

# TORNADO Evaluation

This article presents an executive summary of the overall evaluation of Wind River's Tornado. The idea is to give an impression not only of the operating system but also of the development environment. We examine the installation and configuration process, the kernel, the different tools supplied and the documentation. Furthermore, in the conclusion, we talk about the type of applications the RTOS is suitable for.

## INTRODUCTION

The following article, as announced in the VRTX evaluation article edited in the last magazine, is more an overview of the capabilities of Tornado as a development environment and as a real-time operating system in a real working environment than a thorough evaluation of the RTOS itself. A more detailed evaluation of the RTOS will be available as a report later this year.

In this article, we follow the outline proposed in the previous one, but for the clarity of the text, we will remind the reasons for each examined point.

## DESCRIPTION

Wind River's Tornado is composed of two products, a cross-development environment and a real-time operating system (VxWorks). The host system, i.e. the development environment, and the target system are connected to allow basic manipulation like downloading and debugging applications. Moreover Tornado has been conceived for using tools to analyse and monitor the target system with little intrusion. Some of target-dependent tools (i.e. the shell, symbol table) have been shifted to the host system to consume less target memory.

## EVALUATION CONDITIONS

### Product version

The version of the evaluated product is not the latest one, but there were no major releases during the project.

VxWorks 5.3

Tornado 1.0 for Windows NT

GNU ToolKit 2.6

WindView 1.0.1

StethoScope 1.0

### Environment

We used the following hardware during the evaluation:

- Two Motorola MVME162 model 533 (68040 at 33 MHz with 16Mb Ram) boards as target systems with Ethernet support;
- A VME bus analyser from VMETRO to perform the interrupt-related measures;
- A Pentium 133MHz running Windows NT 4.0 as host system.

The VxWorks kernel had been configured to support only TCP/IP networking.

## Framework

The evaluation was part of a project lasting about 10 man-months and involving 3 people.

## INSTALLATION AND CONFIGURATION

It is important that the installation and configuration are easy and clear. This presents two benefits: the user would not spend valuable time on deciding on the correct option to choose, and there would be less support requests for the vendor. In this, Tornado for Windows NT is ahead of most UNIX products since it is based on the commonly used installation wizard. This tool makes the setup easy and fast. The program goes through a series of detailed questions that the user's guide exemplifies. For instance, the manual clearly explains how to setup the Tornado registry, a service specific to Windows NT.

The configuration of the host system is rendered easy by the use of the Windows graphical environment (i.e. dialog boxes, lists, checkboxes, buttons...). This is a major enhancement from the command line interfaces and script files used in a UNIX environment. For instance, Wind River Systems provides a graphical tool to configure VxWorks so it matches the board and application requirements. For each feature of the kernel that can be included or excluded the tool gives a short description. Unfortunately the description is not specific about dependencies between options. One only becomes aware of the missing one at compile-time.

Although the configuration is good, it is unfortunate that there is an absence of an interface to configure the virtual memory and an interface to set VMEbus options. For changing the virtual memory configuration and the VMEbus parameters to allow onboard access from the VMEbus, different source and header files have to be updated. This might be a little annoying when developing a multi-VME-board application.

## RTOS RICHNESS

### Kernel

The kernel is highly scalable as more than 80 options are available. This allows the developer to include just the required functionality into the kernel and therefore reduce its size to a minimum.

### Application programmer interface

A common fact known to developers is that the more system calls the RTOS includes, the shorter the devel-

opment time for an application is. Wind River's RTOS has a great number of system calls when one takes into account device drivers. The Application Programmer Interface (API) is diversified as different standards are supported (POSIX, TCP/IP, SCSI...). Taking a closer look at the native API, one realises that the implementation of the communication and synchronisation objects lacks some basic mechanisms such as mailboxes (single-element queue) and events. For instance one might think of an application where tasks are synchronised on different events. Without the use of event flags that can be manipulated by groups, one has to develop a mechanism to synchronise those tasks all together. The single-element queues (that we call mailboxes) are especially useful if the receiving task only needs the most recent message to run.

