

# Pico Computing

M-501 / M-503 Getting Started Guide

March 7, 2012

## Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>System Requirements</b>	<b>1</b>
<b>3</b>	<b>Ubuntu Linux Configuration</b>	<b>2</b>
<b>4</b>	<b>Installing the Pico Software</b>	<b>4</b>
<b>5</b>	<b>Monitoring Cards With purty</b>	<b>6</b>
<b>6</b>	<b>Running the Stream Loopback Example</b>	<b>7</b>
6.1	Compiling the Stream Loopback Software . . . . .	7
6.2	Running the Stream Loopback Software . . . . .	8
<b>7</b>	<b>ISE Firmware Projects</b>	<b>8</b>
7.1	Copying and Opening the Project . . . . .	9
7.2	Pico Architecture . . . . .	9
7.3	Simulating the Project . . . . .	11
7.3.1	Launching a Simulation . . . . .	11
7.3.2	ISim GUI . . . . .	12
7.3.3	More Information . . . . .	13
7.4	Building the Project . . . . .	13
7.5	Other Samples . . . . .	14
<b>8</b>	<b>Hardware Troubleshooting</b>	<b>14</b>
8.1	Card not shown in the Pico Card Monitor (purty) . . . . .	14

<b>9 Software Troubleshooting</b>	<b>15</b>
9.1 Unable to compile programs . . . . .	15
9.2 Common Error Codes . . . . .	15
<b>10 Support</b>	<b>16</b>

## 1 Overview

Thank you for choosing Pico Computing FPGA products. This manual will help you setup and interface with the Pico M-501 and M-503 on Linux systems. We'll describe the process in several steps:

1. System requirements
2. How to set up Linux to properly handle hotswapping for the cards
3. How to install the software
4. How to run through an example program that uses the card

**Note: if you ordered a complete SuperCluster from Pico, the first three steps have been done for you, and you can start with step four.**

**Note: if you are migrating a design from 4.x framework to a 5.x version, be sure to read the Migration to 5.x Firmware documentation.**

## 2 System Requirements

In order to operate a Pico Computing FPGA product, your system must meet the following minimum system requirements:

1. System must be able to handle full height, full length PCI card
2. System must have at least one available x16 PCIe slot
3. System's power supply must have available PCIe power (GPU style connector)
4. Power supply that can deliver (per EX-500 backplane) a maximum of 240 W for 6 M-501 LX240Ts and a maximum of 120 W for 3 M-503 LX240Ts
5. System must have cooling fans to keep the FPGA(s) under 85 C
6. Ubuntu 64-bit desktop OS
7. Recommended motherboard chipset is Intel Tylersburg Series

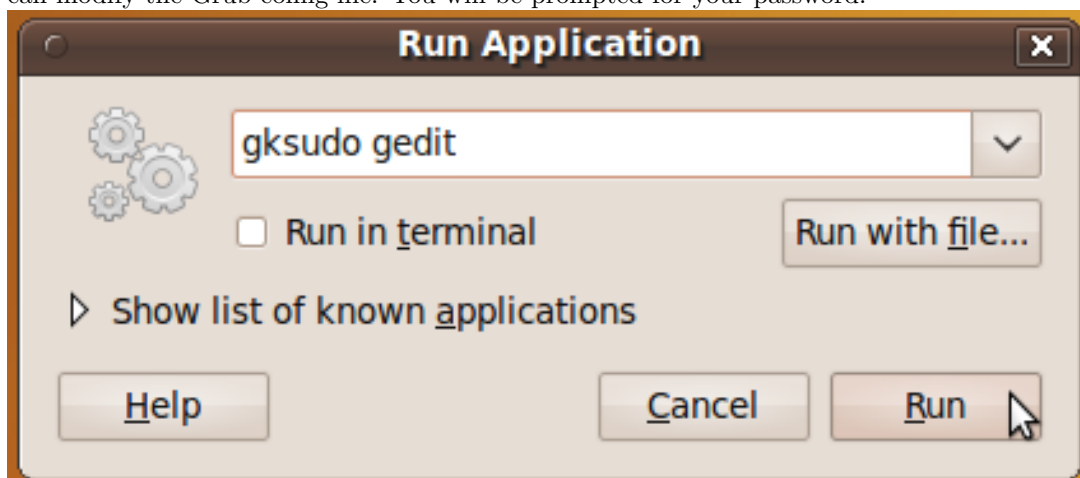
### 3 Ubuntu Linux Configuration

If you ordered a SuperCluster from Pico, all the configuration has been done. You can skip this section. Also, this section may be skipped if you're using Ubuntu 10.04 or newer, since the package installer is able to set these options automatically.

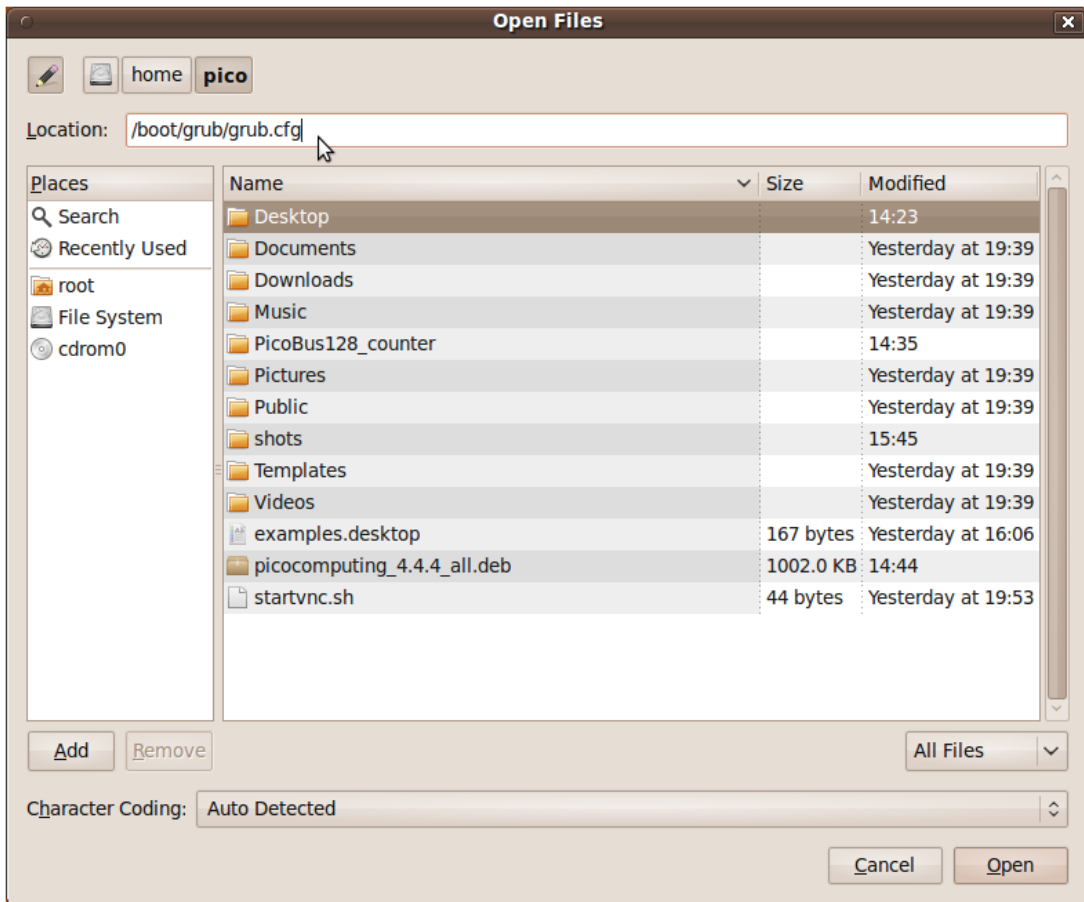
Because the PCIe interface of the M-501 and M-503 is in the FPGA, the PCIe link will go down and then come back up every time a new application is loaded. The Pico driver handles the details of this, and all you need to do is ensure that your system is configured for hotswapping. Since this process of “removing” and then adding a card back into a system is not something that’s usually done, especially on a desktop or server, Linux often won’t choose to enable its hotswapping features. We need to tell Linux at boot time to enable hotswapping.

