http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

RTOS STATE OF THE ART

UNDERSTANDING RTOS TECHNOLOGY AND MARKETS?

BY

PROF. DR. MARTIN TIMMERMAN & LUC PERNEEL

Disclaimer

Although all care has been taken to obtain correct information and accurate test results, Dedicated Systems Experts cannot be liable for any incidental or consequential damages (including damages for loss of business, profits or the like) arising out of the use of the information provided in this report, even if Dedicated Systems Experts has been advised of the possibility of such damages.

http://www.dedicated-systems.com

m.timmerman@dedicated-systems.com

l.perneel@dedicated-systems.com

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

1 Preface

There are lots of conceptions, even more misconceptions and certain confusions about the meaning of real-time and real-time systems compared to embedded systems. As a consequence there are also a lot of controversial statements about the operating systems used as an important component of these RT-systems most commonly named as RTOS (Real Time Operating System).

This short document wants to introduce only the basic notions about real-time and especially RTOS and give a limited discussion about some products and markets.

We will first define what real-time systems are in laymen's terms. In a second part we will use some commercial products such as VxWorks, Montavista Linux and Windows CE, to introduce in a pragmatic way some RTOS characteristics. The third part will describe specific features that might be needed in some vertical markets such as Industrial Automation and manufacturing, Medical Devices, Consumer Electronics, Retail and Automotive. In this part products such as Windows XP embedded and VxWorks 6.x we only evaluated theoretically will also be included.

The products discussed in this document are competing in these markets. There are of course other markets and products. For a more in depth study of the technology, the discussion of other products and markets, we refer to complementary documents available from Dedicated Systems.

After reading this document, you should understand

- what an RTOS is
- how important the RTOS choice is in the framework of the specific system needs

Enjoy,

Martin & Luc

This document is dedicated to our wife's for accepting the endless evenings and week-ends without us busy following up the continued evolution of this technology.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

2 Real-time systems & RTOS

2.1 What is a real-time system?

The first important question is: ***What is a real-time system***? Different definitions of real-time systems exist. These definitions are mostly given in the framework of a specific application field. We therefore tried to make one which is independent of an application domain:

A real-time system responds in a (timely) predictable way to all individual unpredictable external stimuli arrivals.

The most important word is **PREDICTABILITY.** The system should respond to each individual external event in a predictable way, this means before the deadline defined in the system requirement. It is important to note that ***average performance is NOT the issue!***

2.2 Real-time system components

To build a predictable system, ALL of its components, hardware & software, plus a good design are contributing to this predictability. Having both good hardware and a good RTOS is a minimal but not sufficient requirement for building a correct real-time system. A wrongly designed system with excellent hardware and software building blocks may still lead to disaster.

This document deals with the RTOS building block. In general: ***a good RTOS can be defined as one that has a bounded (predictable) behavior under all system load scenarios (simultaneous interrupts and thread execution).***

2.3 Real-time system types

An embedded system does not necessarily need to have a predictable behavior, and in that case it is not a real-time system. However, a quick overview of all possible embedded systems shows that you will rapidly find the need for some predictable behavior and therefore most embedded systems need to be real-time for at least some of their functionality.

In a well-designed RT system, each individual deadline should be met. With the actual state of the art, it is sometimes hard and also costly to achieve this requirement. Therefore people invented different ***types*** of real-time systems.

- **Hard real-time**: missing an individual deadline results in catastrophic failure of the system (and people will hopefully invest sufficient money in this project in order to avoid this catastrophic failure). It also means that the cost of the failure is very high.
- **Firm real-time**: missing a deadline entails an unacceptable quality reduction, technically there is no difference with Hard RT but economically, the “disaster risk” and associated cost is limited compared to the previous case.
- **Soft real-time**: deadlines may be missed and can be recovered from. The reduction in system quality and performance is acceptable and does not introduce another than technical or feature cost.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

– **Non real-time**: no deadlines have to be met. These (general purpose) systems are defined in terms of average performance.

Economical real-time: We recently introduced this notion because it is in practice impossible to design a system that will never miss any deadline. The real question is therefore: "*What is the price or economic damage one (or the insurance) wants to pay for a missed deadline introducing a safety hazard or a quality of service reduction?*

2.4 Modern characteristics for embedded systems

Modern embedded systems have one or more of the following characteristics

Real-Time: have a predictable behavior for some or all of the features of the system

Safe: does not harm the user or in other terms avoid physical or economic damage to persons or property

Secure: only intended use of the system will be permitted. This also means avoiding un-permitted access or modification

Fault tolerant: avoid the systems stops working or fails. It is the ability of a system or component to continue normal operation despite the presence of hardware or software faults. **Robustness** falls also in this category but is a lesser requirement. It means the ability of a system to maintain performance or degrade gracefully when exposed to conditions not well represented in the data used to develop the system.