To quantify the RTOS richness, we have listed the features for the most common system calls and we compared those that are included in VxWorks to this list. Table 1 shows an extract of the list, which is fully available in the report.

Table 2 is the summary of the RTOS richness list. It shows the percentages of the native system calls that are available out of all those that could have been

Mutex		Yes
	Receive non blocking	Yes
	Receive with timeout	Yes
	Pend in FIFO order	Yes
	Pend in priority order	Yes
	Inquiry	Yes
	Priority inversion	Yes
	Task deletion safety	Yes
	Object deletion safety	No
Total		92%

Table 1. Mutex system calls

included for the considered mechanisms. The calculated total is only of about 57%. The conclusion is that the API is quite restrictive. Nevertheless, this should not be misunderstood; the scale only includes the basic system calls, not the system calls related to device drivers.

The POSIX 1003.1b Real-Time Extensions API is complete. VxWorks also offers C++ compatibility with the Wind Foundation Classes library, which includes RogueWave software.

### Standard device drivers

Developers need a RTOS that supports a wide range of device drivers, as most applications developed nowadays are highly focused on connectivity. Within the industry of embedded systems many bus standards are used to obtain this connectivity. Moreover, the decision on choosing an operating system partly relies on its scalability since it will probably be used for more than one project. VxWorks meets these demands as it supports a large variety of standard dri-

Mechanism	System calls
Mailboxes	0%
Mutex	92%
Semaphore	89%
Control shared variables	0%
Event flags	0%
Memory partition	0%
Memory heap	80%
Task ( or thread) management	100%
Timing functions	87%

Table 2. RTOS richness

vers, amongst them fieldbus drivers (IP protocols, CAN protocols) and network drivers (FDDI, ATM,...). A large number of device drivers are available from hardware constructors and third-party vendors. See the product directories on the GRoupIPC web site to learn more about the hardware supported by VxWorks.

For instance, the network interface is comprehensive making the OS suitable for connectivity-oriented applications. The TCP/IP protocol is implemented on top of various physical layers like Ethernet, VME backplane, and serial lines... Different applications and protocols related to the internet protocol such as remote procedure call (RPC), remote login (rlogin, telnet), network file system (NFS), file transfer protocol (TFTP), SNMP and BOOTP are available.

To conclude this section, we want to mention that Wind River follows the general tendency of embedded internet applications??? by releasing an package including Java support, an embedded HTTP server and a web browser.

## PERFORMANCE EVALUATION

### Network

It is surely not sufficient to have an extensive network interface to develop network-oriented applications. To make such applications run fast, the interface must reach high bandwidth utilisation. That is why we use a test application measuring the network throughput and the generated CPU load to quantify the performance level of the network library. The application is based on the client/server model. The measurements are performed within the client, which is sending data to the server using TCP/IP on a quiet Ethernet. The client and the server run on two different MVME162 boards, both under VxWorks.

Figure 1 represents both network throughput and CPU load as a function of the packet size. As one can see from the figure the network performance is excellent since almost 80% of the Ethernet bandwidth can be used with less than 80% of CPU for TCP packets exceeding 512 bytes. Note that sending larger packets is more worthwhile because the CPU load/packet size ratio lowers.

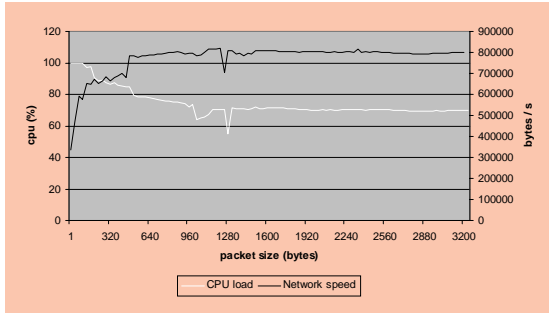


Figure 1. Network transfer rate

## Interrupt handling

Hardware interrupt handling is significant for real-time systems, because it is a means to deal with external stimuli. Developers expect to have fast and predictable handling of interrupts, i.e. the time interval from the generated interrupt until the start of the interrupt-related ISR is fast and predictable. As the used target hardware is based on VME boards, we concentrated on interrupts generated on this bus. Thus the calculated time intervals will be greater than if local interrupts were used.