We do this by adding the hotswapping command to the kernel boot info, found in the config file for the Grub bootloader.

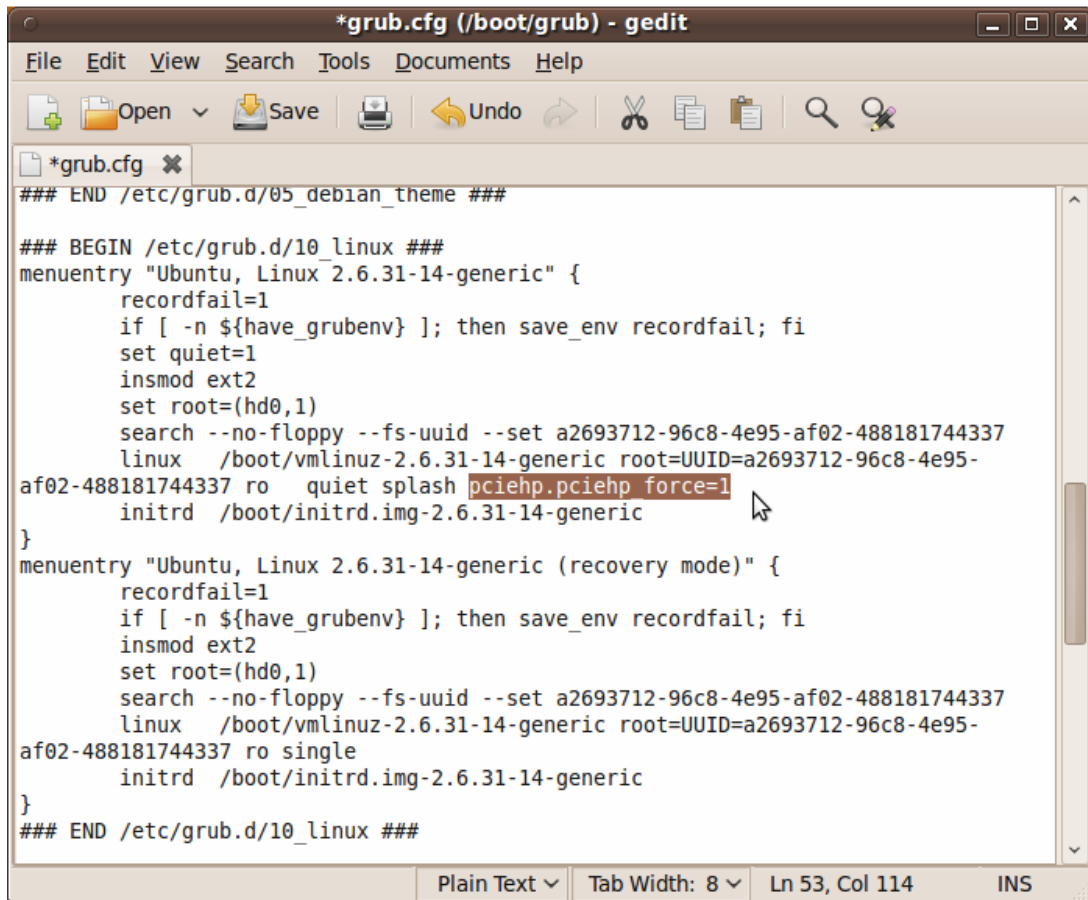
1. After logging in to Ubuntu, press Alt-F2 to bring up a “Run Application” window. Enter “gksudo gedit” (without quotes). This will open the gedit text editor in root mode, so you can modify the Grub config file. You will be prompted for your password.



2. In gedit, select “Open” and enter the filename `/boot/grub/grub.cfg` on Ubuntu 9.10 and `/boot/grub/ menu.lst` on earlier versions.



3. Find the entry for the kernel you're using and add "pciehp.pciehp\_force=1" to the end of the line containing the kernel name. In this picture it's "/boot/vmlinuz-2.6.31-14-generic". For a stock install of Ubuntu 9.10, this is on line 53. Make sure that if you update your kernel you also update this file to include the hotswap option for your new kernel version, or the Pico cards won't work.



```
### END /etc/grub.d/05_debian_theme ###

### BEGIN /etc/grub.d/10_linux ###
menuentry "Ubuntu, Linux 2.6.31-14-generic" {
    recordfail=1
    if [ -n ${have_grubenv} ]; then save_env recordfail; fi
    set quiet=1
    insmod ext2
    set root=(hd0,1)
    search --no-floppy --fs-uuid --set a2693712-96c8-4e95-af02-488181744337
    linux /boot/vmlinuz-2.6.31-14-generic root=UUID=a2693712-96c8-4e95-af02-488181744337 ro quiet splash pciehp.pciehp force=1
    initrd /boot/initrd.img-2.6.31-14-generic
}
menuentry "Ubuntu, Linux 2.6.31-14-generic (recovery mode)" {
    recordfail=1
    if [ -n ${have_grubenv} ]; then save_env recordfail; fi
    insmod ext2
    set root=(hd0,1)
    search --no-floppy --fs-uuid --set a2693712-96c8-4e95-af02-488181744337
    linux /boot/vmlinuz-2.6.31-14-generic root=UUID=a2693712-96c8-4e95-af02-488181744337 ro single
    initrd /boot/initrd.img-2.6.31-14-generic
}
### END /etc/grub.d/10_linux ###
```

