

VRTX Evaluation

This article is the first product evaluation to be launched in Real-Time Magazine. It presents the evaluation of VRTX as well as the evaluation methodology. For a RTOS, we evaluate the ease of installation and configuration, the kernel and BSP richness, the kernel and standard driver's performances and the user friendliness of the whole product and documentation. At the end we give our conclusion about the RTOS and explain for what type of application it is suitable.

One year ago, we announced our project to benchmark RTOS. We have received many enthusiastic responses, but almost nobody was interested in financing the project. Yet, as we still see from our consulting activities that people are hopelessly looking for such information, we have decided to reconsider the project. It has evolved to a product evaluation rather than a benchmark. By evaluation, we do not intend to test fully and systematically the RTOS. We simply evaluate the RTOS in a real working environment. For this, we ask one of our development teams using a particular RTOS in a particular project to produce an evaluation of the RTOS they are working with. We started this with VRTX. The next RTOS that will be evaluated is VxWorks.

We shall first present the framework of the evaluation, then give the details of the evaluation, which is composed of the following parts:

- Ease of installation,
- Ease of configuration,
- Kernel richness,
- BSP richness,
- Kernel performance,
- Standard device drivers performance (SNX, NFS, SCSI, etc),
- User friendliness (documentation, tools,...),
- Standard development tools short evaluation.

Finally, we shall give our conclusion on the product, what kind of application it is suitable for and what enhancements should be done.

PRODUCT INFORMATION

The products evaluated here are:

- The VRTXsa Real-Time Operating System (RTOS);
- The Microtec Cross compiler, assembler/linker;
- The development suite XRAY MasterWorks.

EVALUATION CONDITIONS

Product version

The products evaluated were not those of the last release. However, the last release is not very different from the one tested here. It should be noted that Microtec history has been full of events over the last few years. The company merged with Ready Systems in 1993 then was acquired by Mentor Graphics in 1995 and finally acquired ONE last year. In such circumstances, the product range evolution has led

more to a merge of products of the different companies than to a fundamental evolution and redesign. We had last month the opportunity to see the last release. Comments concerning this release are included where relevant.

Here is the list of the product's versions:

```
MICROTEC DEVELOPMENT TOOLS VERSION :
  master/license/VERSION : 2.1B
  master/xmo68ks/VERSION : 3.3C
  master/xraymaster/VERSION : 1.2D
  master/asm68k/VERSION : 7.0
  master/cc68k/VERSION : 1.2
  mri/license/VERSION : 2.1B
  mri/mcc68k/VERSION : 4.4
  mri/xhi68kh/VERSION : 3.2
  mri/xhm68k/VERSION : 3.2
  mri/xhs68k/VERSION : 3.4
  mri/xmw_68k/VERSION : 4.4
```

```
SPECTRA TOOLS VERSIONS : Spectra 68K
3.Cab
  Xpert Profiler      3.2
  Xsh                 4.1
  Tsh                 2.0
  Esh                 2.0
  Xconfig             4.0
  *RTsource           4.0
  sysview server     4.0
  vserver             4.0
  Virtual Target     4.0
  Logio Library       4.0
  *KernelIntegrator  3.1C
  *KernelBuilder     1.0B
  Toolbuilder lib    4.0
  TargetManager      4.0
  SNX                 3.2
  IFX                 3.2
  C RTL               2.0
  *BSPBuilder        3.2
  Xtrace              4.0
  VRTXsa              4.0A
  Spectra Boot       4.0
  m68k-sgdb 4.0 (based on gdb 4.6.6)
```

* Shipped separately.

```
@dnl Download region (because of UNUSED_HOST)
board.memory.host.address: 0x61ff0
board.memory.host.size: 0xC1E010
board.memory.host.type: BOOTOS_MEMORY_UNUSED_HOST|BOOTOS_MEMORY_COPYBACK

@dnl memory for vrtx from 0xc80000 till 0x1000000 (because of UNUSED_TARGET)
board.memory.target.address: 0xc80000
board.memory.target.size: 0x380000
board.memory.target.type: BOOTOS_MEMORY_UNUSED_TARGET|BOOTOS_MEMORY_COPYBACK
board.memory.target.speed: 200
```

As one can see, the whole product is fragmented in many sub-products. The first ones are related to the cross-development environment. The second ones are related to the RTOS and target manager.