The developed test measures the following two quantities:

- The interrupt latency, i.e. the time interval from an interrupt generated on the VMEbus to the first line in the Interrupt Service Routine (ISR). This indicates how fast a system can respond to an external event;
- The dispatch time, i.e. the time interval from the last line in the ISR to the next running task. This indicates how long it takes to get from interrupt level to task level. The time spend at interrupt level has to be short otherwise lower level interrupts are blocked for too long. Thus not all of the processing can be done at this level. Added to the previous measure and to the time needed to complete the handling at interrupt level, the measure represents the time before terminating the interrupt handling at task level. A simple example of handling at task level is taking a semaphore in response to a specific interrupt.

Figure 2 and Figure 3 are the results of tests where no notification message from the ISR to any task is used. This is done to avoid extra delay. Furthermore a rescheduling is provoked at the end of the ISR. Therefore this measurement represents a more general situation, showing how much time is needed for the context switch and rescheduling only. Note that the cache on the test board has been disabled to avoid that the test loop is in it.

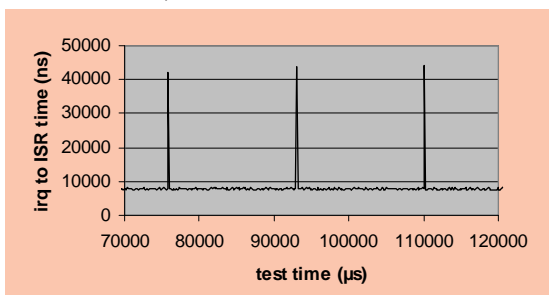


Figure 2. Interrupt latency time (interrupt to ISR)

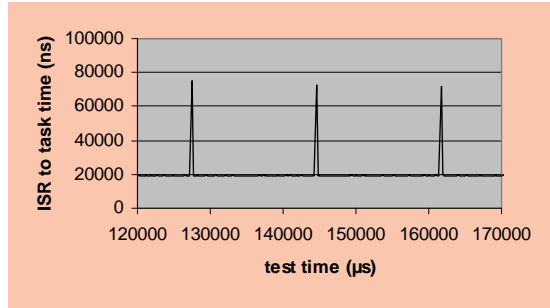


Figure 3. Dispatch time (ISR to task)

Both figures show that the system is fast and predictable. Nevertheless, in Figure 2, one notices the additional time caused by the occurrence of a higher-level interrupt before entering the ISR and in Figure 3, an interrupt occurring when exiting the ISR adds a delay to the dispatch time. In both cases, this higher-level interrupt is generated by the tick timer as the delay between each peak is about 17  $\mu$ s (hence occurring 60 times per second which is the frequency of the timer on a MVM162 board).

Under the test conditions, the average latency time and dispatch time, without any other interrupts, are 10  $\mu$ s and 35  $\mu$ s respectively.

Figure 4 represents the total time from the occurrence of the interrupt on the VMEbus until the first line in the task running after the ISR. This task is pending on a semaphore, which is released in the ISR. Again one notices the extra delay caused by the tick timer interrupt. The average time without taking the higher level interrupt in to account is about 63  $\mu$ s.

Figure 5 presents a summary of our test. We used the same layout as the one presented in (???, ??? and 97Q3). We would like to demonstrate that the graphs presented by vendors do not take into account the delay generated by extra interrupts like the tick timer IRQ. These delays, even if typical, should be clearly stated in the documentation so that the developer can design his application with worst case time figures in mind.

The results of the precedent test show that it is illusory to develop any applications using VxWorks on the MVME162 with time constraints of less than 100  $\mu$ s at task level and less than 50  $\mu$ s at interrupt level. The conclusion is that if the application needs to meet such deadlines one has to use special hardware devices.