4. Save this file and reboot your computer.

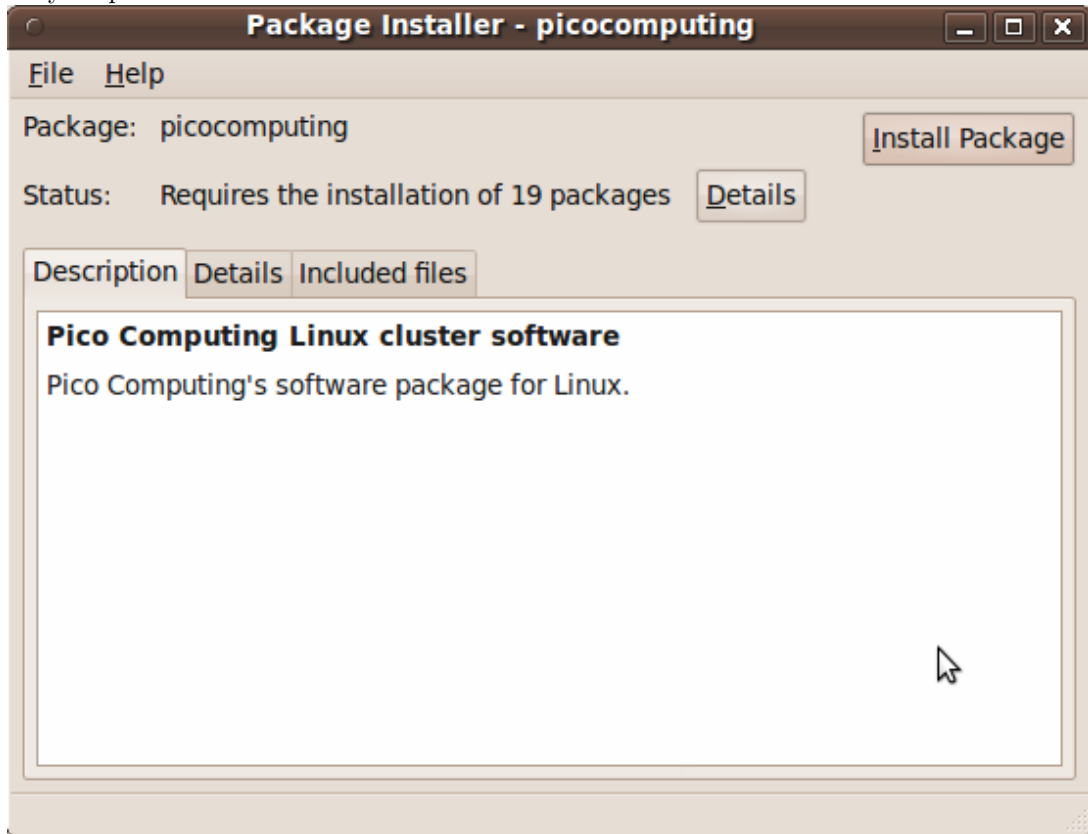
## 4 Installing the Pico Software

If you ordered a SuperCluster from Pico, the installation has been done. You can skip this section.

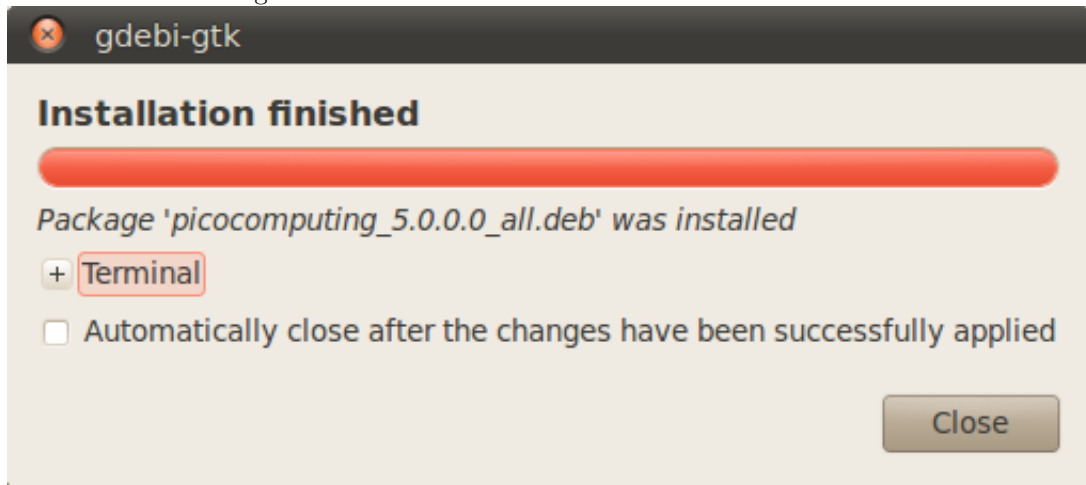
This section describes the procedure for installing the Pico software on an Ubuntu Linux system. After you install the software, you should take time to run the StreamLoopback128 example application to verify correct installation of the software and hardware, and to learn how to use the card. Instructions on how to run the example application can be found in a later section of this guide.

1. The Pico software is distributed as an Ubuntu package named picocomputing\_X.Y.Z.deb. Download this file from [www.picocomputing.com/downloads/software.php](http://www.picocomputing.com/downloads/software.php) and double click it to begin the installation.
2. You'll see this summary of the package information. It may say that it needs to download and install several dependencies like a compiler, etc. Click the "Install Package" button to start the installation. Because you are changing the software on your computer, Ubuntu will ask

for your password.



3. The system will download any required packages and then install them and the Pico package. This may take a minute or two if you have a slow internet connection. When it's finished, you should see this message:

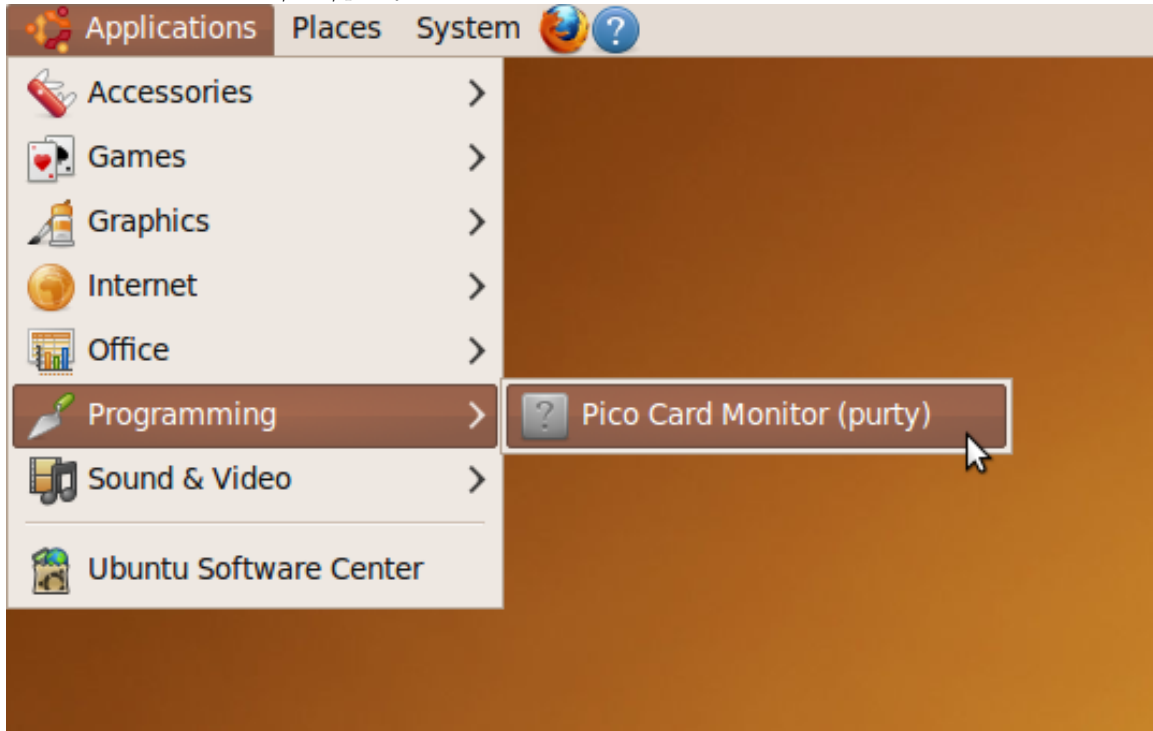


4. Now that the software is installed, take a look at the card with the "purty" monitoring program, which is described in a later section.

