

Windows NT Real-Time Extensions as used in Industrial Automation

During the last two years the Industrial Automation market place has seen the proliferation of Windows NT4.0 working with a number of associated software 'extension' products. Companies like VenturCom, Radisys and Nematron (Imagination Systems) have appeared on the scene. In so doing they have provided the foundation elements for the building of so-called Soft-Logic solutions and 'all-in-one' visualisation and control packages - take as example Intellutions Paradym and WonderWare Factory Suite.

The impact on the industry has been phenomenal, major hardware vendors such as Allen-Bradley, Modicon and Siemens have been hard hit by the swing to the PC as the platform of choice.

- Is this the direction the industry will ultimately take ?
- Why should this interest you ?
- What are these so called Windows NT-Extensions
- and what do they offer that make them so popular

This report sets out to try and fill-in some of these details. It attempts to provide a clearer overview and a better understanding of the current trends, whilst at the

same time providing adequate detail to instruct and inform the Engineering community of the advantages/disadvantages of implementations of this new paradigm.

The target reader is any Engineering group member considering the use of WindowsNT extension products.

WINDOWS NT EXTENSIONS - WHY DO THEY EXIST ?

The goal in creating add-on functionality via extensions to NT is TWO fold :-

A COMPARATIVE DISCUSSION OF REAL-TIME APPLICATIONS... ON WINDOWS NT SERVER OR AN RTOS

(For the Automation & Control Industries)

A Designer's perspective

The designers of Windows NT had as one of their primary goals, that their creation meets the requirements of a NOS (Network Operating System). At the same time and in order to broaden its appeal and provide a framework for supporting the next generation of services, it was decided to implement a number of traditionally RTOS (Real-Time Operating System) specific mechanisms.

The first issue we address and that needs to be born in mind throughout this discussion is:

How focusing on the **primary objective** will impact the **secondary goals**.

Or put another way, if compromises in efficiency must be made, what factors will share in determining where the line is drawn.

NOS vs. RTOS - complementary and contradictory

It has to be acknowledged that many commercially available Real-Time Operating Systems provide very effective Network Services and Scheduling, e.g. QNX' FLEET, Enea Data's OSE, Wind Rivers' VxWorks, amongst others. At the same time several Network Operating Systems possess some powerful features

and functionality one would expect from an RTOS.

When determining the value of a particular system in relation to your own needs, it often helps to understand what was the designers' primary focus at the time of the products' creation?

- On providing a seamless, interactive and standard interface ?
- or in responding to events in an efficient and time centric manner ?

The NOS

Novell, the founders of modern NOS's, define their creation in the following manner:

'The network operating system software acts as the command center, enabling all of the network hardware and all other network software to function together as one cohesive, organized system. In other words, the network operating system is the very heart of the network'

Quite clearly their **focus is on ease of use, transparency.**

The RTOS

This is in contrast with the focus of the creator of an RTOS, where response times and predictability are

CONTINUED ON PAGE 42

Ad US Software

WINDOWS NT

1. To enable the software subsystem to provide improved real-time(RT) response (technical)
2. To assure as far as is possible, compatibility with 'standard' NT and thus benefit from the availability of the many win32 based packages and engineering resources. (commercial)

Let's examine both of these goals, then scrutinize the current commercial solutions.

REAL-TIME RESPONSE - WHAT DO THEY ACCOMPLISH ?

In a recent GMPTG report , it was shown that in the very best case a 20 fold improvement in response time could be achieved by using NT extensions instead of native NT . These figures are most impressive and appear to support the decision by some to consider WindowsNT as a base platform on which all future requirements can be met - Real-Time or other !

However in taking such important decisions and in order not to be carried away with the euphoric rhetoric of the 'she's unsinkable' crowd, all the facts should be considered in the 'cold light of day'. Worst-case scenarios have a habit of showing-up when they are least desired !

A closer look

To understand the possible problems, it is necessary

to understand how the extension systems under NT are implemented. There are basically two software methods and a third involving additional hardware :

1. WindowsNT runs as a task/thread inside the context of a real-time executive. Typical implementations are : Radisys InTime, Imagination Systems HyperKernel, RT-Win
2. Modifications to the HAL (Hardware Adaption Layer) re-direct hardware interrupts to a scheduler .dll that lives apart from the NT Kernel. This method is used by VenturCom in RTX.
3. A hardware device sits on the local bus and traps any CPU bound interrupts, it then generates an NMI that interrupts the processor, no matter what his mask state is ! A small specific ISR may then decide to allow the initial interrupt through to the RT kernel.