Other varieties of performance (i.e. system call through-

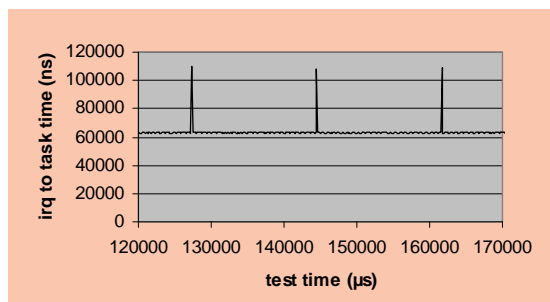


Figure 4. Total interrupt time (interrupt to task)

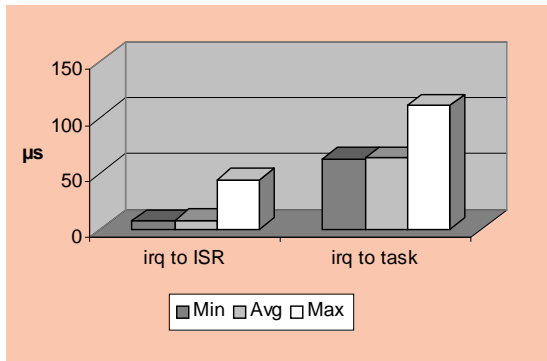


Figure 5. Statistical data

put, context switch time...) are evaluated in the forthcoming report (see Report section below).

## TOOLS

Tornado comes with an Integrated Development Environment (IDE) under Windows NT including a GNU toolkit, a source-code editor, configuration tools and a variety of analysing tools. Most of the graphical user interfaces of the Tornado tools can be customised because they are written in Tcl.

### Editor

The source-code editor includes standard text manipulation capabilities, as well as C/C++ keyword colour highlight. In the version we used, inserting tabulations into the text causes unpredictable behaviour of the editor resulting in text that disappeared.

A negative aspect is the absence of any project manager tool; this means there is no makefile generation utility and no function browser. Such tools can be useful when working on large projects containing many files.

### Debugger

The source-level debugger is an enhanced version of the GDB debugger from the Free Software Foundation (FSF). As it is part of the IDE, the editor window is able to track the code execution. A drawback is that stepping through the code brings the editor window automatically to the top, making it difficult to watch any other window (e.g. watch data, watch execution states...).

It is possible to inspect, in a different window, the value of any object, even complex data structures, if it is in the context of the active function.

### Compiler

The GNU C compiler can compile code written in C, C++ and Objective C for a large numbers of processor families.

### System analysing tools

These tools are an asset when developing real-time applications as they help to track missed deadlines, trace system calls and give information on system objects (i.e. tasks, semaphores, queues...).

The browser views the CPU, memory and stack usage and information on system objects. These values can be updated periodically on demand.

WindView is an enhanced logic analyser visualising

dynamically the evolution of the system. The developer can follow task states.

StethoScope from Real-Time Innovation is a real-time monitor, performance analyser and data collection tool. It lets the developer examine and analyse real-time applications running on a target system.

### Other

The Tornado shell, WindSh, is a host-based shell. It allows the developer to execute any downloaded routine. So it is possible to debug routines before integration. Although the shell may seem UNIX-like, one has to use WindSh's calls instead of the equivalent UNIX call.

## USER FRIENDLINESS

### Documentation

The manuals are well presented: the important facts are highlighted and many diagrams clarify the text.

However, some sections of the programmer's guide are obscure and to understand them one needs a good technical background. Another point is that the indexes are not complete and the references are not always accurate. For instance, it was hard to find explanations on how to boot out of flash memory or how to configure the VMEchip2 of the MVME162 board.

### GUI

Most of the tools have a graphical user interface making them easy to use. Moreover, these environments can be customised using tool command language (Tcl) scripts. This feature is of great use when applied to the debugger. It is possible to automate procedures such as printing the content of a linked list.

## CONCLUSION

The conclusion is divided into two parts: the first one relates our opinion about the product, the second one presents the application profile VxWorks is suitable for. Good points and enhancements needed

The overall impression is that Wind River's product, as well as the IDE as the RTOS, is good, powerful and comprehensive making it widely appreciated.

