VxWorks Real-Time Kernel Connectivity

Andrew Piñeiro
CNT 4104
Janusz Zalewski
FGCU
December 9, 2011
1. Introduction

**VxWorks** is a proprietary **real-time operating system** (RTOS) made and sold by Wind River Systems (now a wholly owned subsidiary of Intel). It is the most popular RTOS for embedded systems. [1] The Workbench IDE (also created by Wind River) is used to develop VxWorks programs. Development of VxWorks programs must take place on a computer running a different operating system (such as UNIX or Windows). From this **host machine**, the executable can be ported to a computer (called the **target**) that runs VxWorks. The reason for this is the target, such as a microcontroller of an embedded system, will most likely not have the resources to do the compiling. [2] For this reason, a VxWorks system may never be self-hosting. VxWorks programs can be **cross-compiled** to run on various target CPU architectures.

Cross-compiling is the act of creating different executables for different platforms, other than the one where the compiling is originally done. A compiler that can do this is called a cross-compiler. As previously mentioned, the microcontroller of an embedded system will not have sufficient resources to do the compiling itself. Consequently, cross-compiling is highly used for embedded systems. [2]

In this project, the work of previous-semester projects is continued. [5] A web server that can run on a VxWorks **SBC** (single-board computer) is created. This particular SBC is also made by Wind River (the **Wind River SBC8349E**).
2. Problem Definition

The projects from previous semesters have failed to establish a target server connection with the Wind River SBC. (A target server connection is the name used in Workbench for a connection between the host computer and the computer running VxWorks. In this case, the VxWorks computer is the Wind River SBC8349E.) However their code was able to work with VxSim (a VxWorks target simulator). So the first objective for this project is to establish a successful target server connection with the actual hardware. Then a simple Hello World program is to be created and run on the target to show that everything works correctly. Only then will there be an attempt to get the web server to run on the SBC.

Since this is the first time working with VxWorks, the most challenging problems of this project are:

- Figuring out how to use Workbench
- Figuring out how to develop VxWorks programs
3. Solution to the Problem

To develop the web server to run on the SBC, the following documentation has to be used:

- VxWorks Kernel Programmer’s Guide 6.7 [9]
- Wind River Workbench By Example 3.1 (VxWorks 6 Version) [8]
- Tornado and VxWorks What’s not in the Manual [10]

All equipment and software required to work on this project is already available:

- The Workbench IDE is installed on a host computer (HP desktop running Windows XP with an IP address of 69.88.163.21)
- A Wind River target SBC with an assigned IP address of 69.88.163.22
- Respective serial and Ethernet cables to make the required connections
- And an Internet connection allowing necessary communication between the host and the target

The system configuration is shown in Figure 1 below.

![Figure 1: The layout of all the hardware and software components.](image-url)
4. Implementation

4.1 Hardware Setup and Booting

The type of SBC used for this project is a Wind River SBC8349E, shown in Figure 2. The exact specification of the SBC is as follows. [3]

- Core processor
  - MPC8349E 667MHz processor

- Available memory
  - 256MB DDR SDRAM SODIMM
  - 8MB on-board flash
  - 8KB EEPROM
  - 128MB local bus SDRAM DIMM
  - 256K-bit I2C serial EEPROM

- Support for the following PowerQUICC II Pro peripherals
  - PCI-X controller: 3.3v 64-bit PCI bus up to 133MHz
  - Memory controller: up to DDR333 memory bus
  - Interrupt controller and timers
  - Two 10/100/1000BaseT internal Ethernet MAC interfaces using RJ45 connectors
  - Two RS-232 interfaces using Mini DB9 connectors
  - USB Host Mini A/B port

- Hard reset via a momentary switch
- Seven-segment display on CPU GPIO
- User switches on CPU GPIO
- 16-pin and 52-pin JTAG debug connections
To set up the SBC, the following steps have to be followed:

1. Make sure that Workbench version 3.1 and VxWorks version 6.7 are available on the host.
2. Connect the SBC and host computer via a serial cable. (The host uses the serial connection to communicate with the boot loader on the target. [7])
3. Connect the SBC and host computer via an Ethernet cable. (In this project, both machines were connected to a wall jack, and thus connected to the school network. The host uses the Ethernet connection to transfer files, including the VxWorks kernel. [7])
4. Power on the host computer (don’t power on the SBC yet)
5. On the host computer:
   a. Open up Workbench (a screenshot of Workbench is shown in Figure 3)
   b. Open up the **WFTPD** server (by default, it should be under the Wind River folder in the Start Menu’s list of programs). This is required for the SBC to download
the VxWorks kernel over the Ethernet connection. [7] A screenshot of it is shown in Figure 4.

