

3

Tornado Tutorial

3.1 Introduction

This tutorial is designed to introduce you to key features of Tornado using the integrated VxWorks target simulator. It does not require any target hardware or special configuration of your host system. It is not a programming tutorial; the sample program was written to exercise Tornado in illustrative ways.

While simply reading the tutorial may be edifying, we encourage you to perform the steps described in this chapter so that you can experience Tornado first hand. In the course of the tutorial, you will:

- Use the project facility to create a workspace and project for a sample program.
- Build the program from the project facility GUI.
- Download the program to the VxWorks target simulator on your host system.
- Use the Tornado shell to run the sample program.
- Use the browser to observe memory usage on the target simulator.
- Use the WindView software logic analyzer to graphically display the execution flow of the sample program and to identify a problem with task prioritization.
- Use the debugger to identify a runtime application error.

Naturally, you will have several opportunities to modify the program to correct its runtime behavior, rebuild it, download it, and run it again.

The tutorial assumes minimal knowledge of multi-threaded programming in C, and basic Windows usage.

You might approach the tutorial with the following scenario in mind:

A colleague has abruptly announced that he is going on vacation, and the VP of Engineering has handed you the delinquent's portion of The Project. You are alarmed. His code is typically cryptic, poorly documented, and full of run-time mischief. The thought of reading his source gives you a headache. And the target hardware seems to have gone on vacation with him. You decide to use Tornado's target simulator and analytical tools to see how the code behaves...



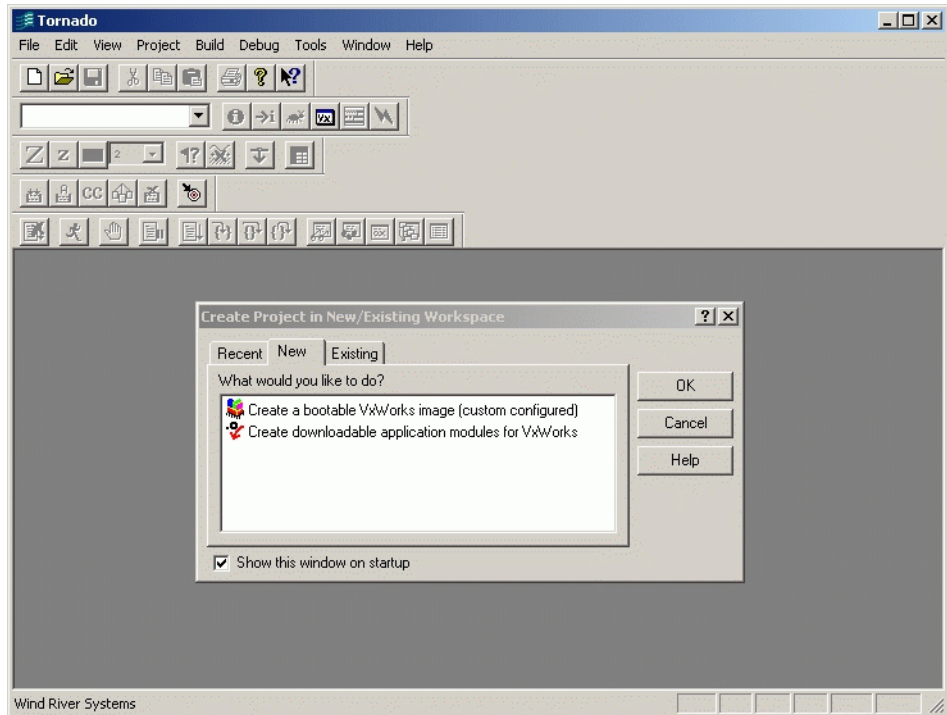
NOTE: This tutorial is intended for use with the integrated VxWorks target simulator. The results may differ if used otherwise (with target hardware, for example).

3.2 Start Tornado

To start Tornado, select Start>Programs>Tornado2.2>Tornado (with a default installation), or select the Tornado desktop shortcut (if you have created one).

If Tornado is unable to connect to a Tornado registry, you will be prompted to start one. See *Step 10: Set Up the Tornado Registry*, p.11 for information about registry configuration options at installation. You can start a registry manually by selecting the Tornado Registry option from the Tornado program group menu.

The main Tornado window appears with the Create Project dialog box open by default:

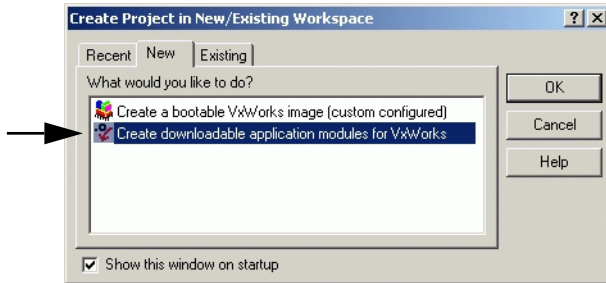


You can use the toolbars as floating palettes or dock them on the sides, top, or bottom of the main Tornado window.

3.3 Create a Project

If the Create Project in New/Existing Workspace dialog box is not displayed, click File>New Project.

Then select the option for a downloadable application:



Click OK to continue.

The Tornado application wizard appears. This wizard guides you through the steps of creating a new project. In the first step, the wizard prompts you for:

- A name for the project.
- The full path and name of the directory you want to use for the project files.
- An optional description of the project.
- The name and location of the workspace file, which contains information about the workspace, including which projects belong to it.

