Tutorials



Impulse C Tutorials for Pico FPGA Cards

[Tutorial 1: Creating an FPGA Accelerator](#)

1.5.1 Tutorial 1: Creating an FPGA Accelerator



Overview

This tutorial will demonstrate how to generate a complete example for the Pico E-14 or E-16.

The application used in this tutorial is a Mersenne Twister random number generator.

The steps involved in generating hardware and software for the Pico card are illustrated in this tutorial using the CoDeveloper Application Manager (Eclipse IDE); if using the non-Eclipse version of the CoDeveloper Application Manager, the steps are similar, but the menus and commands are different.

This tutorial will require approximately 45 minutes to complete, including software run times.

TO BE COMPLETED

Steps

[Loading the Impulse C Project](#)
[Building the Application for the Target Platform](#)
[Generating the FPGA Bitfile](#)
[Downloading the Bitfile](#)
[Running the Application](#)

1.5.1.1 Loading the Impulse C Project

Mersenne Twister Tutorial for the E-14 and E-16, Step 1

To begin, start the Impulse CoDeveloper Application Manager (Eclipse IDE) by double-clicking the desktop icon, or selecting "CoDeveloper Application Manager (Eclipse IDE)" from the Start > All Programs > Impulse Accelerated Technologies > CoDeveloper x.yy program group.

Note: This tutorial assumes that you understand the concepts presented in the "HelloWorld" tutorial, found in the CoDeveloper User's Guide.

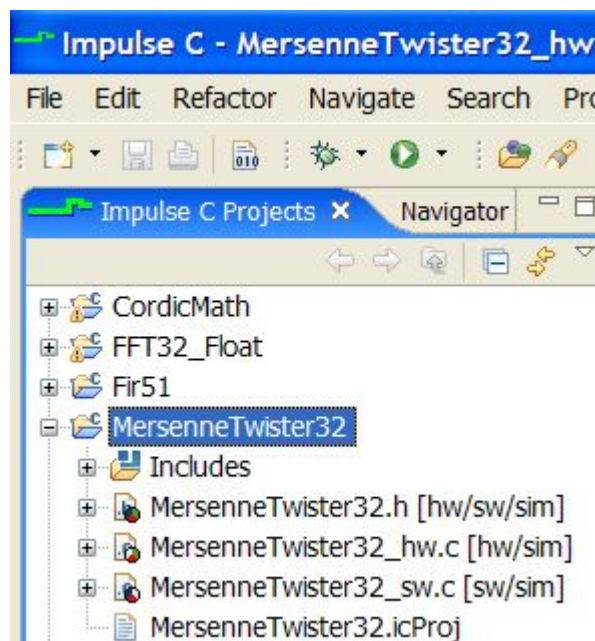
Copy the MersenneTwister32 sample project's folder into your CoDeveloper Eclipse IDE workspace. By default, this is the "workspace" subdirectory of the CoDeveloper installation. The sample project is found in the Examples\Scientific\MersenneTwister32 subdirectory of the CoDeveloper installation.

Create a new project from the files you just copied. First select the File > New > Project menu. The New Project wizard will appear.

Select "Impulse C Project" and click Next.

Enter the project name "MersenneTwister32". This name must be the same as the folder you copied into the workspace. Click Finish to create the project, importing the settings and files from the sample.

Click the project in the Impulse C Projects view to expand the source file tree:



The **MersenneTwister32** project includes three source files:

- **MersenneTwister32.h**: Constants used in both *.c files that define stream parameters and parameters of the Mersenne Twister algorithm.
- **MersenneTwister32_hw.c**: Two hardware processes that implement the Mersenne Twister, along with an Impulse C configuration function that defines the application's processes and their stream connections.
- **MersenneTwister32_sw.c**: Software process for testing the hardware design, as well as a main() function for running the complete application.

The **MersenneTwister32.icProj** file also shown here is a legacy project file from CoDeveloper Version 2. When the project was copied into your workspace from CoDeveloper's Examples directory, settings from this project file were imported into the CoDeveloper environment. Any changes made to the .icProj file from now on will be out of sync with the project as it's stored in CoDeveloper Eclipse IDE. We therefore recommend not editing the .icProj file in your workspace, or opening it in the legacy Version 2 IDE.

Next

[Compiling the Application for Simulation](#)

1.5.1.2 Compiling the Application for Simulation

Mersenne Twister Tutorial for the E-14 and E-16, Step 2

Before compiling a desktop simulation of the Mersenne Twister design, let's examine how the design is structured as an Impulse C application.

The Mersenne Twister pseudorandom number generator algorithm involves two parts: a generalized feedback shift register and a tempering transform. Each generated random value is passed to the tempering process to improve the equidistribution of the bits in the number. These two parts are natural candidates for implementation as Impulse C processes, as they produce one output at a time and are cascaded with each other. We can use a stream to connect such processes. The two processes are implemented in Impulse C as the **genrand** and **temper** processes, in the source file **MersenneTwister32_hw.c**. These processes, which perform the algorithm's critical computation, are defined as hardware processes so they can be optimized and compiled as HDL for the FPGA.

Two more processes are defined in **MersenneTwister32_sw.c**, to send seed data to the **genrand** process and receive the generated random numbers from the **temper** process. These processes, **Producer** and **Consumer**, are designed to run as a software testbench in either desktop simulation or on the host system containing the Pico FPGA card. Streams connect the software test process to the hardware processes that will run in the FPGA hardware.