Environment (board, host, network...)

The products were evaluated on a Motorola VME board: the MVME162 Model 533 (68040 at 33 MHz with 16Mb Ram) with Ethernet and SCSI support. SNX, IFX, NFS, SCSI are part of this evaluation. We used a VME bus analyser of VMETRO to measure time from interrupt to ISR and to task run.

The development platform used was composed of SunSparc Stations with SunOS 4.1.3. The development tool suite used was that of Microtec (MasterWorks, MCC68K, ASM68K, etc.)

Framework

The products have been evaluated during a project of about 2.5 manyears. Four people have been involved in the project, so the evaluation is quite objective. We had no pre-set opinions before starting the project.

INSTALLATION AND CONFIGURATION

Ease of Installation

For this kind of product (and at such a price) you normally have someone from the company to install the product on your site. We found, however, that it would be a good test to try to install it ourselves. It is effectively a good starting point to evaluate the installation procedure: you learn and discover if the product has been developed correctly, the installation program included, or if the company rushed to market. This tells you more about the time the company spent to produce quality software than any other test.

Installation was more or less typical for the Unix world. You need an expert in Unix who has installed many products, and even though, it was not easy. The shell that installs the products is not very clear on where it installs the components and what you need to do to select the components you want to install. In the competition between Unix and Windows NT as development platform, it is quite strange that the Unix world has not learned yet from the Windows world how to make good installation products.

The user will not normally have to do the installation

themselves so these problems will not affect them. It should be noted that the last release of the software seems to be better on this point. The software comes on a CD with reasonable documentation concerning the installation. However, it is still a far cry from the MS-Windows based installation program simplicity.

Ease of Configuration (Xconfig)

Another important point to look at is the ease of the configuring the RTOS to match the board and application requirement. Every manufacturer today claims that its RTOS is scalable and configurable. This means that the RTOS needs to be configured during the compilation phase so one needs to modify Makefiles and/or some source files that configure the memory map.

Manufacturers usually let the developer edit these files. To simplify things they put many comments in the files need to be modified. Microtec has chosen another policy. They have developed a configuration program (Xconfig) and a meta-language that is used by Xconfig to build the needed Makefiles and configuration source files. The advantages of such an approach are the following:

- the user does not need to edit Makefiles and source files himself.
- the configurator can check if the user choices are relevant and consistent.

The only drawback, however, is that the user needs to learn a new language. This could be avoided by developing a graphic tool that will generate the meta-language.

Microtec solution does not simplify things. The advantages that the user could expect are not there, as Xconfig does not perform any verification on the user inputs. Moreover, there is no syntax checker so the configurator will produce a result even though some words are misspelled!

In the previous example, one can see that syntax is not simple and could very easily lead to syntax errors. These will not be checked. Therefore, one ends up with troubles.

The Xconfig configuration tool could have been very useful and powerful if some syntax and consistency checking was included. Maybe this will be corrected in a subsequent release. For the time being, the user

The products have been evaluated during a project of about 2,5 manyears.

program (Xconfig) and a meta-language that is used by Xconfig to build the needed Makefiles and configuration source files. The advantages of such an approach are the following:

should be very careful with the syntax and with the inconsistencies he could introduce. The only certainty he may have is that if the RTOS does not start, there is a problem in the configuration files used by Xconfig.

To use Xconfig effectively frequents calls to support are needed.

With the Xconfig method, also another problem arises: all configuration information is created during compilation time. This is also the case for TCP/IP information tables: host addresses, netmask, NFS mounted filesystems, user-id, group-id, ... With such a mechanism you can not change the configuration during run-time (or installation time). Therefore, our development team needed to change the generated code to create our own network configuration mechanism. As such, it was possible to change the configuration in run-time.