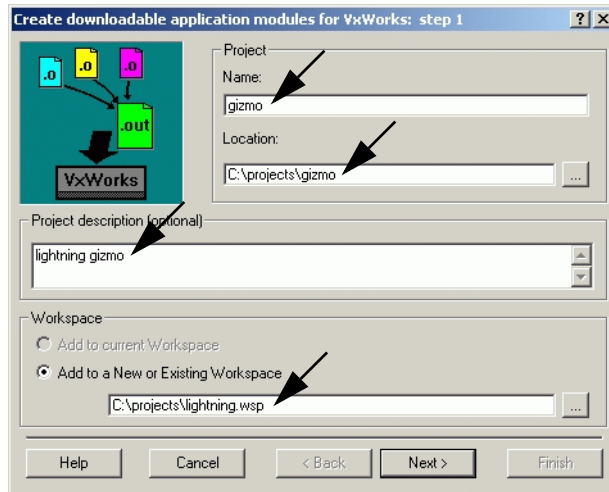
A *project* consists of the source code files, build settings, and binaries that are used to create an application.

A *workspace* is simply a grouping of one or more projects, which provides a useful means of working with related material, sharing code between projects, and associating related applications. Once a project and workspace have been created, a workspace window displays information about the projects that it contains.

You can accept the wizard's defaults for the project and workspace, but it is preferable to set them up outside of the Tornado installation tree. (If you do so, you won't have to untangle your work from the Tornado tree when you upgrade Tornado.)

In the wizard dialog (displayed below):

- The project name is gizmo.
- The project directory is `c:\projects\gizmo`.
- The project description is lightning gizmo.
- The path and file name for the workspace are `c:\projects\lightning.wsp`:



NOTE: The directory path for the workspace must exist—or at least match the path (or an initial segment thereof) defined in the project Location field—for the wizard to complete project creation.

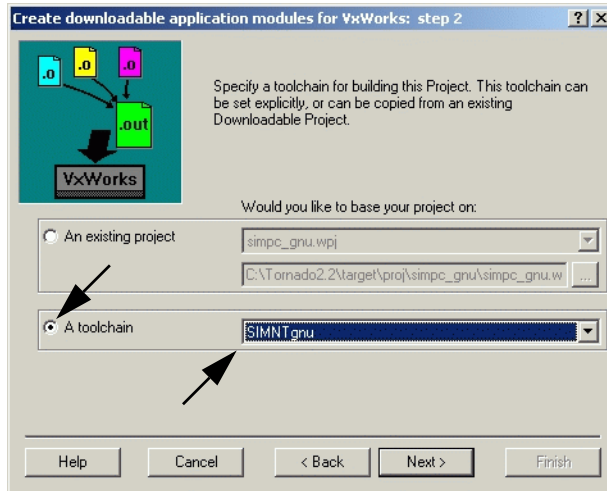
Click Next to continue.

In the second wizard dialog, you identify the toolchain with which the downloadable application will be built.

A *toolchain* is a set of cross-development tools used to build applications for a specific target processor (preprocessor, compiler, assembler, and linker).

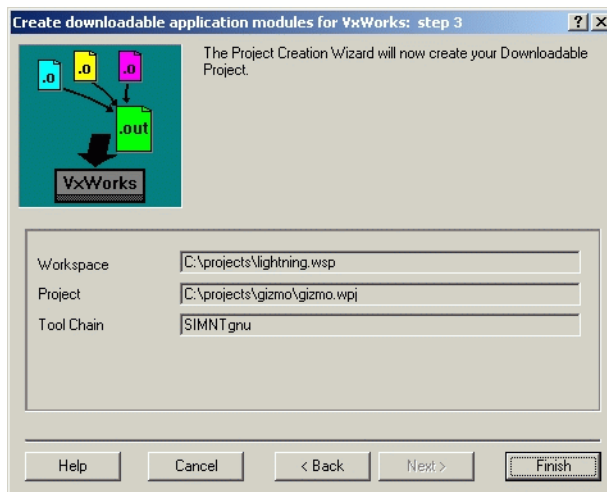
The default toolchain names for target simulators take the form `SIMhostOsgnu` (for example, `SIMNTgnu`).

Select A toolchain and the option for the target simulator from the drop-down list, as illustrated below:



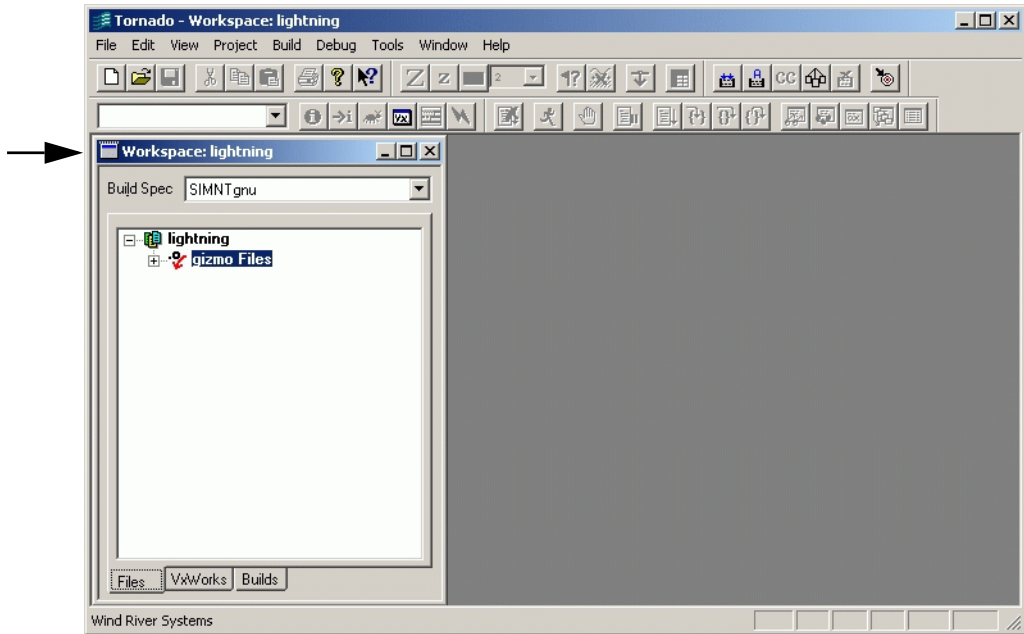
Click Next to continue.