6. In Workbench, open up the **Terminal view** (select Window → Show View → Terminal)

7. In Terminal view, set up a serial connection between the host and SBC.

8. Power on the SBC.

9. Communication between the SBC and host soon begins and is shown in the Terminal view (the boot loader’s output is shown in Figure 5)

10. After a countdown, the boot loader chooses a previously configured (see next section) VxWorks kernel to boot. Then VxWorks starts up on the SBC. WFTPD shows the boot loader logging in and downloading a VxWorks kernel (see Figure 4)

![Figure 3: The start page of Wind River Workbench.](image-url)
4.2 Configuring the VxWorks Kernel

When the VxWorks kernel is loaded into the SBC for the first time, it boots up normally, but has no command-line interface. By default, VxWorks Image projects (VIPs) do not set the kernel to have a shell. Also, the VIP bundle “Real Time Process Development” is not set by default. In this project, the programs are loaded into the SBC as Real-time Process (RTP) projects.
Therefore it is necessary to tweak the kernel configuration. The following steps are needed to configure the VxWorks kernel correctly:

1. Open up the VIP folder in **Project Explorer** that needs to be configured
2. Double-click **Kernel Configuration**
3. Select the **Bundles** tab
4. For each required bundle, right-click and select **Add**. In this case, the required bundles are:
   a. Real Time Process Development
   b. Standalone kernel shell
5. After adding the desired bundles, save the project and rebuild it

A screenshot of the kernel configuration used in this project is shown in Figure 6.

![Kernel Configuration Screenshot](image)

**Figure 6:** The kernel configuration. Bundles Real Time Process Development and standalone kernel shell are added.
4.3 Running a Hello World RTP Application

Since VxWorks is a real-time operating system, any process that runs on it is an RTP. However, Workbench makes the distinction that RTP projects run in user space. Projects that create a VxWorks OS image or kernel modules run in kernel space. [5]

After creating a new RTP Project, the Hello World program is coded. Before building the project, one has to make sure that the RTP is compatible with the kernel loaded into the SBC. To do this, check the **build properties** of the kernel project in Workbench. In Workbench, kernel projects are referred to as **VIPs** (VxWorks Image projects). In this case, the name of the VIP is “Test”. A screenshot of its build properties is shown in Figure 7.

![Figure 7: The build properties of a VIP (VxWorks Image project).](image)
What needs to be checked is the “Build configuration for real time processes and shared library subprojects” section of the VIP’s build properties. To get there, the following steps need to be followed:

1. Right-click the project in the Project Explorer window and select Properties
2. Select Build Properties
3. Select Subprojects
4. Examine “Build configuration for real time processes and shared library subprojects”

The build configuration of the RTP must match the build spec in the VIP’s “Build configuration for real time processes and shared library subprojects” section. In this case, the build spec is “PPC32gnu_RTP”.

To modify the build configuration of the Hello World RTP, the following needs to be done:

1. Right-click the project in the Project Explorer window and select Properties
2. Select Build Properties
3. Select Build Support and Specs
4. Examine “Default build spec” (a screenshot of the Hello World RTP build configuration is shown in Figure 8)

Now the build spec from the drop list that matches the kernel’s build spec for real-time processes must be selected. Then building the RTP can be done.
Once the Hello World executable is built, it’s time to load it into the SBC. In VxWorks, the file type for an executable is “.vxe”. In this case, the Hello World .vxe is located on the host computer. So the SBC would have to download the .vxe via WFTPD. This can be done in the Terminal view of Workbench. If the kernel on the SBC is configured correctly, there should be a command-line prompt in the Terminal view. To load and run the Hello World .vxe on the SBC, the following steps need to be followed:

1. The command-line prompt may start out as the C interpreter by default. To switch to the command interpreter, the `cmd` command must be used. [6]
2. The `rtp exec` command must be used to execute the Hello World real-time process. In this case, it uses a full path to the location of the executable on the host:
Figure 9 shows a screenshot of the Terminal view, which contains the command-line.

![Terminal View Screenshot]

Figure 9: Workbench’s Terminal view is used as a command-line interface for the SBC.

4.4 Running the Web Serve RTP Application

For the web server RTP, Andon Coleman’s eHTTP server is used. [5] This server program was created in the previous semester (Spring 2011). The source code was stored in a Downloadable Kernel Module project. But since RTP projects are now being used, the source code was copied
into an RTP project named **eHTTP_Test**. Some changes needed to be done to the source code in order to get it to compile. Mostly it was commenting out unnecessary C preprocessor lines.