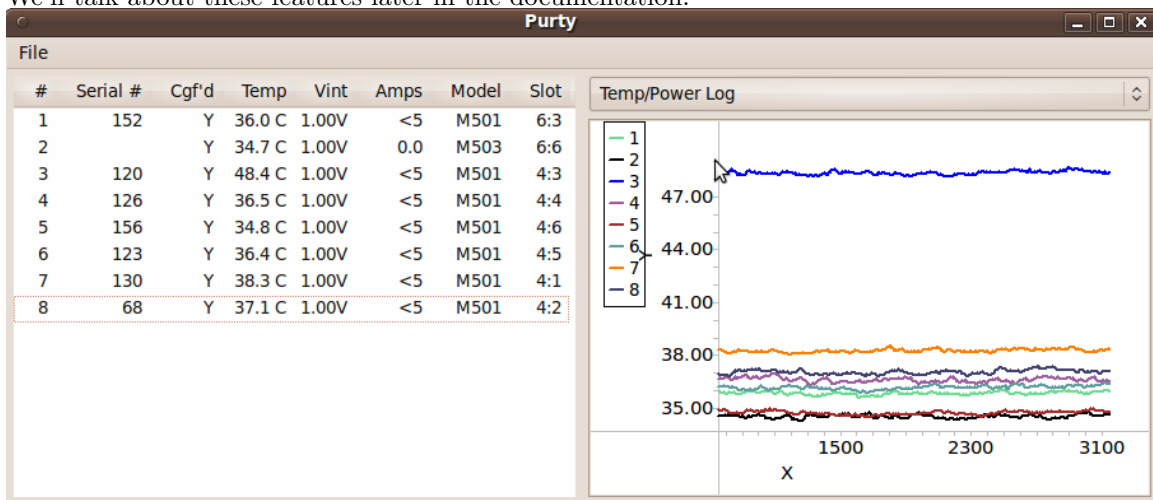
## 5 Monitoring Cards With purty

Pico provides a program for monitoring the state of Pico cards, called “purty” - Pico Utility for Reliability and Testing. Purty reports the status and health of all the cards in the system. It reports the temperature and voltage of the FPGA, whether the FPGA is loaded properly, the physical location in the system of each of the cards, and serial numbers.

To start purty, go to the “Applications” menu at the top of the screen and select “Programming -> Pico Card Monitor (purty)”. Alternatively, you can launch purty from the command line. It’s located in \$PICOBASE/bin/purty.



Purty also contains some features to read and write simple values to the cards, for testing purposes. We’ll talk about these features later in the documentation.



## 6 Running the Stream Loopback Example

Now that you've installed the software and looked at your card in the card monitor, you can run a simple application. It's called the Stream Loopback, and it communicates with a piece of firmware that's built in to the FPGA. The firmware in the FPGA is very simple; you write a stream of numbers to it, it modifies the numbers slightly, and you read the modified numbers back from it. Despite the simplicity of the actual firmware, the Stream Loopback uses many of the important pieces of the Pico firmware and software framework, so it's a good way to get acquainted with the card.

### 6.1 Compiling the Stream Loopback Software

1. The Stream Loopback software project is found in the `samples/StreamLoopback128/software` directory of the Pico installation. You can refer to this location with the `PICOBASE` environment variable, like:

```
$PICOBASE/samples/StreamLoopback128/software
```

or you can spell out the full location, like:

```
/usr/src/picocomputing-5.0.0.0/samples/StreamLoopback128/software
```

Since this is a system directory and you're probably not running with root access now, copy the software folder to your home directory and navigate to that directory. From the command line:

```
cp -r $PICOBASE/samples/StreamLoopback128 ~/  
cd ~/StreamLoopback128/software
```

2. Now you can build the executable by running the “make” command. This produces an executable called “StreamLoopback”. Run the program by typing “./StreamLoopback”. If you run the program just like that, you'll see that it asks you which FPGA bit file you want to use. You can use the pre-built bit file found in:

```
$PICOBASE/samples/StreamLoopback128/firmware/M501_LX240T_StreamLoopback128.  
bit
```

(for the M-501) or

```
$PICOBASE/samples/StreamLoopback128/firmware/M503_LX240T_StreamLoopback128.  
bit
```

(for the M-503)

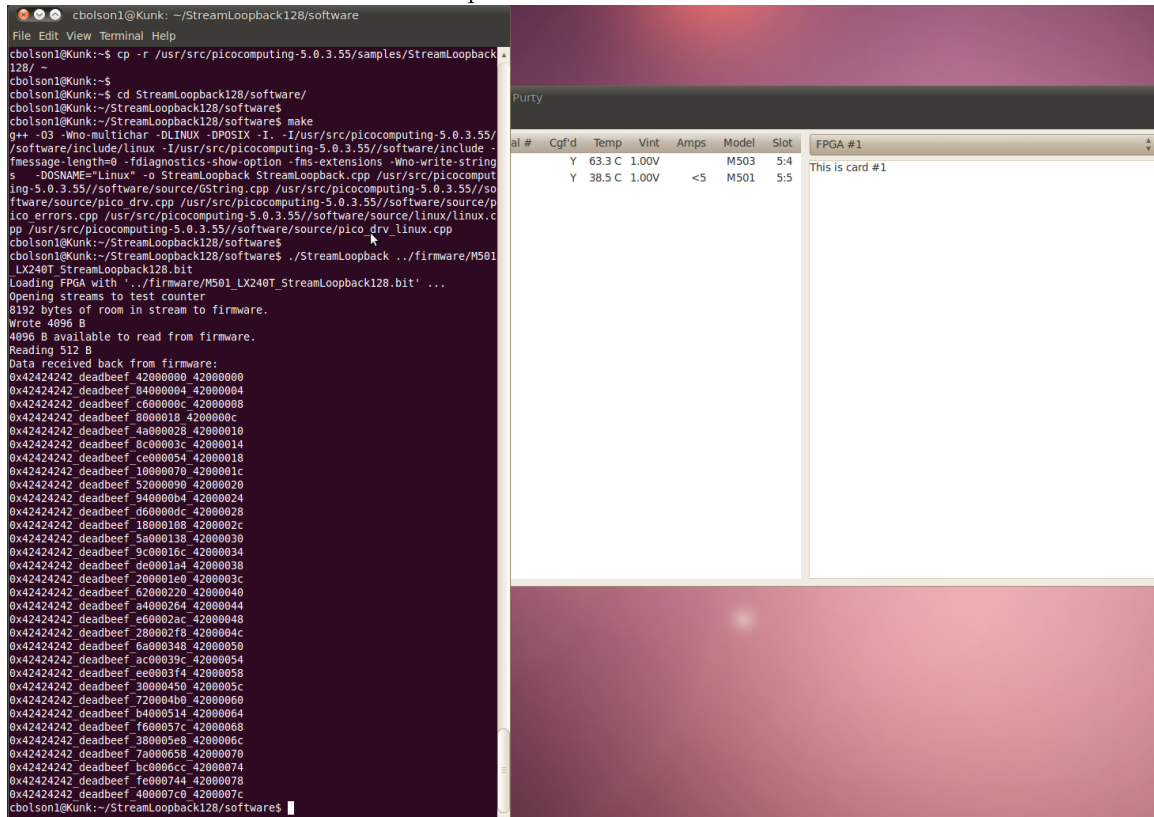
So, for the M-501, we would type:

```
./StreamLoopback $PICOBASE/samples/StreamLoopback128/firmware/  
M501_LX240T_StreamLoopback128.bit
```