The final wizard dialog confirms your selections:



Click Finish to continue.

The Workspace window appears. The workspace window title includes the name of the workspace, and the window itself includes a folder for the project:



NOTE: Context menus provide access to all commands that can be used with the objects displayed in, and the pages that make up, the Workspace window. Use the right mouse button to display an object's context menu.

3.4 Add the Sample Source File to the Project

The sample program for this tutorial is **cobble.c**. It is a simple multi-tasking application that simulates a data collection system in which data is retrieved from an external source (for example, a device that generates interrupts as data comes in). The four **cobble** tasks are:

- **tCosmos**, which simulates an interrupt service routine (ISR) and generates new data.
- **tSchlep**, which collects the data.
- **tCrunch**, which performs calculations on the data to obtain a result value.
- **tMonitor**, which monitors the results and prints a warning message if they are outside of the safety range.

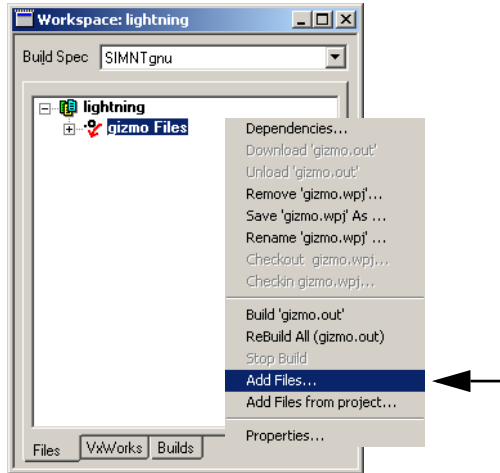
The **cobble** program also includes the **progStart()** and **progStop()** routines. Later in the tutorial, you will use these routines to start and stop the program.

Before you add the sample program to the project, copy it from the Tornado installation tree to another location, such as the project directory you have set up with the Tornado project wizard.

The sample file **cobble.c** is located in *installDir*\target\src\demo\start.

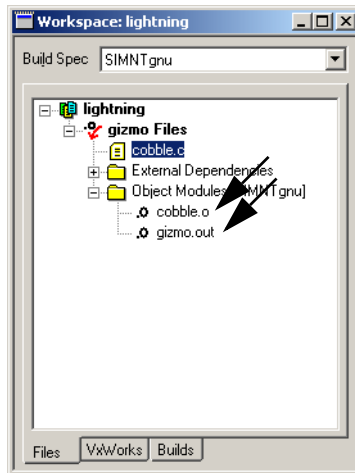
In the examples provided thus far in the tutorial, **c:\projects** has been used as the base directory for our project work, and subsequent examples are based on **cobble.c** being copied to that directory.

After you have copied **cobble.c** to another location, add it to the project. Select Add Files from the context menu in the File view of the workspace:



Then use the file browser that appears to select **cobble.c**.

Open the Object Modules folder in the Files view so you can see the names of the object files that are built from the project:



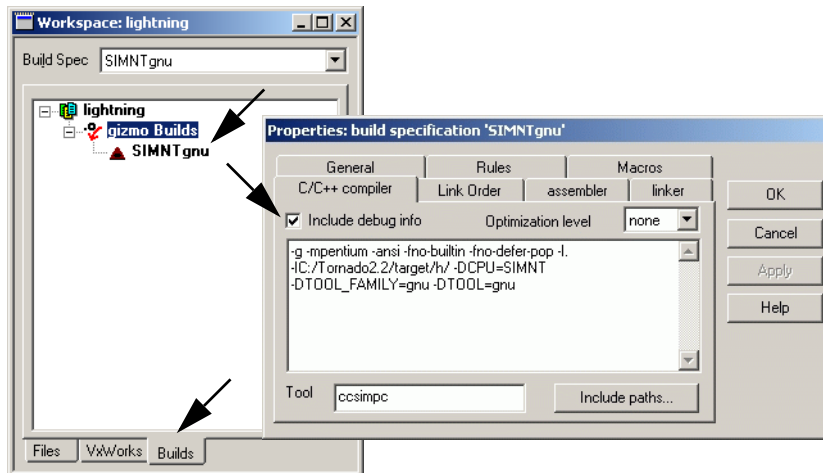
The Tornado project facility automatically creates the file *projectName.out* as a single, partially linked module when a project is built. It comprises all of the individual object modules in a downloadable application project, and provides for downloading them all to the target simultaneously.

3.5 Build the Project

To review the default build settings defined when you created the project, select the Builds tab on the Workspace window, open the gizmo Builds folder, and double-click on the default build name (for example, SIMNTgnu).

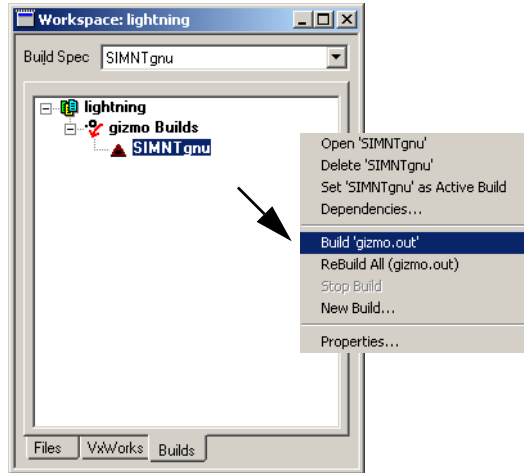
The properties sheet for the build appears, which you can use to review makefile rules and macros, as well as the compiler, assembler, and linker options used for the build.