As with the Hello Word RTP, the project’s Build Properties window needed to have “**PPC32gnu_RTP**” set as the build spec. The same instructions in the Hello World section for changing the build spec need to be followed. Afterwards, the eHTTP RTP can be compiled. Then it can be executed with the **rtp exec** command in the Terminal view. The exact command in this case is as follows:

```
rtp exec
JZ-HP-
WINXP:C:/WindRiver_OCD/workspace/eHTTP_Test/PPC32gnu_RTP/eHTTP_Test/Debug/eHTTP_Test.vxe
```

Figure 10 shows WFTPD’s output when the SBC requests eHTTP_Test.vxe.

![Figure 10](image)

```
[9 356:14 Connection accepted from 69.88.163.22
[9 356:14 Command "USER spinoiro" received
[9 356:14 PASSword accepted
[9 356:14 User spinoiro logged in.
[9 356:14 Command "TYPE I" received
[9 356:14 TYPE set to I
[9 356:14 Command "PASS" received
[9 356:14 Entering Passive Mode (69,88,163,22,1,1)
[9 356:14 RETR C:\WindRiver_OCD\workspace\eHTTP_Test\PPC32gnu_RTP\eHTTP_Test\Debug\eHTTP_Test.vxe received
[9 356:14 RETRiVe started on file C:\WindRiver_OCD\workspace\eHTTP_Test\PPC32gnu_RTP\eHTTP_Test\Debug\eHTTP_Test.vxe
[9 356:14 Transfer finished
[9 356:14 GetFile C:\WindRiver_OCD\workspace\eHTTP_Test\PPC32gnu_RTP\eHTTP_Test\Debug\eHTTP_Test.vxe successfully
[9 356:14 Command "QUIT" received
[9 356:14 QUIT or close - user spinoiro logged out
```

**Figure 10**: eHTTP_Test.vxe gets loaded into the SBC successfully.

Now that eHTTP is running on the SBC, it is ready to accept HTTP requests. It was tested by using Mozilla Firefox to request a web page from it. In this case, the IP address of the SBC was **69.88.163.22**. The output for the request “**http://69.88.163.22/index.html**” is shown in Figure 11.
Figure 11: Despite not having the specified file, eHTTP still responds to the web browser successfully.

As one can see, the page index.html does not exist. But the primary goal has been met; getting the web server to run on the SBC instead of the VxSim simulator. Figure 12 shows the Terminal view’s output from the Firefox HTTP request.

Figure 12: The output of eHTTP in the Terminal view. eHTTP can even detect the web browser’s name and version number.
Figure 13 shows the output from WFTPD when Firefox makes the request. The SBC interacts with WFTPD, implying that eHTTP’s file system is still stored on the host computer.

One might be wondering why the SBC still interacts with the host after eHTTP gets loaded and is running. The only explanation for this is that eHTTP’s file system is still located in the host. When Firefox requested `index.html`, eHTTP checked its file system in the host for `index.html`. But since there was no `index.html`, WFTPD outputted “RETR failed – file does not exist”.
5. Conclusion

The main objective of getting the VxWorks programs to run on the SBC was accomplished. The projects from previous semesters have succeeded in getting their programs to run on the VxSim simulator. But now future students can execute their programs on the actual hardware. It took quite a bit of digging through documentation. But at last, there was success. For future students assigned to this project, it is highly recommend that the documentations listed in the “Solution to the Problem” section be referenced. [6, 7, 8, 9, and 10]

In this report, emphasis was placed on the problems that took quite some time to solve. One example is figuring out why the Hello World RTP wouldn’t run in the SBC. But finally it was discovered that one had to set the RTP’s build spec to match the VIP’s build spec for real-time processes. Step-by-step solutions to these problems are described in this report.

Future improvements to this project would be moving eHTTP’s file system to the SBC. That way once eHTTP is loaded and running on the SBC, there will be no more dependency on the host computer. Another thing would be to load some actual web pages into eHTTP’s file system. This would be ideal for fully testing eHTTP. And it may even expose some bugs in it that need to be fixed.
6. References

   http://en.wikipedia.org/wiki/VxWorks
   http://en.wikipedia.org/wiki/Cross-compilation
   Retrieved 10-7-11.
   http://itech.fgcu.edu/faculty/zalewski/CNT4104/Projects/ColemanSTABLE_FinalV2.pdf
   Retrieved 10-7-11.
   http://itech.fgcu.edu/faculty/zalewski/CNT4104/Projects/Coleman_HTTP_Draft5.pdf
   Retrieved 10-9-11.
   Retrieved 10-9-11.
   Published 12-3-08. Retrieved 10-9-11.
   http://www.scribd.com/doc/49033052/wr-workbench-vxworks-6-by-example-3-1
   Retrieved 10-9-11.