## 6.2 Running the Stream Loopback Software

The first thing the program does is load the bit file into the FPGA. This can take a few seconds. After the file is loaded, the program opens streams to the FPGA. Next, it fills a software buffer that will be recognizable when it comes back. It then streams the contents of the software buffer to the FPGA. The FPGA modifies the data that it receives from the host slightly, and places the data in an output stream FIFO. The software then reads data from the FPGA via a stream and prints the received data to the screen. The whole process looks like this:



```
cbolson1@Kunk: ~/StreamLoopback128/software
File Edit View Terminal Help
cbolson1@Kunk:~$ cp -r /usr/src/picocomputing-5.0.3.55/samples/StreamLoopback128/ .
cbolson1@Kunk:~$ cd StreamLoopback128/software/
cbolson1@Kunk:~/StreamLoopback128/software$ make
g++ -O3 -Wno-multichar -DLINUX -DPOSIX -I. -I/usr/src/picocomputing-5.0.3.55//software/include/Linux -I/usr/src/picocomputing-5.0.3.55//software/include -Wmessage-length=0 -fdiagnostics-show-option -fas-extensions -Wno-write-strings -DOSNAME="Linux" -o StreamLoopback StreamLoopback.cpp /usr/src/picocomputing-5.0.3.55//software/source/GString.cpp /usr/src/picocomputing-5.0.3.55//software/source/pico_drv.cpp /usr/src/picocomputing-5.0.3.55//software/source/pico_errors.cpp /usr/src/picocomputing-5.0.3.55//software/source/linux/linux.cpp /usr/src/picocomputing-5.0.3.55//software/source/pico_drv_linux.cpp
cbolson1@Kunk:~/StreamLoopback128/software$ ./StreamLoopback ../firmware/M501_LX240T_StreamLoopback128.bit
Loading FPGA with '../firmware/M501_LX240T_StreamLoopback128.bit' ...
Opening streams to test counter
8192 bytes of room in stream to firmware.
Write 4096 B
4096 B available to read from firmware.
Reading 512 B
Data received back from firmware:
0x42424242_deadbeef_42000000_42000000
0x42424242_deadbeef_b4000004_42000004
0x42424242_deadbeef_c600000c_42000008
0x42424242_deadbeef_80000018_4200000c
0x42424242_deadbeef_4a000028_42000010
0x42424242_deadbeef_8c00003c_42000014
0x42424242_deadbeef_ce000054_42000018
0x42424242_deadbeef_10000070_4200001c
0x42424242_deadbeef_52000090_42000020
0x42424242_deadbeef_940000b4_42000024
0x42424242_deadbeef_d00000dc_42000028
0x42424242_deadbeef_18000108_4200002c
0x42424242_deadbeef_5a000138_42000030
0x42424242_deadbeef_9c00016c_42000034
0x42424242_deadbeef_de0001a4_42000038
0x42424242_deadbeef_200001e0_4200003c
0x42424242_deadbeef_62000220_42000040
0x42424242_deadbeef_a4000264_42000044
0x42424242_deadbeef_e60002ac_42000048
0x42424242_deadbeef_280002f8_4200004c
0x42424242_deadbeef_6a000348_42000050
0x42424242_deadbeef_ac00039c_42000054
0x42424242_deadbeef_e80003f4_42000058
0x42424242_deadbeef_30000450_4200005c
0x42424242_deadbeef_720004b0_42000060
0x42424242_deadbeef_b4000514_42000064
0x42424242_deadbeef_f600057c_42000068
0x42424242_deadbeef_300005d0_4200006c
0x42424242_deadbeef_7a000658_42000070
0x42424242_deadbeef_bc0006cc_42000074
0x42424242_deadbeef_fe000744_42000078
0x42424242_deadbeef_400007c0_4200007c
cbolson1@Kunk:~/StreamLoopback128/software$
```

More details about the Stream Loopback sample can be found in the StreamLoopback128 documentation.

## 7 ISE Firmware Projects

When you install the Pico package, one of the things you install is a set of sample Xilinx ISE projects that you can use as templates for your own firmware. The best way to get started on your own project is to copy the sample project that best fits your application, and then expand upon it. (These projects are intended for use with Xilinx's ISE Foundation version 13.2. Newer versions of ISE should work, but older ones will not.)

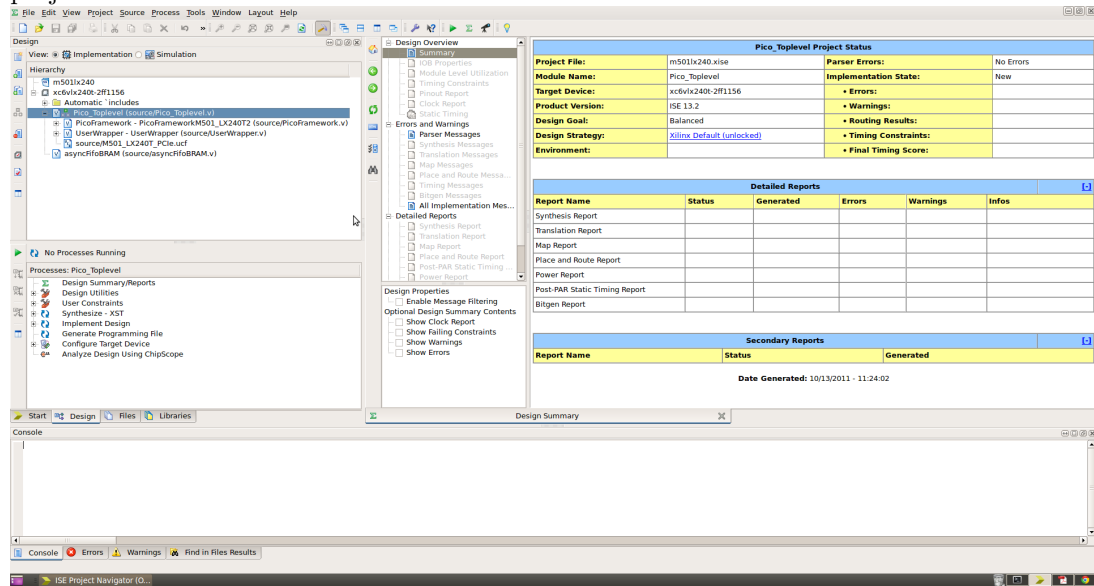
Let's take a look at a commonly-used sample project, the StreamLoopback128 example. The StreamLoopback128 demonstrates DMA transactions with the host, but it does not demonstrate the PiCoBus or the DDR3.