The hardware source file, **MersenneTwister32_hw.c**, contains the Impulse C configuration function, **config_MersenneTwister32**, which creates all the processes and their I/O connections.

Next

[Building the Application for the Target Platform](#)

1.5.1.3 Building the Application for the Target Platform

Mersenne Twister Tutorial for the E-14 and E-16, Step 3

Specifying the Platform Support Package

The next step, prior to compiling and generating the HDL and related output files, is to select a platform target. Which target you select has a number of implications, including:

- The output file format (e.g., VHDL, Verilog, or other intermediate language)
- The primitive components that will be referenced in the output (e.g., memory components or buffer types)
- The types of optimizations performed during compilation
- The software library components that will be generated

- The type of I/O to be generated for the input and output streams

To specify a platform target, open the Project Properties dialog by selecting the MersenneTwister32 project in the Impulse C Projects window, then selecting the Projects > Properties menu item (or by right-clicking the MersenneTwister32 project and selecting Properties):



Select the C/C++ Build section from the list on the left. In the Configuration drop-down list, choose "Hardware", as shown:



(You may be prompted to save changes to the current configuration, even if you have not made changes. This is a known bug. You may click No if you are sure you have not changed the current configuration.)

In the Tool Settings tab, select the HDL Generation page, under Impulse CoBuilder. From the Platform drop-down-list, make sure "Pico E-14/E-16 (VHDL)" is chosen, as shown:



Some options are not supported by this PSP and should not be selected. In particular, do not select the "Generate dual clocks" or "Active-low reset" options. Click OK to save the options and exit the dialog.

Generate HDL for the Hardware Processes

To generate hardware in the form of HDL files, and to generate the associated software interfaces and library files, right-click the MersenneTwister32 project and select "Hardware" from the Active Build Configuration menu item. From the same right-click menu, select Build. A series of processing steps will run and print output to the Console window.

When processing is complete you will have a number of resulting files created in the **Hardware\hw** subdirectory of your project directory. Take a moment to review these generated files. They include:

- Generated VHDL source files (**MersenneTwister32_comp.vhd** and **MersenneTwister32_top.vhd**) representing the hardware processes and the generated hardware stream interfaces
- Additional Verilog source file (**pcore\picobus_impulse.v**) representing the interface between streams and the CardBus (E-14) or PCIe (E-16) interface
- A **lib** subdirectory containing required VHDL library elements.

In the next step, we will describe how to compile these generated files into an FPGA bitmap file compatible with the E-14 or E-16, using the Xilinx ISE tools.

Next

[Generating the FPGA Bitfile](#)

1.5.1.4 Generating the FPGA Bitfile

Mersenne Twister Tutorial for the E-14 and E-16, Step 4

Using tools from Pico Computing and Xilinx, you will now compile the VHDL and Verilog generated by the Impulse C compiler into a bitfile that can be used to program the FPGA device.

Export Generated Hardware

To make the process of building the bitfile easier, the "Pico E-14/E-16 (VHDL)" PSP can export the generated HDL files into a directory structure particular to the Pico and Xilinx tools. This export process also generates project files for the Pico and Xilinx tools. After exporting the generated hardware, the bitfile can be built using a few simple commands from the Windows Command Prompt.

Look in your project's **Hardware\export** subdirectory; if the following files are present, the generated hardware has already been exported and you may skip to the next section:

- **impulse_lib**: Directory containing Impulse HDL libraries, exported from **hw\lib**
- **e14fx20.fwproj**: Pico project file targeting the E-14 card with a Virtex-4 FX20 FPGA
- **e14fx60.fwproj**: Pico project file targeting the E-14 card with a Virtex-4 FX60 FPGA
- **e16lx50.fwproj**: Pico project file targeting the E-16 card with a Virtex-5 LX50 FPGA
- **ImpulseC.prj**: Xilinx ISE project file
- **MersenneTwister32_comp.vhd**: Exported from **hw**
- **MersenneTwister32_top.vhd**: Exported from **hw**
- **picobus_impulse.v**: Bus interface wrapper, exported from **hw\pcore**

By default, the HDL files are automatically exported whenever you build the project's "Generate Hardware" configuration in the CoDeveloper Eclipse IDE. If you do not see the exported files, make sure the "Export generated hardware" and "Export generated software" options are both selected in the Project Options dialog.

Generate the Bitfile

Open a Command Prompt window (Start > Run > "cmd"). Navigate to the MersenneTwister32 project's **export** subdirectory.

Run the Pico build script, passing it the name of the ".fwproj" file that is appropriate to the Pico card you're using. If the command is not found, make sure the **bin** subdirectory of your Pico installation is in your PATH (e.g., **C:\Pico\bin**). For example, to build a bitfile for the Pico E-14 with an FX60 FPGA, run the command:

```
build-pico-fw e14fx60.fwproj
```

See Also

[Downloading the Bitfile](#)

1.5.1.5 Downloading the Bitfile

Mersenne Twister Tutorial for the E-14 and E-16, Step 5

See the Pico tools documentation for instructions on how to download the bitfile to the Pico card.

Next

[Running the Application](#)

1.5.1.6 Running the Application

Mersenne Twister Tutorial for the E-14 and E-16, Step 6

At this point, if you have followed the tutorial steps carefully you have:

- Generated VHDL hardware files from the CoDeveloper environment.
-
- ...

You are now ready to download the complete application to the target E-14 and E-16 platform.

First...