Both solutions (1) and (2) work very well in most cases BUT there are some inherent problems that can and do limit the ability of the extension to work efficiently....but first, let's enjoy a moments' distraction !

Every cloud has a silver lining - but it also means bad weather is on it's way !

In order to cross a long and potentially treacherous

CONTINUED FROM PAGE 40

foremost - i.e. the maximum time to do something should be known in advance and guaranteed.

The focus is on **responsiveness**.

A technical perspective

Most often the use of an RTOS implies multitasking, i.e. that the system can run several threads of execution that appear as parallel activities. The kernel, the heart of the OS, must also be able to preempt any of these threads when higher priority activities become ready for execution. This invariably implies that each thread be assigned a level of importance or priority!

All of these potential dynamic activities will need at some point to share resources. Maybe a data structure or access to a serial device. In order for this sharing of resources to be properly managed, there need to be some tools or mechanisms made available to the kernel to handle the worst case conflict scenarios. (E.g. Access to a shared resource - a potential source of conflict and thus missed deadlines !)

This might be illustrated as seen below:

Father : "I want the keys to MY car and I want them NOW! "

Mother : "I would like the car too, but you know John took it this morning and I haven't seen him since then... "

(time passes...<enter John>)

John : "Hi folks, ...what's the noise about, nobody was using the car when I took it this morning! "

Rules might be defined as to how access to the keys is managed! So it is in a RT kernel, the basic design focus is on handling such conflicts in a deterministic manner.

A second problem area might be illustrated as below:

John takes the car to collect his mother from the shops, however she has forgotten John is outside waiting and she meets an old friend... Both Father, John and the car are idling due to a low-priority but indefinitely lasting activity!

The RT application developer needs to have tools that allow him to manage suchlike occasions - if he is to be sure not to miss his deadlines. In this example, use of a priority inheritance mechanism would permit John to 'grow in importance' and 'stop' his mother's conversation...until some future time when father/John are watching the World Cup and she takes-off with the car!

Let's come back to look at Windows NT! - The concerns

1. Windows NT is based on a multi-tasking pre-emp-

CONTINUED ON PAGE 43

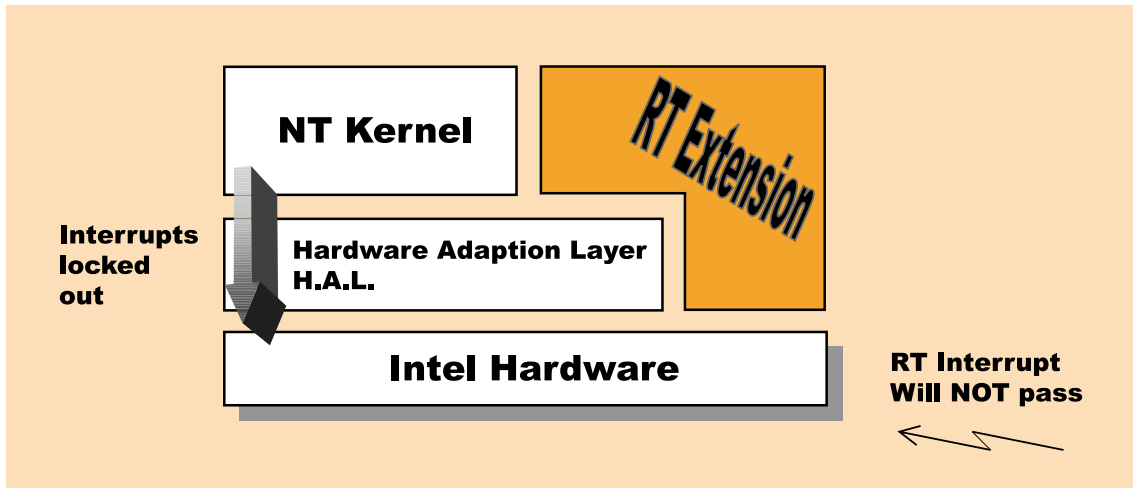


Figure 1. Architecture Diagram

mountain track, you might decide to :

- Add a winch to your audi quattro,
- Raise the wheel base,
- Add cross-country tires and a
- Add a bigger petrol tank.

You may have no problem passing the track. However, if the rains come and the track becomes a river, you

may well wish you had started the journey with a Land Rover and left the Audi for your daughter ! This is more valid than ever if you know the route is often flooded.