When you display the C/C++ compiler page, you will note that the Include debug info checkbox is selected by default. This selection ensures that optimization is turned off when the project is compiled with debug information:



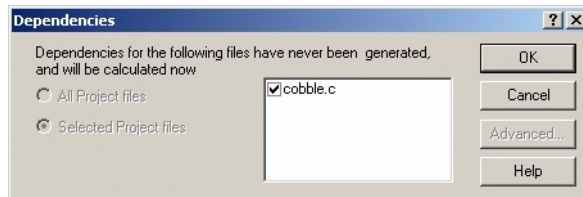
Close the property sheet by clicking on the Cancel button.

Build the project by selecting Build 'gizmo.out' from the context menu:



The option Build '*projectName.out*' builds all project modules as a single, partially linked module that is optimal for downloading to a target.

Before Tornado builds the project, the Dependencies dialog box warns you that makefile dependencies have not been calculated for **cobble.c**:



Click OK to continue.

Tornado calculates makefile dependencies and proceeds with building the project. If any external dependencies are found, they are automatically included in the project and listed in the External Dependencies folder in the Files view of the workspace.

The build output window displays any errors and warnings. In this case, there are none:

A screenshot of a 'Build Output' window. The window title is 'Build Output'. The text inside shows a series of compilation and linking commands for 'csimpc' and 'ldsimpc'. The commands include flags like '-g -mpentium -ansi -fno-builtin -fno-defer-pop -I. -IC:\Tornado2.2\target\h\ -DCPU=SHBTF -DPOOL_FAMILY=gnu -DPOOL=gnu -c ..\..\cobble.c' and 'ldsimpc -r ..\prjObj.s.lst -o partialImage.o'. The output ends with 'Done.' and a scrollbar is visible at the bottom.

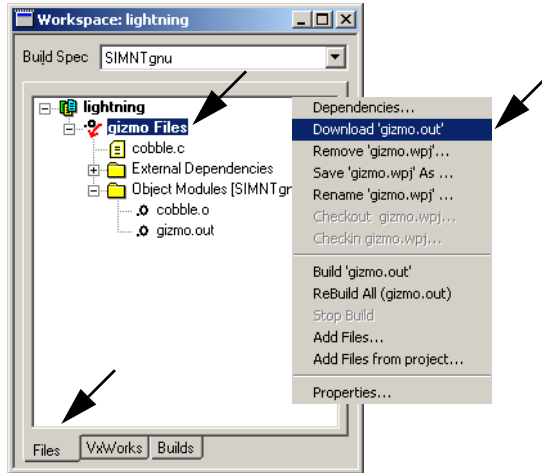
Close the Build Output window with the window control.

3.6 Download the Project to the VxWorks Target Simulator

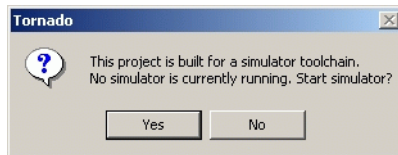
You can download your project from the Files view of the workspace, and start the integrated simulator, all as part of the same process.

First display the Files view of the workspace window.

Then select the project folder name with the right mouse button and select the Download 'gizmo.out' option from the context menu:

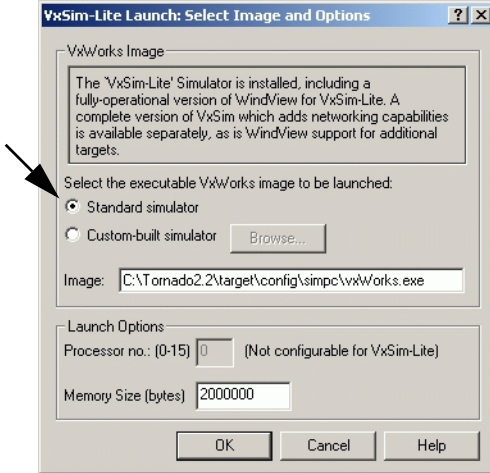


Tornado prompts you to start a simulator:



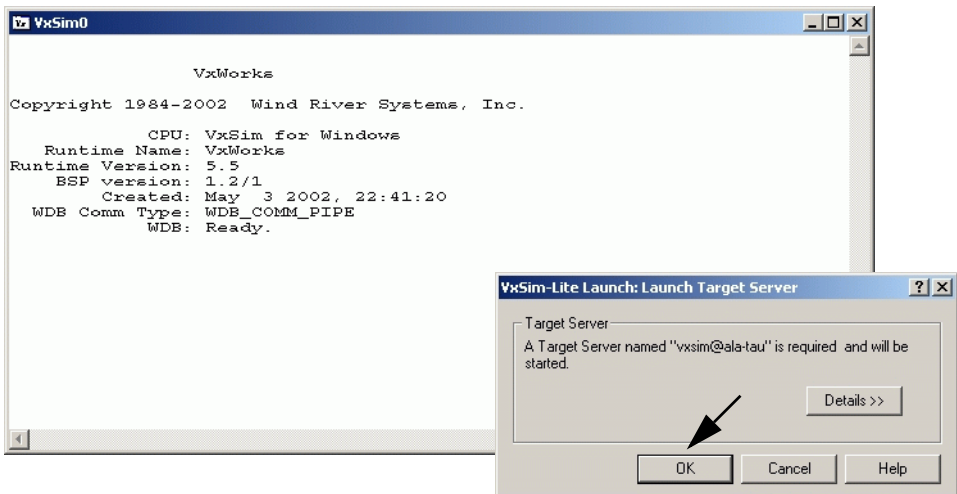
Click Yes to continue.

The VxSim dialog box appears:



Leave Standard Simulator selected, and click OK to continue.

The target simulator window opens, and Tornado prompts you to start a target server. Click OK in the VxSim-Lite Launch dialog to continue:





NOTE: Only one instance of the integrated VxWorks target simulator (VxSim-Lite) can be run at a time. To stop the simulator, close the VxWorks Simulator window.

A target server manages all communication between Tornado host tools, such as the debugger, and the target. The convention for target server names is *targetName@hostName*. In this example, the name of the target is *vxsim* (the default for the integrated simulator), and the name of the PC host is *ala-tau*.