Like in most RTOS, configuration is foreseen in boot PROMs only, and thus fixed during compilation time. This makes the configuration less flexible.

RTOS RICHNESS

Kernel

One important point for the developer of a real-time system is the richness of the RTOS. It is important to have many mechanisms included in the RTOS to reduce the time spent on the application development. For example, once you have a pre-emptive multi-threaded scheduler with a mutex you can develop any other synchronisation mechanism. However, this is surely not what the designer of a real-time system wants to do. He wants these mechanisms included in the RTOS.

To evaluate the richness of a RTOS, we have listed the system calls you find in many RTOS and we have compared with the ones included in VRTX.

Here we do not display the full list, it will be available on the WEB. As an example, we take the table relative

| | Property | VRTX |
|--------|---------------------------|---------|
| Queues | | Yes |
| | Message by address | Yes |
| | Message by data | No |
| | Send with acknowledge | No |
| | Send with msg priority | No |
| | Send on top of queue | Yes |
| | Send with timeout | No |
| | Send broadcast | Yes |
| | Receive with timeout | Yes |
| | Receive non blocking | Yes |
| | Receive blocking | Yes |
| | Pending in FIFO order | Yes |
| | Pending in priority order | Yes |
| | Notify | No |
| | Inquiry | Yes |
| Total | | 66.67 % |

Table 1.

to Message Queues:

The Total is the percentage of system calls that are available out of all those that could have been included.

For the other categories we have:

- Mailboxes (Queue with only one slot): 40%
- Mutex: 90%
- Semaphore: 90%
- No Control Shared Variable: 0%
- Event Flags: 92%
- Memory Partition: 60%
- Memory heap: 50%
- Task (or thread) management: 80%
- Timing functions: 87%

So we get a total of about 70%.

As you can see, VRTX is quite rich. Almost all the most common calls are there. A few are missing, like creating a task without starting it, post by data a message to a queue, etc. Nevertheless, the most important calls are there.

Standard Device Drivers

More and more developers need many standard device drivers, as the present real-time world is not composed of simple I/O anymore. The developers demand network, disk, and fieldbus connectivity.

The connectivity of VRTX is reasonable. One has TCP/IP (NFS is also included), a serial driver, a SCSI driver, etc. Some hardware component manufacturers support VRTX, but this is not widely the case. For example, most of IndustryPack and PMC modules do not support VRTX, so it is likely you will have to write your own device driver.

Moreover, the customer of a RTOS wants to use it in many projects and in many different environments. Therefore, the supported board list should be long. Most boards support VRTX even if the supported board list is not as long as the one of VxWorks for example.

PERFORMANCE EVALUATION

Kernel

The purpose of this evaluation is not to go into deep details about measures. So we will not give any figures about interrupt latency, context switch, interrupt to task run, and so on. Of course, we have measured some of these but not extensively with different load situations (number of interrupts, number of object in the system, etc.). The conclusion is that the kernel is quite fast and predictable.

We have measured, on the MVME162 33MHz, a time of about 20 μ S from interrupt on the VME bus to the first line of ISR and of about 90 μ S from interrupt on the VME bus to task run (using a mailbox message notification from ISR to task).

So one could be confident that the system would always react in about 100 μ S to an external event (if there were only one at the same time, of course).

