

Pico Computing Inc.



Building a Firmware Project using makefile-V4

Version 5.0.0.2. Mar 16th, 2010.

Pico Computing, Inc.
150 Nickerson, Suite 311
Seattle, WA, 98109-1634
(206) 283-2178
www.picocomputing.com

1 Overview

This document describes the mechanisms provide by Pico Computing to build a firmware or software module.

Other manuals in this help library can be located at [GuideToDocumentation.pdf](#)

NOTE: This link will access the pdf from the PicoComputing.com Website .

Updates to the Pico software and firmware can be obtained directly from the Pico Computing website (www.picoComputing.com).

The Software build process uses a makefile and the msys software to provide a Unix-like environment. The msys directory will have been installed under your c:\pico directory. The makefile uses the the standard design flow provided by Xilinx. This can be obtained at:

Xilinx Webpack (Free)

– Free account registration with Xilinx is required.

Web Address: http://www.xilinx.com/ise/logic_design_prod/webpack.htm

2 Installed Directory Structure

Firmware for the Pico E-12, e-14, E-15, and E-16 card is generated using the fundamental Xilinx command line tools **xst**, **ngdbuild**, **map**, **par**, **bitgen**, and **data2mem**. This manual describes the command line tools that can be used to invoke these tools.

The firmware is installed under the following directories:

c:/pico				base directory specified by environment variable PICOBASE
./msys				standard Linux tools including make utility required for tool chain
./firmware				firmware directory
	makefile-v4			standard makefile
	./src		common Verilog source files	
	edk_user_repository		standard library files	
	<model>		build.bat	
		./src	source Verilog files .v	source code for base Pico firmware
			bash .sh files	script files used by makefile-V4
		./projects	.fwproj files	parameters to makefile
			.bit files	result files from synthesis
			.log files	log of results from synthesis
			temporary directories	

The model will be one of the following

<model>

<code>picoE12</code>	Pico-E12
<code>PicoE14_15</code>	Pico E-14 and Pico E-15
<code>picoE16</code>	Pico E-16

`makefile-v4` is the makefile which is passed to the msys make utility to drive the work flow.

3 Synthesizing a project

There will typically be several `.fwproj` in the projects directory with one or more of the following suffices:

Category	Meaning
<code>ppc</code>	build ppc version of firmware and include monitor.elf file
<code>apu</code>	build apu interface into firmware
<code>edk</code>	build base firmware for EDK

The file `build.bat` is a simple shim for the file `pico-build-fw`

```
$(PICOBASE)/bin/pico-build-fw %1 %2 %3 %4 %5 %6 %7 %8 %9
|$(PICOBASE)/bin/tee -a %1.log
notepad %1.log
```

`$(PICOBASE)/bin/tee.exe` is a utility provided by Pico Computing which writes the input stream to STDOUT and a file simultaneously. It also converts `\n` to `\r\n` to make the output files compatible with the Ansi standard.

It is sufficient to enter `build <project name>`, however, other options include:

Parameter name	Description
<code>Build</code> <code><projectName><category>*</code>	Remove all temporary files and directories
<code>clean</code> <code>Build</code> <code><projectName><category>*</code>	Perform the XST step only
<code>xst</code> <code>Build</code> <code><projectName><category>*</code>	Perform the ngdbuild step only
<code>ngdbuild</code> <code>Build</code> <code><projectName><category>*</code>	Perform the map step only
<code>map</code> <code>Build</code> <code><projectName><category>*</code>	Perform the par step only
<code>par</code> <code>Build</code> <code><projectName><category>*</code>	Perform the bitgen step only
<code>bitgen</code> <code>Build</code> <code><projectName><category>*</code>	Display parameters for work flow but do not perform tasks
<code>dbg</code> <code>Build help</code>	Invokes this help manual

For example:

```
build E14fx20-ppc
```

This will invoke the entire tool chain including `xst`, `ngdbuild`, `map`, `par`, the compile step to build `monitor.elf` and `data2mem` which integrates the two binary files.

When it is finished the file `c:\pico\firmware\picoE14_15\E14fx20-ppc.bit` will contain the new FPGA bit file, including `monitor.elf`.

4 The .fwproj file

The files with the extension `.fwproj` are parameters to the build process. A typical `.fwproj` file is:

```
PICO_MODEL=E14FX20
ENABLE_PPC=y
```

This specifies that the firmware is being built for a Pico E-14 card with an FX20 FPGA, and the PPC should be included in the synthesis.

There are many other options that can be specified as defined in the following table:

<code>PICO_MODEL=E14FX20</code>	Specifies Pico card and FPGA
<code>ENABLE_PPC=y</code>	Includes PPC support
<code>ENABLE_RAM=y</code>	Includes multiport RAM interface
<code>ENABLE_PIC=y</code>	Includes programmable Interrupt Controller support
<code>PROJ=CBBase</code>	Highest level module name
<code>EXPORT_BIT_FILE=User.bit</code>	Name of output bit file
<code>USER_MODULE_NAME=UserModule</code>	Name of user module
<code>ENABLE_USER_PICOBUS=y</code>	Enable Pico Bus
<code>USER_VERILOG_FILES=userSource/userlog ic.v</code>	Source for user Verilog files

5 Adding a user module

A user module can be added to the work flow. The `.fwproj` file for the projects should have the following lines added:

```
USER_MODULE_NAME=UserModule
USER_VERILOG_FILES=userSource/userlogic1.v userSource/userLogic2.v
```

A user would typically add:

```
ENABLE_USER_PICOBUS=y
```

so that the logic to support the host -> Pico Card was included.