The name of the target server is displayed in the Tornado launch toolbar:



The toolbar includes buttons for starting Tornado tools such as the browser, shell, and debugger. The buttons are activated when the target server is started.



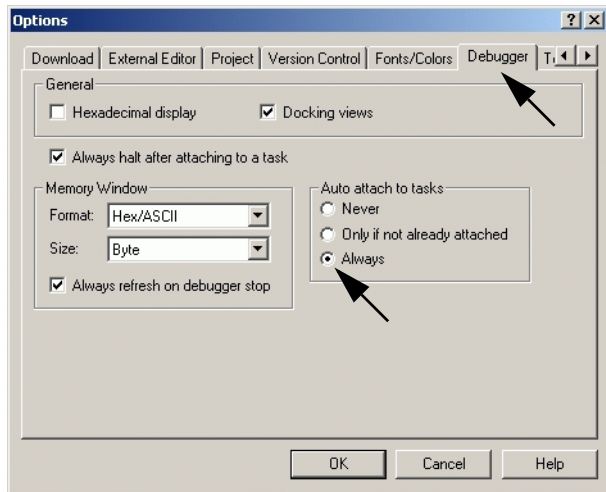
NOTE: Both the browser button and the shell button both have the letter *i* in them, which may be confusing. The *i* in the shell button follows the *->* shell command prompt, where it is a shell command that displays target system information. The *i* in the browser button stands for *information* about the target.

3.7 Run the Application from the Tornado Shell


Before you run an application, it is often useful to configure and start the Tornado debugger, so that the debugger can respond automatically to any program exception.

The Tornado debugger (CrossWind) combines the best features of graphical and command-line debugging interfaces. The most common debugging activities, such as setting breakpoints and controlling program execution, are available through convenient point-and-click interfaces. Similarly, program listings and data-inspection windows provide an immediate visual context for the crucial portions of your application.

To configure the debugger, select Tools>Options>Debugger from the main Tornado window. When the Options dialog appears, select Always under Auto attach to tasks, so that the debugger will automatically attach to a task when an exception occurs:




Click OK to continue. For more information, see the *Tornado User's Guide: Debugger*.

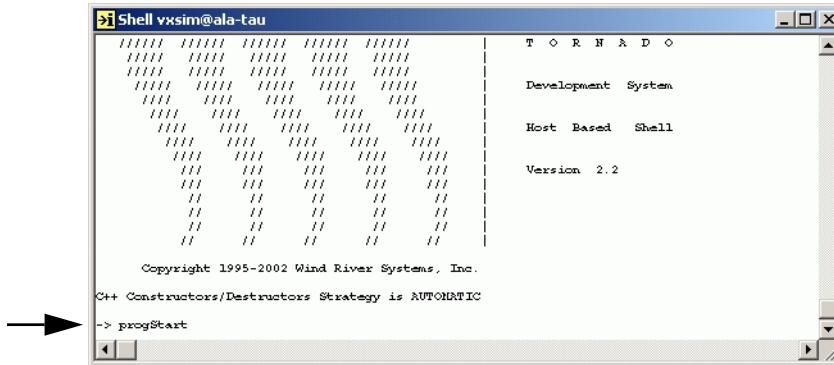
To start the debugger, use the debugger button  in the Tornado launch toolbar. The back trace and variables windows open (you may close them), and the status line at the bottom of the main Tornado window announces that the debugger is running:



The Tornado shell (also known as WindSh) is a C-language command interpreter that allows you to invoke any routine from the shell command line that has been downloaded to the target. The shell also includes its own set of commands for managing tasks, accessing system information, debugging, and so on.


You can run the program from the shell command line. To start a shell, click the shell button .


When the shell window opens, run the program by entering the name of the main routine, **progStart**, at the command line:

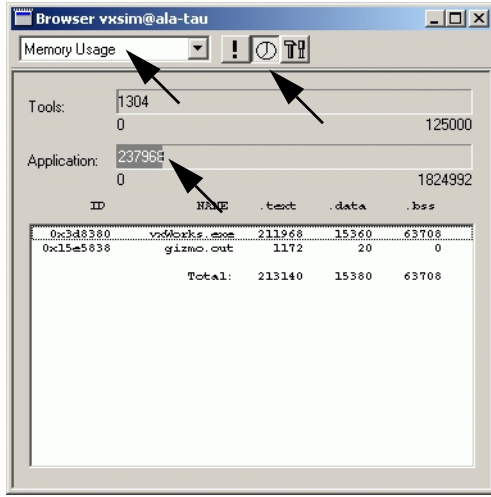


3.8 Examine Target Memory Consumption

The Tornado browser is a system-object viewer that is a graphical companion to the Tornado shell. The browser provides display facilities to monitor the state of the target system, including summaries of active tasks, memory allocation, and so on.

To start the browser, click the browser button  in the Tornado launch toolbar.


When the browser appears, Select Memory Usage from the drop down list and click the periodic refresh button . This will update the display every few seconds, and you will note that **cobble.c** has a voracious appetite for memory:

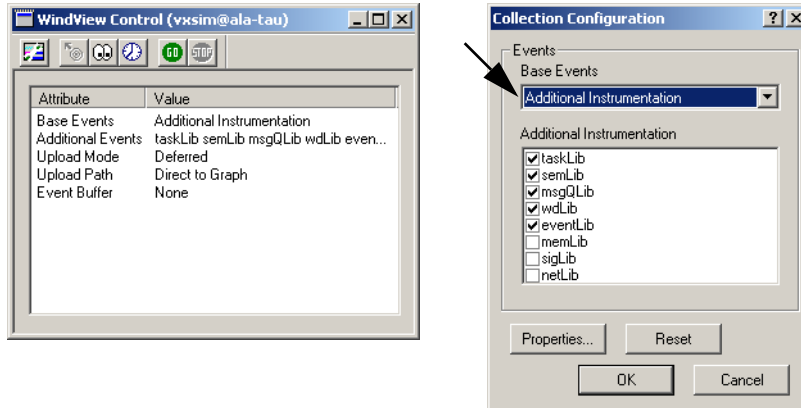



Stop the browser with the window control in the title bar.