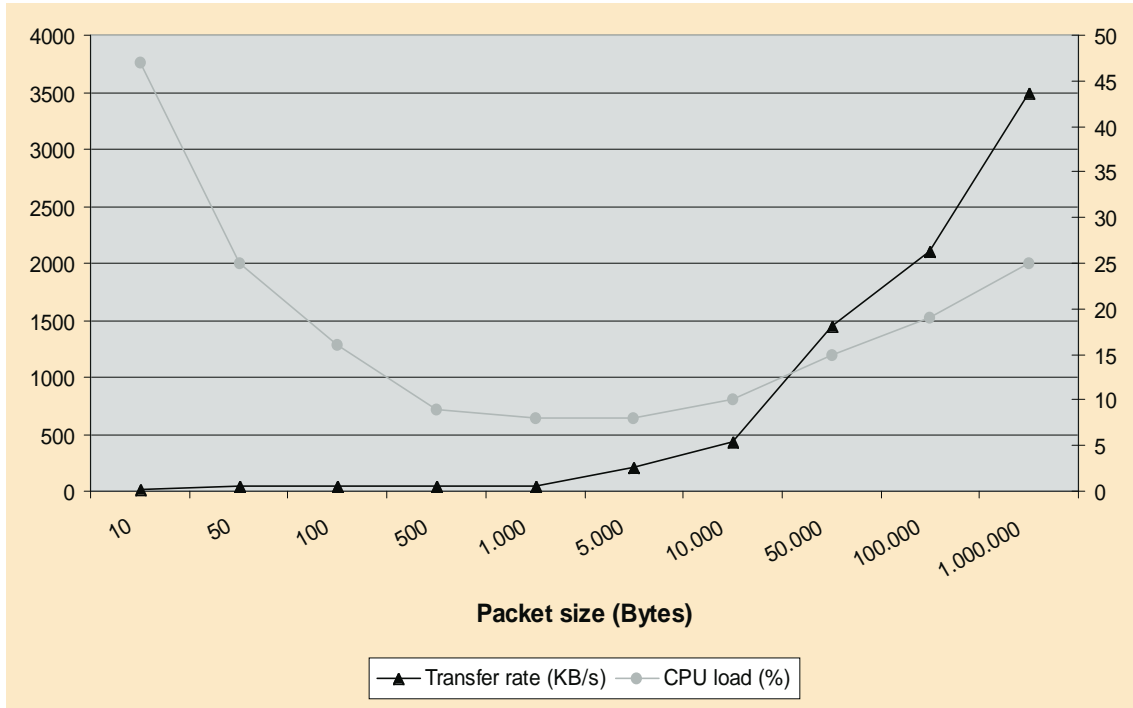


Figure 1. SCSI Transfer Rate

Standard Device Drivers

Today's real-time applications demand connectivity. This could be disk, network, serial I/O, etc. So it is important that such connectivity be included in standard with a good performance level. Is this achieved in VRTX?

The SCSI driver is acceptably fast (figure 1).

One only needs to tune the block size to get the best "Transfer Rate / CPU usage" ratio, but this is logical. Maybe Microtec should include a performance test program or such figures to help developers in their work.

However, the network driver is unacceptably slow.

We have tested the network (TCP/IP) as we needed it in our project. Microtec documentation states that 99% of the Ethernet bandwidth can be reached with TTCP. But this is a non-controlled stream test. So we decided to test the socket implementation as this is what we needed. It was discovered that it is hard to get a higher bandwidth than 350kB/s. And even then the CPU load is very high (almost 100%). This performance can only be reached using large packets (around 1.2 kB/packet). With smaller packets (300 Bytes) we got only 100kB/s throughput.

NFS is also very slow: about 50 kb/s throughput. But using NFS, the CPU load remains low (15% to 20%),

even on maximum NFS throughput, which is quite strange.

The serial console driver can disturb the application.

We have noticed many times when producing a large quantity of data on the console that the kernel is so disturbed that it missed deadlines of 10 ms (which was the hard deadline in the developed application). Care must be taken when using it to produce debug output for example. Note that we have not checked the serial driver itself but the console driver on top of the serial driver.

For this section, we conclude that the performance of the kernel is good but the connectivity is not as powerful as it could be. This means that VRTX could not be used in an environment where this connectivity will be heavily used. In that sense, it is more designed for embedded applications.

We are about to re-test this with the last release to see if performances have increased. We will publish the updated results in a next issue of Real-Time Magazine.