## 7.1 Copying and Opening the Project

1. Start by copying the samples/StreamLoopback128 directory to your home directory, where we'll work on it. For example, run:

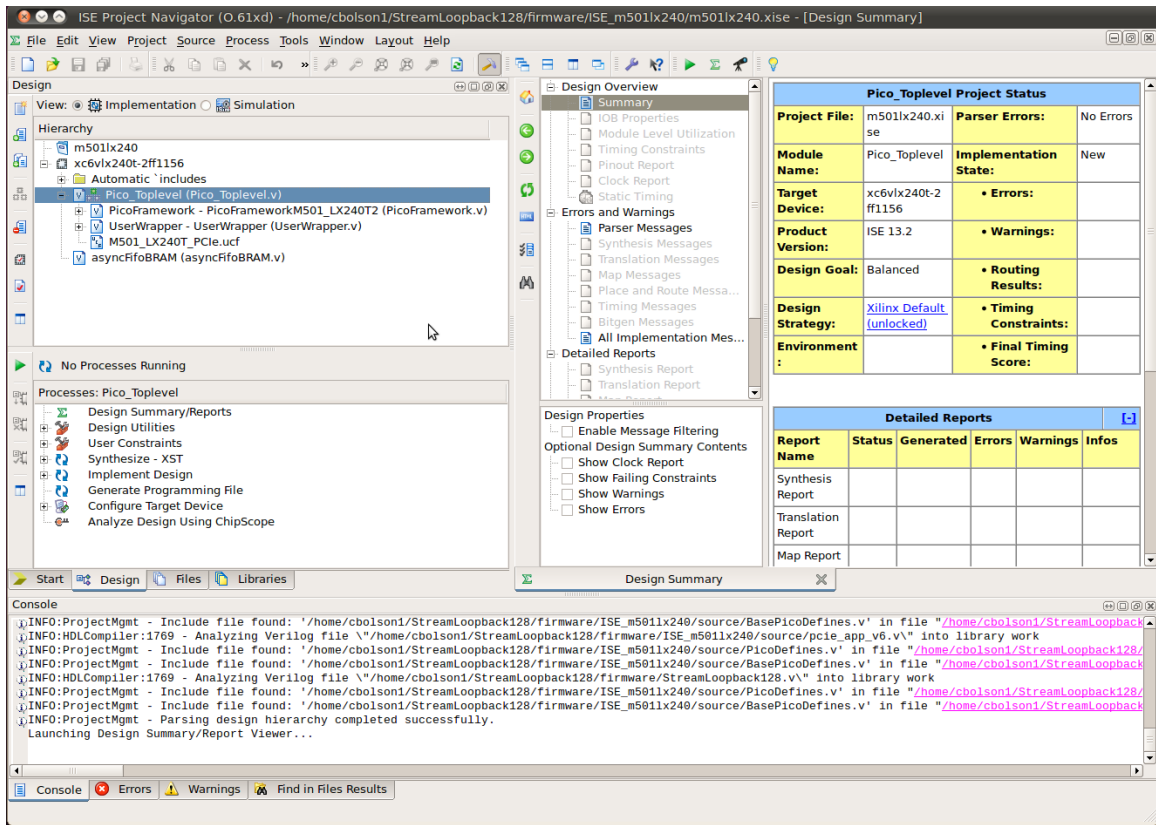
```
cp -r $PICOBASE/samples/StreamLoopback128 ~/
```

2. Open ISE Project Navigator from your Xilinx installation. In ISE, select File->Open Project and point to StreamLoopback128/firmware/ISE\_m501x240/m501x240.xise, which is the ISE project file. Your screen should look like this:

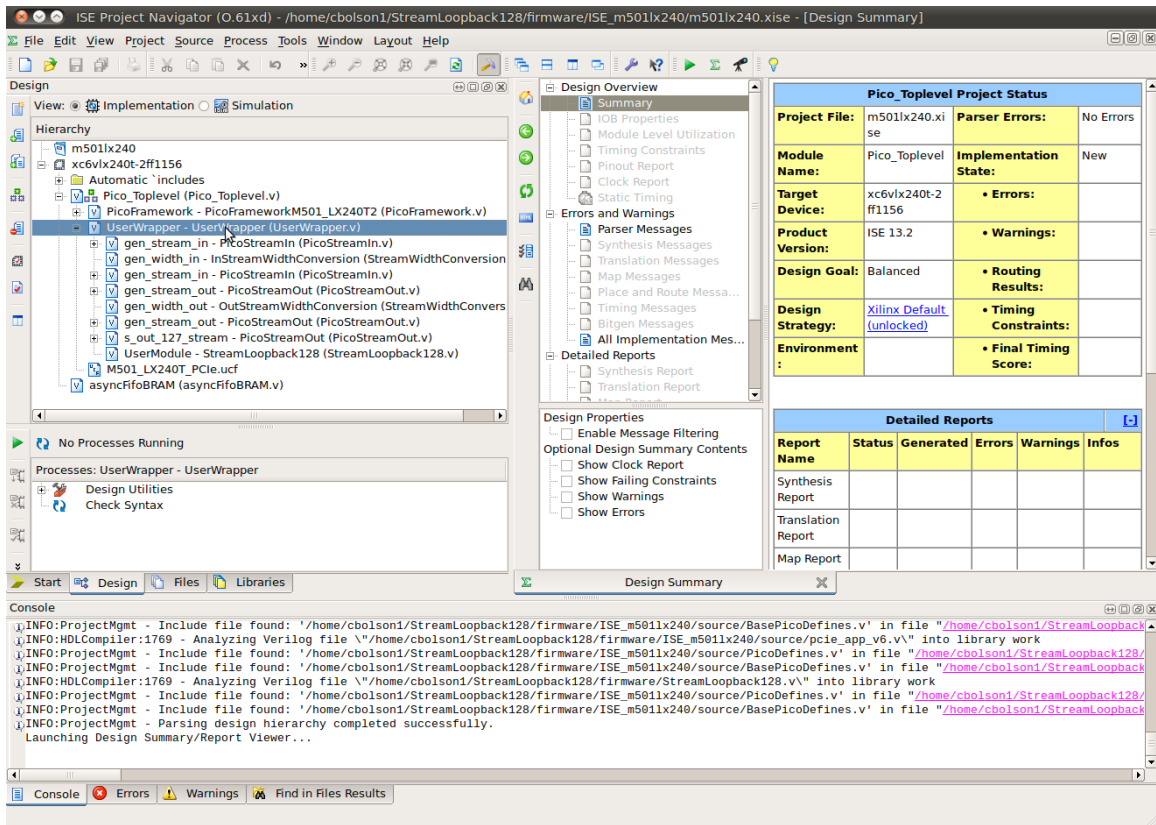


## 7.2 Pico Architecture

The top-level module for the M-501 and the M-503 is called Pico\_Toplevel. That top-level module instantiates the systems required for every project, which include the PicoFramework and the UserWrapper modules. Projects that interface with a DDR3 system will also instantiate at least one Xilinx Memory Interface Generator (MIG) module. User Constraints Files (UCF) convey pin placement and timing information to the Xilinx ISE, which is used during the mapping and the place and route phases of generating a bitfile.