3.9 Examine Task Activity

WindView is the Tornado logic analyzer for real-time applications. It is a dynamic visualization tool that provides information about context switches, and the events that lead to them, as well as information about instrumented objects such as semaphores, message queues, and watchdog timers.

Click the WindView button  in the Tornado launch toolbar to display the WindView Control window. The WindView Collection Configuration dialog box also appears. Select Additional Instrumentation from the drop-down list and click OK:

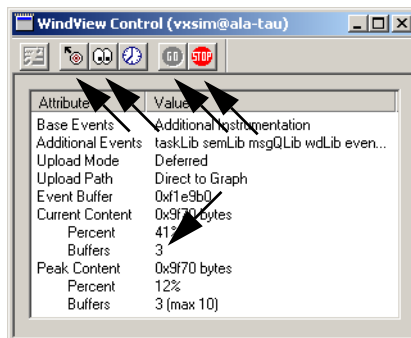



To begin data collection, click the GO button  in the WindView Control window.



NOTE: Except where otherwise specified, the default configuration of WindView is used for this exercise. If WindView settings have been changed during another session, the behavior may vary from the description below.


Wait a few seconds, and then click the update button  in the WindView Control window to update the status of data collection:



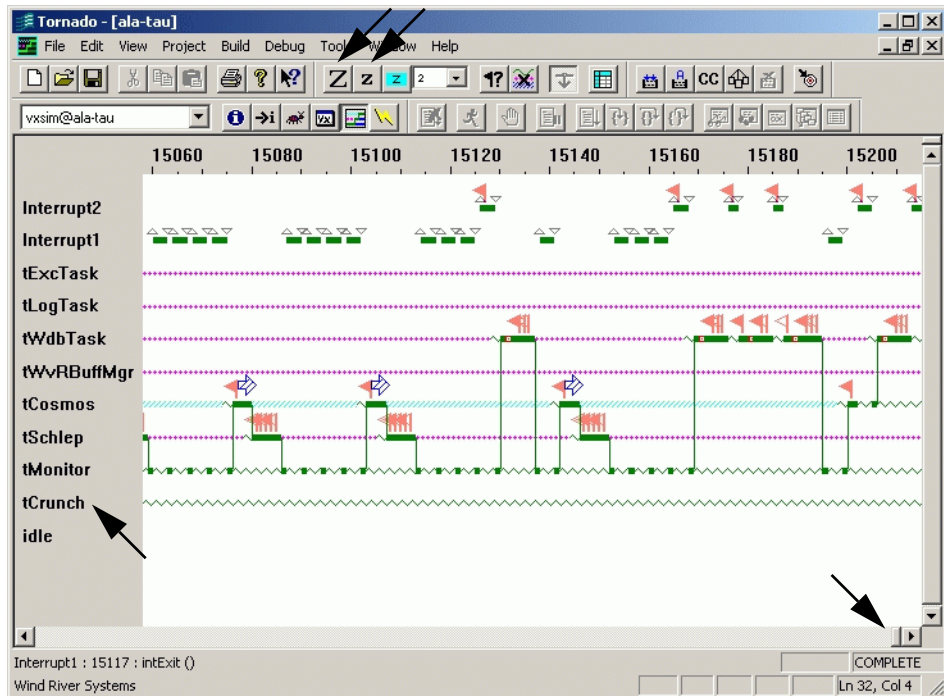
When the Buffers value listed under Current Content reaches at least two, click the Stop button  to end data collection.




You may notice a certain sluggishness in your machine's performance, which is due to the sample program's runtime misbehavior on the integrated simulator.

Before you upload the WindView data from the simulator target to the host, stop the sample program by entering **progStop** at the shell command line.

Then use the upload button  to upload the data. A view graph is displayed while the data is uploaded.

When you maximize the view graph window, the data display should look something like the following:¹



You can use the zoom buttons   on the WindView toolbar to enlarge or decrease the size of the scope of the display, and scroll bar at the bottom of the view graph to display the rest of the data. (The zoom 100% button  displays all of the

1. The color of the view graph display can be changed with Tools>WindView>Options.

data at once.) The flag-shaped icons indicate semaphore gives and takes. The horizontal lines indicate task states (executing, pending, ready, and so on). For information about the meaning of the WindView icons, select Help>WindView Help>Legend.

Note that **tCrunch**, the task that processes data and removes nodes from a linked list, never runs.

Close the view graph and click No in response to the dialog that prompts you to save the data with: Save changes to *hostName*?

Close the WindView Control window as well.

3.10 Fix Task Priorities and Find the Next Bug

Open the source code in the editor by double-clicking on **cobble.c** in the workspace Files view. Then find the **progStart()** routine, where you'll discover that **tCrunch** is assigned a lower priority (240) than **tMonitor** (230), which never allows it to run. As a consequence, data is never processed, and nodes are never deleted from the linked list.

Edit the source file to reverse the priorities between the two tasks:


- Change the 240 argument in the **taskSpawn()** call for **tCrunch** to 230.
- Change the 230 argument in the **taskSpawn()** call for **tMonitor** to 240.

Then save the file.

Use the context menu in the Files view of the workspace, select the ReBuild All (gizmo.out) option to rebuild the project. When it is done, close the Build Output window.