- The major asset of the product is its capability to accept third-party software including numerous device drivers and system analysing tools;
- The network driver is fast;
- The tool to configure the RTOS is a good feature;
- The synchronisation and communication API is sufficient even if some basic system calls (event flags, mailbox, etc.) are missing;
- The editor is not perfect: de facto Windows editor standards like shortcuts are not always the same;
- A subsequent version should include a project manager to ease the overview of large projects;
- The index of manuals should be completed, as some subjects are hard to find or they are simply not at the indicated reference;
- The Board Support Package (BSP) documentation is too basic and far too incomplete.

## **Application profile**

VxWorks is suitable for a wide range of applications. Network-oriented applications requiring high bandwidth can be based on VxWorks. It is possible to develop small embedded applications as the memory footprint of the kernel can be reduced to a minimum. Finally from the large availability of device drivers, applications can interface with many different devices.

## **Report**

The forthcoming detailed Evaluation Report will include the following sections:

- Ease of installation
- Ease of configuration
- Kernel richness (system calls, priority inversion prevention mechanism, scheduling policies, RMA availability..)
- Compliance to standard (POSIX, Win32, Java..)
- BSP richness (List of supported boards, mezzanines, networks..)
- Memory footprint (ROM, RAM, minimum, per object..)
- Kernel performance and predictability (System Calls,

**Which OS next ?  
Make your choice  
at page 11**

Interrupt latencies, Context switch..)

- Standard device drivers performance (TCP/IP, SCSI, Serial driver..)
- User friendliness (documentation, tools..)
- Standard development tools short evaluation (debugger, profiler, configuration tool..).

Expected availability of the report is 1st quarter of 1998. For more info e-mail to [info@realtime-info.be](mailto:info@realtime-info.be) or consult the RTOS section on our web-site at <http://www.real-time-info.be>.

## **REFERENCES**

Timmerman M., and Monfret J-C., VRTX Evaluation in Real-Time Magazine, RTOS Update Part II issue - 97/3, P.12.

Ready J. F., Designing for Worst-Case: The Impact of Real-Time OS Performance on Real-World Embedded Design in Real-Time Magazine, RTOS Update Part II issue - 97/3, P.52.

Timmerman M., Real-Time Operating Systems Market Review in Real-Time Magazine, INSTRUMENTATION & AUTOMATION Issue - 95/4, P.11.

Timmerman M., and Monfret J-C., RTOS Benchmark Program in Real-Time Magazine, RTOS I Issue - 95/2, P.12.

Cathey D., All Things Considered...Important Factors in Choosing a Real-Time Development System in Real-

Time Magazine, Tools Issue - 93/2, P.87.

Fritch D.G., and Cardoza R.W., Benchmarking Real-Time Systems in Real-Time Magazine, Relaunch Issue - 92/2, P.18.

---

*Dr. Ir. Martin Timmerman is graduated in Telecommunications Engineering at the Royal Military Academy (RMA) Brussels and is Doctor in Applied Science at the Gent State University (1982). He became the director of the System Development Centre (SDC) at RMA, which he created in 1983, and converted himself to a Computer Science Engineer. Actually he is giving general courses on Computer Platforms and more specific courses on System Development Methodologies at RMA.. Outside RMA, Martin is known for his audits, reviews and seminars for his two companies Real-Time Consult and Real-Time User's Support International. RTUSI provides hardware and software support services and is involved in project engineering for Real-Time Systems.*

*Ir. Jean Christophe Monfret graduated in 1993 as an Ingénieur Télécom Bretagne, France. He has a master's degree in Parallel Computer Science from the University of Rennes I, France (1993), a specialisation he had the opportunity to put to practice for a year at the Commissariat à l'Energie Atomique, France. He has been working since 1994 for RTUSI and Real-Time Consult as a project manager where he is also involved in audits and review activities. Laurent Uhres graduated in 1997 as an analyst-programmer from the Haute Ecole Rennequin Sualem, Belgium. He has an additional diploma in Object Oriented Development from the Instituto Politécnico do Porto, Portugal. Laurent is working as a software engineer at Real-Time Consult since mid 1997 where he is involved in the RTOS Evaluation program.*