The PicoFramework module instantiates modules used for PCIe communication with the host system. The UserWrapper module instantiates modules for streaming data to and from the host (PicoStreamIn and PicoStreamOut), PicoBus communication with the host (Stream2PicoBus), streaming data to and from the DDR3 system (PicoStreamToAXI), interfacing with the DDR3 system from firmware (PicoAXIInterconnect), and the user module for a given design (UserModule).



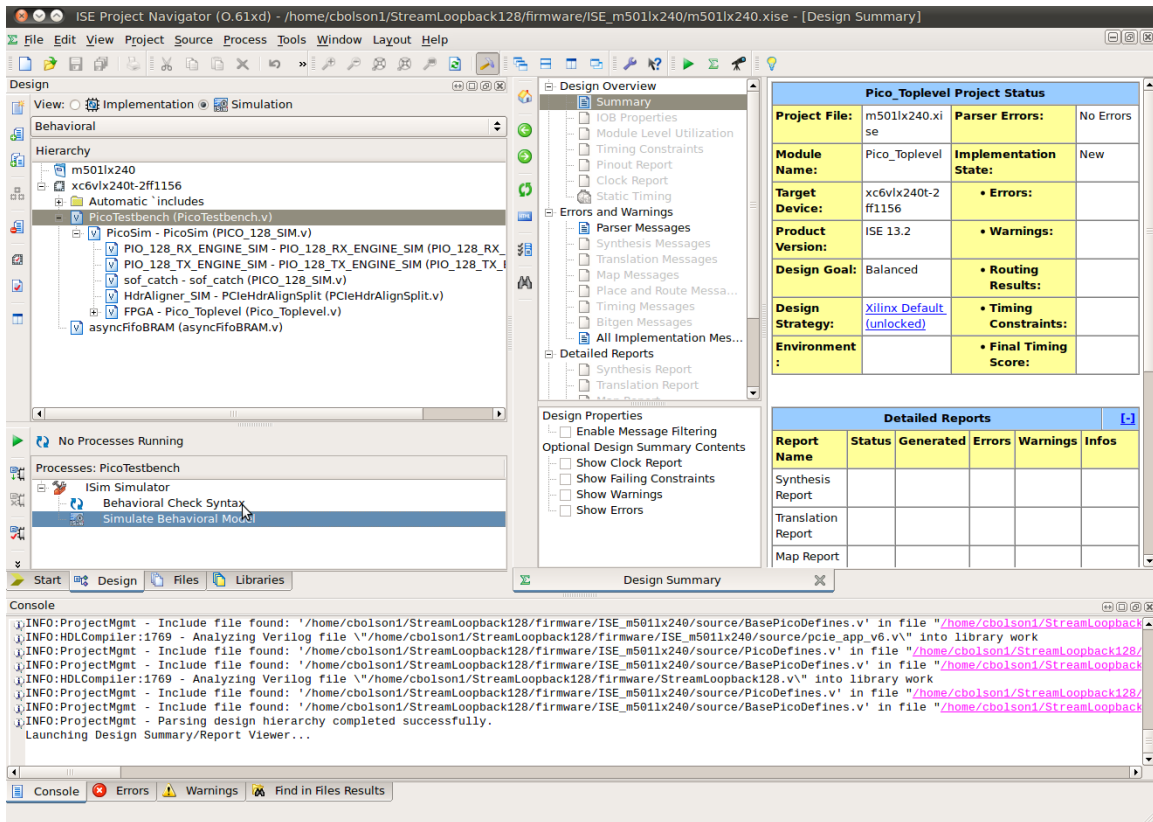
For a given design, the user should not modify any of the previously mentioned files, except they should create their own UserModule file for their application. They should then define the name of their UserModule in PicoDefines.v

## 7.3 Simulating the Project

Simulation is a powerful tool for debugging new designs. Pico utilizes ISim, which is the Xilinx simulator provided with the ISE tools, for running simulations.

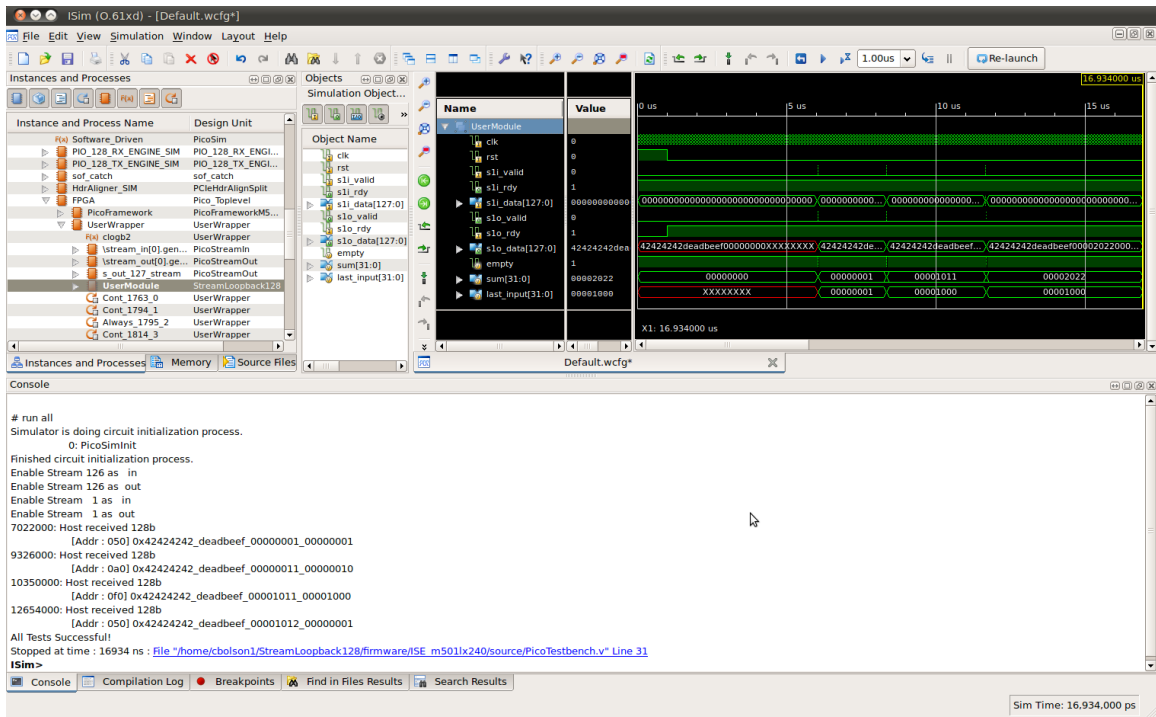
### 7.3.1 Launching a Simulation

Simulation can be started by first selecting the “Simulation” view, which can be found directly above the “Hierarchy” pane in the Xilinx ISE. Select the PicoTestbench.v file in the “Hierarchy” pane, and then double-click “Simulate Behavioral Model” in the “Processes” pane. If ISE asks you to regenerate any IP cores, click yes. The design will compile for simulation, and the ISim GUI will launch.