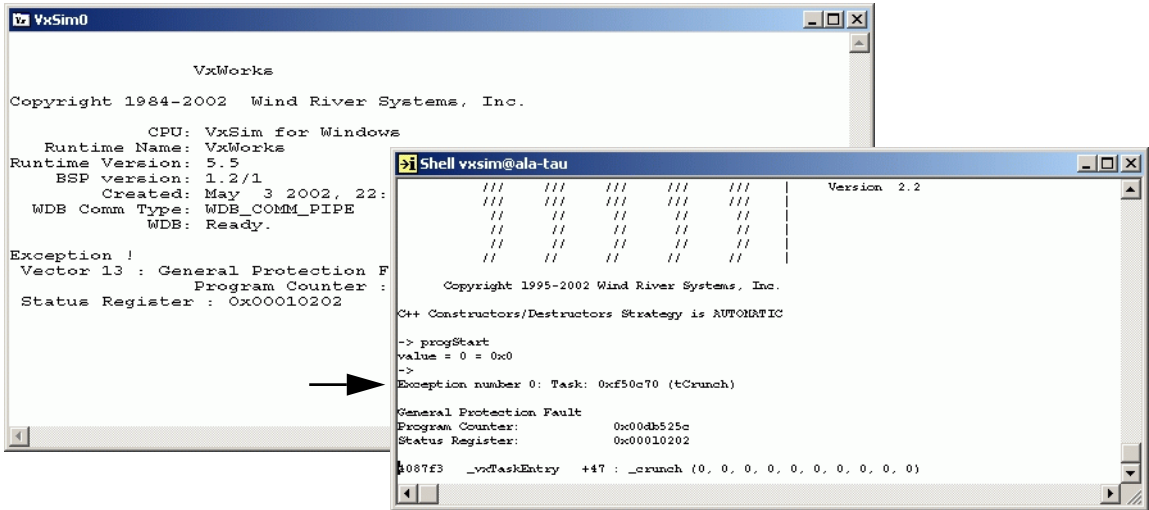
Then download the project to the target with the Download 'gizmo.out' option from the context menu.



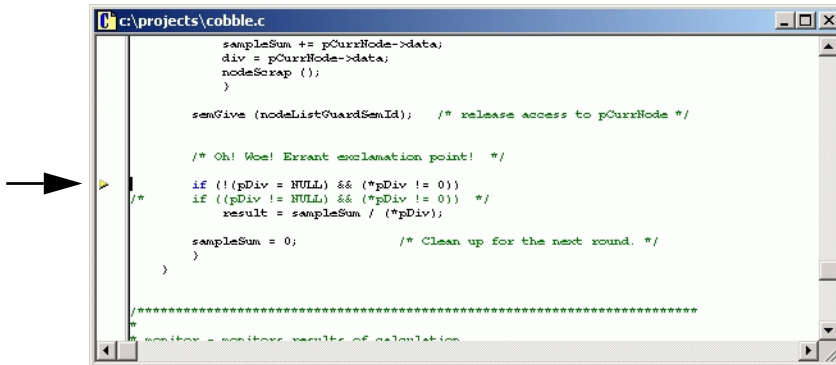
NOTE: Do not forget to download the new **gizmo.out** to the target! And if you have stopped the debugger, restart it with the debugger button .

Start the program again from the Tornado shell with the command **progStart**.

Both the simulator and shell windows soon announce an exception and the number of the task in which it occurred:



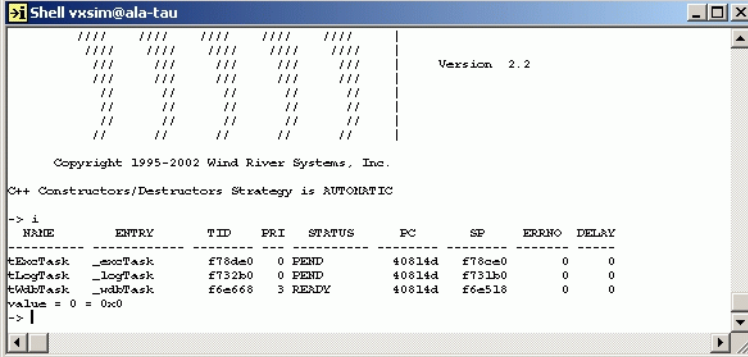
In addition, the debugger opens the editor window automatically with a context pointer marking the position in the `crunch()` routine where the error was generated:



Before you proceed, reboot the target simulator: in the shell window, press **RETURN** to display the prompt, and enter **reboot**.

A message box announces that the target connection has been lost, and that the debugger is being stopped. Click OK.

If you use the `i` command in the shell window after you have rebooted the simulator, you see that only VxWorks system tasks are now running:



```

Shell vxsim@ala-tau
|||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||
|||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||
|||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||
|||    |||    |||    |||    |||    |||    |||    |||    |||    |||    |||
||     ||     ||     ||     ||     ||     ||     ||     ||     ||     ||
||    ||    ||    ||    ||    ||    ||    ||    ||    ||    ||    ||
||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||   ||
||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||
||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||  ||

Version 2.2

Copyright 1995-2002 Wind River Systems, Inc.

C++ Constructors/Destructors Strategy is AUTOMATIC

-> i
-----
NAME          ENTRY          TID    PRI    STATUS    PC      SP      ERRNO  DELAY
-----
tExecTask    _execTask       f78de0 0    PEND     40814d f78ce0    0      0
tLogTask     _logTask        f732b0 0    PEND     40814d f731b0    0      0
tWebTask     _webTask        f6e668 3    READY   40814d f6e518    0      0
value = 0 = 0xc0
-> |


```

3.11 Fix the Last Bug and Take it for a Spin

Open the `cobble.c` file in your editor by double-clicking on the file name in the Files view of the workspace window. Fix the source of the exception error in the `crunch()` routine by using the commented code immediately below the division operation—or something more interesting. Save the file and close the Source view window.

Then use the context menu in the Files view of the workspace to select the ReBuild All (gizmo.out) option. When the build is done, close the Build Output window.


Download the application to the target with the Download 'gizmo.out' option from the context menu.