USER FRIENDLINESS

Documentation

A very strange point in the VRTX documentation is

```
int accept(int s, struct sockaddr *addr, int *addrlen, int *err);
```

Example 1.

```
int accept(int s, struct sockaddr *addr, int *addrlen);
```

Example 2.

that it is unclear which product release is being referred to. It seems to have been designed before the product. For example, there is no reference to the product version. In the track changes page it is always a first version that is mentioned (we have seen many different versions of the documentation). Sometimes the system call descriptions are not accurate. For example in SNX 3.2, the accept call has the parameters of example 1. But in the documentation, you find the parameters of example 2.

Hopefully, you get a compilation error and you can check in the prototype files...

It seems that Microtec has a lot of problems with version maintenance. For example, we needed a patch for the Board Support Package of the MVME162-533 serial interface driver. We have received the patch in October 1995. But in a newer release of the product in August 1996, the patch was still not in the BSP.

Another big problem with the documentation is that there are almost no examples. Every system call should be explained with at least one example. Moreover, Microtec has included an online help à la man (sman) but they removed the examples and the references to the other related commands. So it is in the end much less powerful than the standard Unix man.

These remarks also apply for the last release.

GUI

The GUI for their development tools (called MasterWorks) is more or less a graphical version of a text interface. This means that the user should know which command is behind a button. A good example is the makefile GUI. If you do not understand which flag is behind a checkbox, you cannot easily know if you need to check it or not. Moreover, it is absolutely not user friendly. For example, you have to scroll to see the full page. You cannot resize the window. (see figure 2)

All these windows are an old style of GUI. (figure 3)



Figure 2.



Figure 3.

We asked some external development teams whether they were using MasterWorks or not. Most of them replied they were not using it and the ones who did only made use of the text editor and the debugger.

Xconfig

As it has been said above, Xconfig is far from being user friendly. You should at least become an expert with the tool before being able to get a right configuration meeting your requirements. This is a pity as it could have been very powerful and could be an essential asset for Microtec's product.

DEVELOPMENT TOOLS

Developers today do not only demand a good kernel but also a good development environment as this reduces the development time and allows a shorter time to the market. Here Microtec has a complete set of applications, MasterWorks. It is composed of compilers, linker, editor, source code control system, makefile generator, debugger, profiler, target control shell. The tools are undeniably there. But is it worth using them?

Xray MasterWorks

The GUI is so bad that developers prefer not to work with it (see above). All the functionalities are present but the GUI spoils everything and reduces its usefulness. The editor is not bad but is far from what you have been able to find on PCs for a long time.

Xsh

Xsh is acceptable and does what it has to do: download, debug, etc. The only reproach is that background commands cannot be launched (which could be useful for UNIX commands). It could also be slightly faster when downloading programs to the target.

Debugger

As with almost all debuggers in the real-time multi-threaded world, it is not a bad source code debugger but a bad real-time debugger. It is not adapted to debug real-time problems so one will not be able to find missed deadlines. Moreover, at source code

debug level, it is not possible to have the mapping of structures, so one is not able to click on a pointer to display the structure it points to. One has to display the memory oneself and to map mentally the structure to it! This is not acceptable anymore today.

Remark that this has improved greatly in the last version, where you can click on a structure and its elements. But sometimes it reacts strangely, or it does not find which structure is behind it (lucky you can cast it to whatever you want). Also the GUI for it is still clumsy. As an example, if you opened a structure with substructures in it (like a tree) you can not close the root of the tree without closing all branches...

Profiler

It is a good profiler at the task level. But when it comes to functions you need to include the tracing instructions in the code yourself, as the compiler does not do it for you. So it introduces two overheads: one in writing the code and one at execution time.

Compiler

The compiler is good and it produces its best output when the code is simple and the functions short (like any compiler). We could have expected a better level of integration with other tools like the profiler. But as mentioned above, the compiler does not produce any output for it. Moreover, it does not allow mechanism to track stack overflow during runtime.