### 7.3.2 ISim GUI

When the ISim GUI launches, you will see the same hierarchy from the ISE “Hierarchy” pane on the left side of the ISim GUI in the “Instances and Processes” pane. The top-level instance for simulation is the PicoTestbench, which instantiates a PicoSim module. The PicoSim module instantiates the FPGA top-level module, any required DDR3 or SRAM simulation models, and PCIe communication models. The PicoSim module also contains many functions from the PicoAPI, which provide a common interface between simulation and software execution.



To add signals to the waveform, select a module in the “Instances and Processes” pane. The module’s signals will then be displayed in the “Objects” pane. Users can add individual signals to the waveform by dragging them from the “Objects” pane to the waveform window. Similarly, users can add all signals for a module to the waveform by selecting the module name in the “Instances and Processes” pane and dragging it to the waveform. Signals may be grouped in the waveform by selecting multiple signals, right-clicking, and clicking on “New Group”.

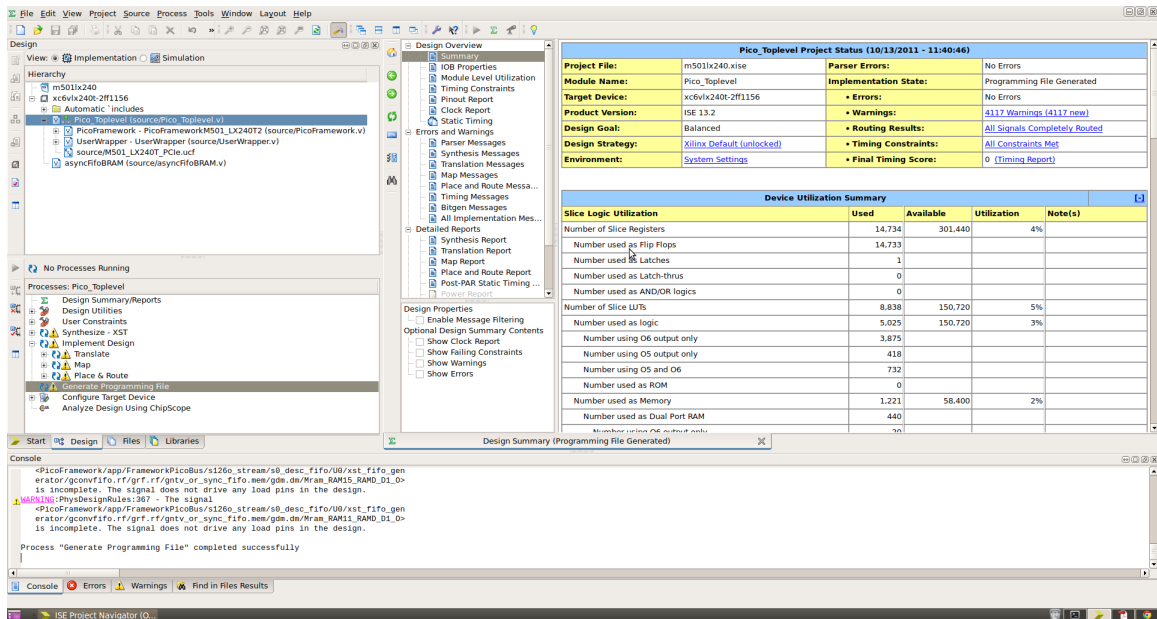
Errors, warnings, information, and debug messages printed by the testbench or by the firmware will be displayed in the “Console”, which is one of the tabs at the bottom of the ISim GUI. Successful sample simulations will print “All tests Successful!” at the end of the simulation.

### 7.3.3 More Information

For more information about simulating designs using ISim, users should consult the Xilinx ISim User Guide or the Xilinx ISim In-Depth Tutorial.

## 7.4 Building the Project

Now that you have the project open, go to the “Hierarchy” pane in the upper left corner, and make sure the “Pico\_Toplevel” module (or the similarly-named module for your card) is selected, which sets it as the toplevel. Double-click “Generate Programming File” in the “Processes” pane on the left. The whole build process takes 5 or 10 minutes depending on the speed of your computer. When it is finished, you will have a bit file, called Pico\_Toplevel.bit. You can run through the Stream-Loopback128 software again, but this time use this file instead of the precompiled one provided in the distribution. You should do this to make sure that your ISE installation is operating correctly.



After ISE has built the firmware, it's always a good idea to check and make sure the timing score is 0. You can see the timing score in the design summary section in the previous screenshot. Also notice that this entire sample project used just 5% of the LUTs in the FPGA, mostly by the Pico framework. This means that the FPGA is wide open for you to use in your application.

## 7.5 Other Samples

If you instead would like to use a sample that used the PicoBus, you may try the PicoBus128\_HelloWorld sample. For information on how to use the sample, please see the PicoBus128\_HelloWorld documentation.

If you would like to use a sample that interfaces with the DDR3 system, you may use the DDR3\_MovingAverage sample (for the M-501 or the M-503), depending on which card you are using. More information about that sample can be found in the DDR3\_MovingAverage documentation.

## 8 Hardware Troubleshooting

Be sure to verify that your system meets all system requirements.

### 8.1 Card not shown in the Pico Card Monitor (purty)

If your card does not show up in the monitor program, open a command prompt and type (case sensitive):

```
lspci -nn | grep Pico
```

If you don't see anything, then the system does not see the card at all. There is nothing that can be done in software. If you have a board that is a module on the EX-500, such as the M-501, M-502,

or M-503, please check the LEDs to ensure the board is getting power. See the hardware manual for LED locations.

If, however, the lspci program does report a Pico card, the problem is likely with the driver. To see if the driver is loaded, run (case sensitive):

```
lsmmod | grep pico
```

If you see no output, the driver is not loaded. Try loading the driver by running the following as root:

```
depmod; modprobe -v pico
```

Now use dmesg to look at the system log to see if the driver has reported any errors:

```
dmesg
```

If the modprobe command didn't fix the problem, please contact Pico with the output of these commands.

## 9 Software Troubleshooting

**Note:** Details about the software API and a description of the supported functions available to users can be found in the Pico API documentation. Customers should only use functions found in the Pico API documentation and should not use internal functions found in the distributed Pico drivers.

### 9.1 Unable to compile programs

Check to make sure the PICOBASE environment variable is pointing to the latest Pico installation. This is set when the Pico package is installed. You should see something like this:

```
echo $PICOBASE  
[output] /usr/src/picocomputing-5.0.0.0
```

If you don't see PICOBASE pointing to a valid directory, you can set it with:

```
export PICOBASE=/usr/src/picocomputing-5.0.0.0
```

Now rebuild the program with the the PICOBASE variable set.

### 9.2 Common Error Codes

These are some of the commonly seen error codes from the Pico software:

- 8003: Couldn't find an appropriate Pico card. Are you using an E-18 .bit file when you have just an M- 501, for example?
- 8004: .bit file not found. Make sure the file exists.



- 8014: The card is already in use by another program. You can't have more than one program using a card simultaneously unless you explicitly allow it.

For a more detailed list of error codes provided by the Pico software, please consult the Pico API error code appendix.

## 10 Support

For further support using Pico products, both firmware and software, please feel free to contact us at [support@picocomputing.com](mailto:support@picocomputing.com).