So it is with systems requiring worst-case real-time response. By simply adding features to a normally well adapted vehicle (NT), you might make the journey more comfortable and apparently less risky... but is it sure that you have carefully evaluated your real needs and prepared for the worst case ? If so, you'll have

CONTINUED FROM PAGE 42

tive kernel, however the number of different priorities that can be assigned to each activity is very limited usually 5 (at best 7). In many RT Applications this is far too few! Most RTOS vendors today offer 256 levels. Remember, if you have several activities of the same priority, the kernel may be UNABLE to assure meeting the delays.

2. Rules governing resource sharing. Windows NT does not possess a priority inheritance or ceiling algorithm in order to avoid the classical lockout problem. (As illustrated in Eg.2 above)
3. The win32 interface is implemented in such a way that some of the non-real-time threads can suspend in system calls (SIO) and thus force the higher priority real time threads to have to wait for access to the resource.
4. Device drivers pass information to what are called DPC's, unfortunately these DPC's all run at the same level and can be interrupted by ISR's. This means that a RT application may have to wait on information that is being held up by other non-rt drivers DPC 's that have been badly written...there are many in this category!
5. Memory footprint! Most developers of embedded devices have strict limitations on the amount of available memory. RAM limitations (cost based) being the most critical. Windows NT minimum requirement is 8M, whilst for comfort more is often

desirable. Many embedded devices must be able to run both the kernel and the application in less than 50K!

Clearly these factors impose severe limitations on the type of embedded application that could be built using Windows NT as the base OS. For anything other than simple control tasks not demanding deterministic response Windows NT cannot be relied upon.

An examination of the RTOS - The concerns

A number of COTS (Commercial Off-The-Shelf) RTOS products exist that respond perfectly well to the concerns raised above for Windows NT. On the other hand their programming interface is much less well known. There are two very real advantages in providing an NT only solution:

1. The ability to bridge between the Office environment and that of the Shopfloor in a transparent fashion.

(Incidentally: the RTOS vendors are fighting back by providing object level interoperability that bridges to the world of COM/DCOM from that of CORBA. This provides the same services, but with true RT response, as can be obtained on a single platform environment).

2. The use of the large product resource base (applications, drivers, ...) coming from the office environ-

CONTINUED ON PAGE 44

peace-of-mind because nothing will take you by surprise. If not you might be calling for assistance, or worse...

After all, good engineering practice attempts to eliminate the taking of risks - as far as is practically possible !

Real-Time response - the true story

One of the primary goals of a designer of a RT system is to minimize latency (pre-emptive & interrupt latency - PL & IL) . It has to be said that in order to achieve minimum IL, both application AND operating system have a part to play. Unfortunately ALL Microsoft kernels (on the market at the time of writing) are unable to provide deterministic response. In addition, the kernels themselves cannot be modified by anyone other than Microsoft themselves - this introduces a serious problem to any RT system designer.

Interrupt latency has its greatest impact when the kernel locks interrupts for a given period whilst caring for some critical activity. In the Intel environment this masking is achieved via the interrupt enable flag. A number of tests have shown that WindowsNT4.0 disables this flag for respectively long periods of time - upto 2ms on a SCSI transfer for example ! There are many elements of the WindowsNT subsystem (kernel & HAL) that disable the IE flag.

What does this mean ? That no matter how important

the Real-Time interrupt may be, the CPU/kernel partnership will NOT allow the interrupt to pass through to be handled, until this flag has been re-enabled by the kernel routine currently locking it out. (See architecture diagram below)

As regards pre-emptive latency PL, this is largely due to the disabling of the scheduler (taskLock), little or no known information has been made available by Microsoft to indicate that there are OR are not extensive periods of context switch lockout. One very interesting aspect is that in WindowsCE there is currently no taskLock or means to disable the scheduler. This would seem to indicate that they did not wish to adopt the model used under NT - perhaps because of its inefficiencies !

This assumption would seem a reasonable one, particularly in light of recent announcements from Microsoft, one such highlighted by EE Times (1/4/98) :
"... revamp its Windows CE operating system to include "hard" real-time features which can support time-critical applications...It's not clear when Microsoft's beefed up version of CE will be available, but code is not expected before the end of the year. "

Resume of Software NT-Extensions and their RT responsiveness

It is clear that implementing one of the so-called real-time extensions may to a very large extent improve the

CONTINUED FROM PAGE 43

ment market leadership.