For example, the file `navsys.fwproj` might contain:

```
PICO_MODEL=E14FX20
ENABLE_PPC=y
ENABLE_RAM=y
ENABLE_PIC=y
ENABLE_GPIO=y
PROJ=CBBase
EXPORT_BIT_FILE=NavSys.bit
USER_MODULE_NAME=navsys
ENABLE_USER_PICOBUS=y
USER_VERILOG_FILES=navsysSource/navsys_userlogic.v
navsysSource/navsys.v
```

Entering the command:

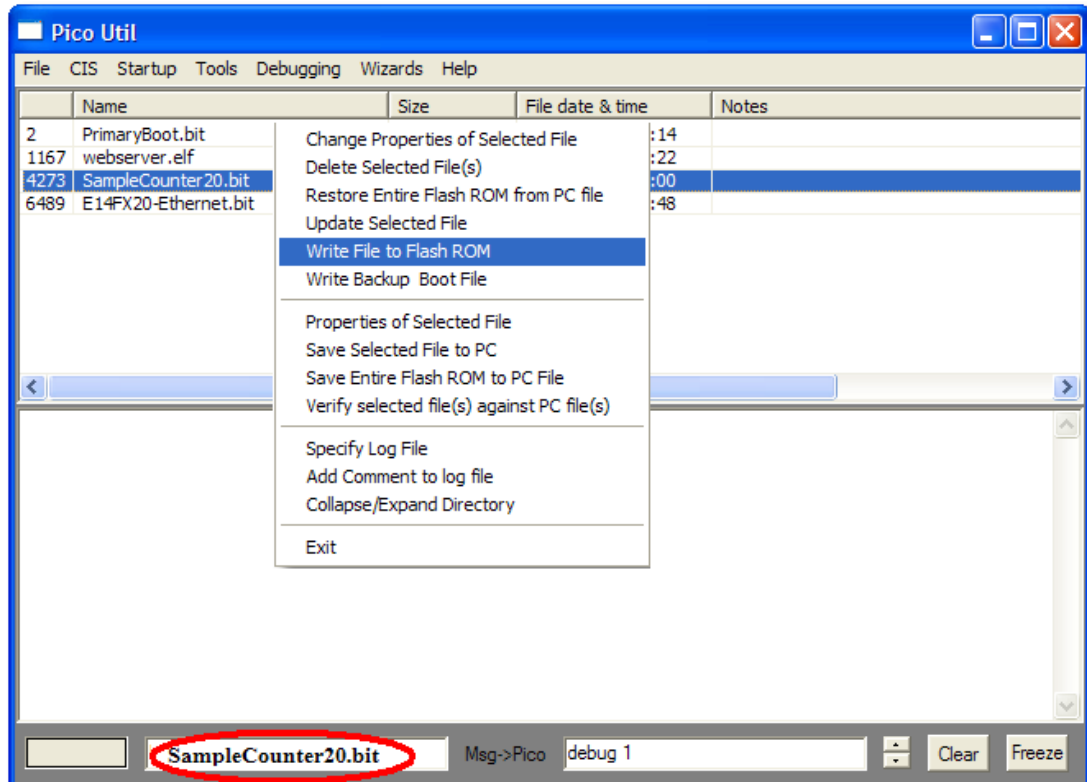
```
build navsys
```

would cause the build chain for the entire Pico firmware base plus the specified user modules to be built.

6 Loading the Bit file onto the Pico Card.

The new image **SampleCounter20.bit** can be loaded into the FPGA on the Pico Card using a number of strategies. We will describe here the strategy of writing the file to the flash ROM and then rebooting the FPGA with the new file. The tools PicoCommand.exe and PicoUtil.exe can be used for this purpose. In PicoUtil:

- From the main menu press **File / Write File to Flash ROM**,
- Select the file **c:\pico\sample\E14_counter\firmware\SampleCounter20.bit**.
- Press **OK** on the file parameters screen.



To reboot the FPGA with **SampleCounter20.bit** select the file **SampleCounter20.bit** and press enter (or right click and reboot, or menu / startup / boot). The new image will sign on with a message similar to the following:

```
PicoUtil.exe: V4.0.1.1 Release: Aug 1 2006 15:19:10.  
Pico.sys: V4.0.1.1 Release: Aug 1 2006 12:04:54.  
Pico.dll: V4.0.1.1. Release: Aug 1 2006 15:17:47  
Firmware: (4vfx20ff672) V4.0.1.1: Jul 14 2006 13:25  
PPC: Pico Monitor V4.0.1.1 Jul 14 2006 13:55:25.
```

The lower dashboard will display **SampleCounter20.bit**. You are now running the new

SampleCounter20.bit This image is essentially the same as PrimaryBoot.bit with the addition of the counters we have built in.

NOTE: The screen pictured above is a composite screen. Refer to [PicoUtil.chm](#) or [PicoUtil.pdf](#) for a discussion of writing files to flash ROM.