Linker

The linker is good and is an incremental linker. It links only what is needed. But care must be taken in the order of library linking. Sometime libraries need to be included twice. Here is an example out of a Makefile generated by Xconfig:

Do not try to remove one of the kernsupp.lib entries as it will not link anymore.

```

vrtxos_LIBS = $(LIBSDIR)/vsyslib.lib \
  $(LIBSDIR)/fppsp.lib \
  $(LIBSDIR)/nfs.lib \
  $(LIBSDIR)/rpc.lib \
  $(LIBSDIR)/appl.lib \
  $(LIBSDIR)/snx.lib \
  $(LIBSDIR)/resolv.lib \
  $(LIBSDIR)/sockets.lib \
  $(LIBSDIR)/swksyscall.lib \
  $(LIBSDIR)/ifx.lib \
  $(LIBSDIR)/vrtxsa.lib \
  $(LIBSDIR)/kernsupp.lib \
  $(LIBSDIR)/kernel.lib \
  $(LIBSDIR)/kernsupp.lib \
  $(LIBSDIR)/vsyslib.lib \
  $(LIBSDIR)/cpu.lib \
  $(LIBSDIR)/rtifxnofp.lib \
  $(LIBSDIR)/tli.lib

```

CONCLUSION

We divide the conclusion in two parts; the first one will present the application profile VRTX is good for, the second part deals with our opinion about the products.

Application Profile

It appears that VRTX is suitable for small-embedded real-time applications up to connected real-time applications. The memory footprint could be reduced to a minimum so small embedded applications can be based on VRTX. Real-time applications can be developed upon this good kernel. Finally connectivity to the world is present so connected real-time applications could be developed on top of VRTX. However, the connection to the network should not demand a high bandwidth, as SNX is not really powerful. It is good for configuration purposes and low bandwidth usage. On the other hand, the SCSI implementation permits real-time use of a disk. One should keep in mind that VRTX is more suitable for simple real-time application as multiprocessing is not supported.

Good points and Enhancements needed

The good points are the following:

- The kernel is rich, good and predictable,
- There is a good SCSI driver (at least for the MVME162),
- The idea of having a configuration program to configure the RTOS for your needs is a good thing (one does not have to look in makefiles and source files). The only drawback here is that the Xconfig tool is not complete and should be enhanced.

Apart from Xconfig, the following tools should be enhanced:

- The GUI of MasterWorks is not User Friendly. It is still a pity to have nowadays such "Graphical Text" interface.
- The documentation could be much better and useful if more examples have been incorporated. Nevertheless, we see a trend to enhance it in the last release.
- The documentation version should match the product version or a link should exist between them.
- The network connectivity should be more powerful. If you need a good TCP/IP driver at the moment you have to purchase one from a third party.

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.

Take two minutes time to influence the order of RTOS evaluation.



Reply possible in two ways:

1. take a copy of this sheet and FAX TO 32-2-520.83.09
2. surf to our web site at <http://www.realtime-info.be> and fill in the online form

Your E-mail address:

A. Do you use already a commercial RTOS?

- | | | | |
|------------------------------|--|------------------------------|-----------------------------|
| <input type="checkbox"/> YES | If yes for A, do you intend to change your RTOS? | <input type="checkbox"/> YES | <input type="checkbox"/> NO |
| <input type="checkbox"/> NO | If no for A, do you intend to start to use a RTOS? | <input type="checkbox"/> YES | <input type="checkbox"/> NO |

B. Give a list of the RTOS you want us to evaluate shortly in priority of order (give the first three):

1.
2.
3.

C. Cross the types of reports you are interested in (cross multiple boxes if necessary):

- Executive summaries in Real-Time Magazine
- Complete evaluation report of one product at 990 USD
- Complete evaluation report of a series of 4 RTOS products at 2990 USD.
- Feasibility study of the use of one or more RTOS products in your future applications (price offer on request).

D. Additional comments?

.....