(In response to this, a large number of major hardware and software vendors have teamed-up to define and implement a number of standard based products i.e. Drivers - I2O, interoperable applications - JAVA, OPC or CORBA)

Customer needs in Industrial Automation are many and varied, perhaps having even greater diversity than the services offered by any one vendor today. Requirements may include:

Services

1. Scaleable GUI builder, integrated with the scaleable libraries used to provide on-board graphics
2. Standards respecting API for portability.
3. Multi-window IO requirements or simple form creation.
4. Cross-architecture support: same (recompiled) program will run on several hardware platforms. E.g. Intel Pentium, PowerPC, ARM, MIPS, SHx ...
5. Graphical Client-Server type remote system management (Supervisory role - SCADA).
6. Database and object inter-operability at the 'click-of-a-button', seamless OPC support.

Most vendors today offer extensive graphics tools. Much noise is being generated around the use of OPC(OLE for Process Control) as THE interface

between tools. However it needs to be said that, as with most standard API 's the implementation methods and obligation to use all services proposed is NOT covered in the specification. The informed user will be aware that support for services based on emerging standards is extremely varied from one vendor to another and is very rarely efficient. This raises questions concerning true interoperability and support down through the years!

One other issue not as yet discussed, is that of tool support.

Tool support

Working with deeply embedded devices has traditionally been the domain of the technically astute individuals often having a strong hardware appreciation. Development tools have largely been inappropriate (except in large scale projects) due to:

Cost	The price of an In-circuit emulator is often restrictive
Time	Many debuggers had to be ported to the particular platform, before any real work be done.
Availability	Manufacturers preferred to concentrate on providing tools for 'richer' markets.
Resource	Often memory limitations proved

CONTINUED ON PAGE 46

Ad Enea OSE

COMPANY NAME	PRODUCT NAME	PRINCIPLE OF OPERATION / NOTES
VenturCom	RTX	Modified HAL with .dll scheduler
Radisys	InTime	WindowsNT runs as InTime task
Nematron	HyperKernel	Modified HAL with .dll scheduler
LP Elektronik	LP-Win	Hardware addition (PAL) to assure Hard RT.

Table 1. Implements of NT Extensions

responsiveness of native WindowsNT in given circumstances. However by design NT has a number of inherent weaknesses with regard to real-time response, most notable is the interrupt locking problem highlighted above. Thus no real-time extension software provider can rightly claim to be able to provide hard RT response. A true definition would be that NT software extensions are soft real-time. This means that :

“most of the time I can meet my deadlines - but I must be able to miss them occasionally”.

So simply adding features to a system NOT designed for the terrain in question, may be perfectly appropriate in some circumstances. On the other hand it may leave you disappointed and/or in potential difficulty.

The author of this report is not decrying the use of NT-Extensions but rather attempting to clearly outline their limitations and inadequacies in certain situations !

Hard Real-Time Extensions - an implementation

One way of working around the problem described above is by establishing an external 'watchdog' that will intervene when RT interrupts arrive AND are currently locked out. This is the method implemented by LP Elektronik. In effect, assuming that Windows-NT has caused the masking of interrupts and an RT interrupt arrives, the hardware device (PAL) detects the request on the local bus and causes an NMI (non-maskable interrupt) to be sent to the CPU. This in turn runs a specific interrupt service routine (ISR) that will handle the event and intervene to cause the RT kernel to respond to the interrupt. Thus a deterministic response time (in the order of several micro-seconds) can be assured. However this method is not without its drawbacks as an additional hardware device is required. Yet it remains the only way of assuring true RT response on a mono-processor environment running Windows NT.

CONTINUED FROM PAGE 44

to be inhibitive.
(Continued overleaf)

Because Windows NT has never traditionally played in the embedded market, today's development tools for this system are very limited in both quality and feature set. In traditional cross-development platforms this is one area where there has been the huge improvement in product technology. Today developers have at their finger-tips, rich and powerful readily available graphical tools that provide:

Modeling (high-level definition) & code generation tools

E.g. MathWorks, ControlShell, ObjectTime, Telelogic and Verilog amongst others

Symbolic debugging environments

E.g. Industry standard tools such as Crosswind, SingleStep and XRay+

On-chip debug environments

From EST, HP and Abatron

Dynamic on-line object browsing

E.g. Wind Rivers' - Look!

Static object code browsing

E.g. Wind River's Navigator and Take Five's SNIFF+.

System & Application Dynamics Visualization

E.g. WindView from WRS, TotalView from Lynx and esp from ISI.

Integrated Code-Coverage Tools

E.g. AMC CodeTest, Attol TestWare

Perhaps the most important step forward of recent times was in introducing integrated development environments to the Industry (IDE's). In such, both vendor supplied and third-party tools seamlessly interoperate with each other and the target platform via a clearly defined publicly available interface.