To be sure that everything is working, start the debugger with the debugger button .

Start the program again from the Tornado shell with the command `progStart`.

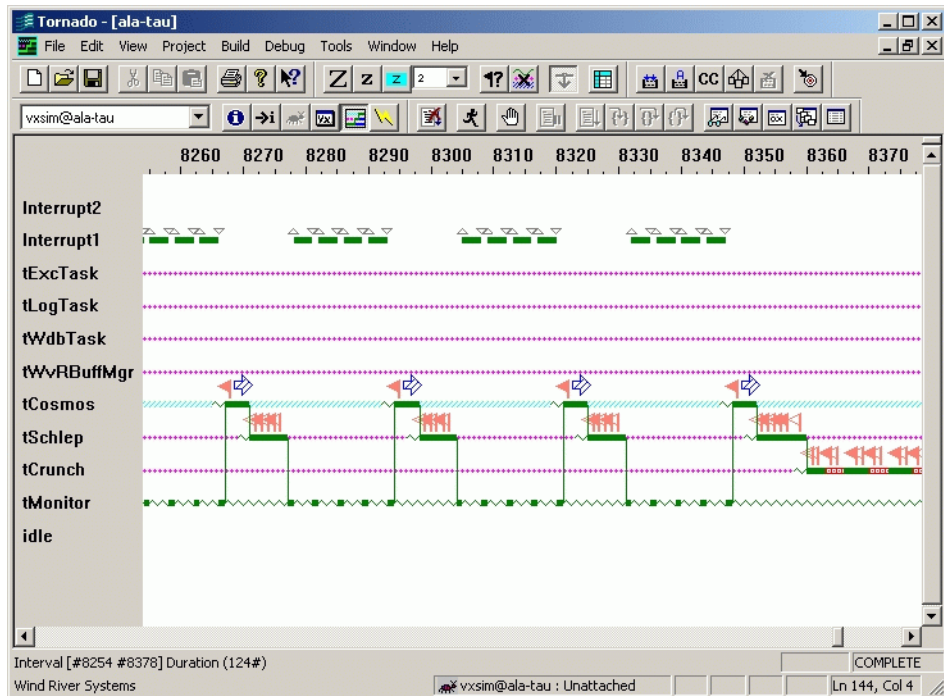
Click on the Windview button  to display the WindView Control window. Then click the GO button  in the WindView Control window to begin data collection.


Wait a few seconds, and click the update button  in the WindView Control window to update the status of data collection.


When the Buffers value listed under Current Content reaches at least two, click the Stop button  to end data collection.

Then use the upload button  to upload the data. A view graph is displayed while the data is uploaded.

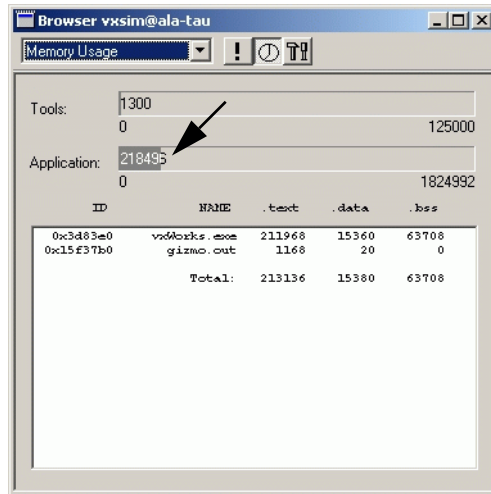
Maximize the view graph window and scroll through the WindView data. You should find that all tasks, including **tCrunch**, are running in an orderly manner, and the view graph data should look like the following:



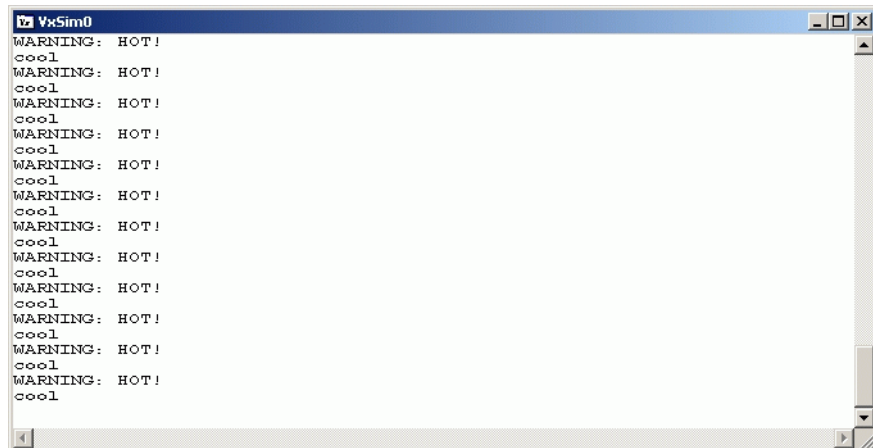
You can check memory consumption once again with the browser by clicking the browser button  in the Tornado launch toolbar.

When the browser appears, select Memory Usage from the drop-down list and click the browser's periodic refresh button . Every few seconds the display will be updated.

Notice that memory consumption fluctuates modestly within a constant range:



Moreover, output in the simulator window indicates that the program is humming along nicely, providing you with information about when it's hot and when it's not:



Stop the program by entering **progStop** at the shell command line.

Bye! Tschuess! Adieu!

3.12 What Next?

For detailed information about using features of the Tornado IDE such as the project facility, shell, browser, and debugger, see the *Tornado User's Guide*. The guide also provides information about setting up your Tornado development environment with target hardware.

For information about VxWorks itself, see the *VxWorks Programmer's Guide*, the *Network Programmer's Guide*, and the *VxWorks API Reference*.

The Tornado documentation set is available online in HTML format from the main Tornado menu Help>Manuals Contents, or directly from the top level HTML file `installDir\tornado\docs\books.html`.