E.g. Wind River Systems Tomado,

There is much activity currently around the development of object distribution interfaces (as defined by CORBA and COM/DCOM). Will these provide the next generation of IDE ? Current implementations tend to favor a superset of the two interfaces with proprietary services for low-level work.

Conclusion

As discussed at the outset, the author believes that effective market solutions will be provided by the vendors smart enough to broaden their products' appeal by focused application and not by a stubborn force-fit approach. I.e. by creating scaleable and modular products that share a core of common technology rather than adding-on attractive features to what is essentially a poor fit for the embedded arena.

A number of common implementations of NT Extensions is shown in table 1.

COMPATABILITY AND WIN32

Application developers have long dreamt of a standard programming interface that would allow them to write once - run anywhere. Unfortunately for a number of technical and political reasons, such an interface has never really appeared. There have been a number of well intentioned efforts, more recently :

- POSIX 1003 (born of UNIX and spawned into many flavors - 1003.1b/c.),
 - ESSE (targeted to general embedded) and
 - OSEK (Automotive).
- (see 'authors' opinion' in appendix)

In parallel to this, major players in the marketplace have sought to provide 'solutions' to the problem that may become 'de-facto' standards due to their wide-spread use and commercial presence :

- SUN Microsystems with their JAVA approach and
- Microsoft with their slightly less ambitious but nonetheless ubiquitous WIN32 API.

So back to the subject matter - WIN32. Today this interface purports around 1000 calls - including areas such as GUI control, kernel access (multi-threads), message and loader interfaces. This is a very broad range of service to offer and far too many for the developer of embedded products. Again, this is made clear by the fact that WindowsCE latest offering implements only a reduced subset of WIN32 ie. around 500 calls - which means there is no longer ONE standard win32, but several! Basic excel, word, powerpoint applications will NOT run on this new platform. Remember, what was the goal :

'write-once, run everywhere (that it runs)'

"... all animals are equal, but some are more equal than others..." George Orwell - Animal farm

Concerning compatability, an application that has RT requirements, HAS to be written using the proprietary API for the NT extension chosen. A 'standard' NT application will not become Real-Time by simply adding an RT-Extension product ! In addition, if there is to be Real-Time control of Input/Output devices, the 'standard' NT drivers just will NOT do. The driver has to be written for the proprietary driver interface for the RT-Extension as well ! So where is the portability ?

CONCLUSION

Certainly, both from a technical and a business perspective, NT-Extension products will continue to be used in many Automation and Control applications. However, hopefully this paper has raised a number of important considerations, most notably:-

- Lack of true deterministic response from Soft RT

Extensions

- Lack of portability.

...that provide the reader a more complete picture to reflect on when making a choice of a suitable platform for his application.

Currently many Systems Integrators and Equipment Manufacturers are building their products using commercially available RTOS's. An explanation as to the reasoning why can be found in the associated paper entitled "Real-Time Applications... on WindowsNT Server or on an RTOS".

APPENDIX

Authors opinion...more vertical market specific proposals (as in OSEK) are far more likely to have success than the more 'general' horizontal approach of POSIX - largely due to the \$ factor. The one thing able to unite competitive vendors in such force that they dictate to their suppliers what they will buy, is their fear (real or imaginary) of losing market share. Some of the critical elements in obtaining and keeping market share are :

- Technological innovation - real product advantages
- Effective marketing - perception of product advantages
- Financial execution - ability to deliver more of these product advantages for less cost.

Concerning the latter element, in particular cost control (thus the price of the end product) - cost control is believed to be associated with flexibility in design, implementation and working practise. This is the force that has prompted the automotive market to define their needs for their OS of the future. No such clarity or shared inter-dependancy exists across the horizontal plane - thus the author believes, no industry defined standard will succeed in meeting the needs of the market as a whole...end of opinion. ■

Stephen Porter is currently working as Market Development Manager for Wind River Systems in the Industrial Measurement and Control market place. He began his career at Imperial Chemicals Industries (ICI), and for four years worked in the instrumentation and control department, charged with engineering maintenance through project design. Following a one year interval as College Lecturer, Stephen returned to engineering, designing software for DSP based products in the Food Industry.

In August 1989, he moved to France (where he is still based today), with the remit to develop telecommunication software destined for use by France Telecom. Stephen joined WRS in 1992 and of recent years has been employed as Field Application Engineer with the Sales team caring for Southern Europe, South Africa and the Middle east. Stephen has an HND in Electronics