This is the main reason why we started using the word DEDICATED systems instead of REAL-TIME or EMBEDDED systems in the framework of the work our company is performing.

If we evaluate Linux or Windows XP embedded, you know from the very beginning that these OS have no real-time characteristics. However, designers claim to have sufficient reasons to use them in embedded systems and therefore we also wanted to evaluate these non real-time products.

Dedicated system: *The functionality of the system is predefined by the available hardware & loaded software. Hardware and software reprogram ability may be used to a certain extent to change the functionality through the lifetime of the system*

Since the explosive use of FPGA's and similar hardware giving the possibility to the designer to change not only the software, but also the hardware functionality during the system's lifetime, we needed to "complement" the basic definition of a dedicated system with the last sentence in that definition.

It should be clear that these system modifications can at our knowledge only be done when the normal function of the system is stopped or at least partially stopped. This is a very important remark as one expects an embedded system to work in most cases indefinitely or continuously.

The use of this changing hardware functionality during the lifetime of the system introduces a new challenge for RTOS components: how to adapt preferably automatically to these changes?

2.5 Multitasking or multithreading

Even the simplest embedded system will probably deal with more than one event. One can deal with these multiple events in various ways, but a multitasking or multithreading approach on one processor is a very

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

common solution. RT engineers are used to multithreading an application since the very beginning of real-time embedded systems. Only more recently, general purpose software is evolving in the same direction.

Using an RTOS as basic component with many tasks or threads working closely together to deal with the application, is not just another flavor of software writing. It is for time being completely different from business software writing. A real-time software engineer is constantly busy designing the collaborative mechanism between threads and tasks and will daily deal with device drivers and interrupt handling what most software engineers hate doing.

2.6 Deadline spectrum

The deadlines to respect in RT systems might range from Pico seconds to seconds and even hours. Today's processors might be fast and deal with interrupts in the microseconds range, but power consumption may then be a serious problem. Portable embedded systems will have limited processor power for that reason.

Therefore there is a spectrum of technologies to be used to deal with the different deadline requirements.

The shortest deadlines (less than a microsecond) will have to be handled by hardware without software being involved. Less short deadlines (1 to 10 microseconds) can be dealt with one processor and just one program and some interrupt routines. Using an RTOS will help a lot in designing complex embedded system but only deadlines in the range of .01 to 100 millisecond's and more can be reasonably handled. For very long deadlines, one might even think using humans if reaction times are specified in the hours range.

The fire brigade is considered a real-time system, because one expects them to be there within some 10 minutes or so after a call. This simple example shows also that sufficient resources need to be present in idle condition to realize this constraint. Indeed, if all fire brigade cars left for multiple fires, you will have no response if you call them.

2.7 Hardware freedom

System designers are always looking to have as many degrees of freedom possible. In practice this means that they would like to make software that will be portable to "any hardware". A solution here is to use an (RT) OS. If the RTOS is supporting multiple kinds of flavors of processors, you have definitely a solution.

However most designers dream of going one step further: being then independent of the RTOS. This is wishful thinking. Our research shows that the behavior of the available RTOS is so different that one cannot think about a migration from one RTOS to another without redesigning and rewriting the whole application.

IF one day, automatic code generation will be fully operational and used
AND OS standardization has taken one step further than it is today (including object behavior)
AND designers taking time to analyze and architectural design the application using these tools
THEN one might expect some advances in porting applications
HOWEVER, the problem of how to cut an application into a multitasking approach in an automatic way is still a serious research topic.

You see that there are challenges left …

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

2.8 This chapter's conclusion

2.8.1 Why is RT important?

Some (and certainly not all) of the functionality of your system will require a predictable response on one or more events. That specific part of your system will need a real-time or predictable behavior. You should avoid having too much hard or firm RT requirements in order to limit system budget. This is probably the main reason why so many systems have just (unjustified) soft RT characteristics due to budget limitations. In that way, systems might lack quality of service. One might expect that in the coming years, this quality of service will become a major requirement, pushing designers to use more firm RT solutions and requesting for a real RTOS instead of just one or another non real-time OS.

2.8.2 Which aspects are important when evaluating a real-time solution?

Making the difference between the categories hard, firma and soft RT functions in a system is an important issue.

A lot of people are confusing fast response with predictability and are basing their hardware and software choices on performance benchmarks. Of course, task switching and interrupt latencies for an RTOS are important issues, but they do not guarantee the right real-time behavior. It is mandatory to concentrate more on the behavior issue before starting designing.

We will not focus on hardware behavior in this document, but the reader should be aware that the knowledge of this hardware behavior is of similar importance.

Our research studies show how different RTOS' "behave". The difference in behavior comes from the difference in design choices for the OS itself.

This has several consequences

- Some OS are more suitable for some type of applications than others
- Requiring a design to be portable from one (RT) OS to another is for time being utopia. (This also means that you need to get it right from the very beginning, because a wrong RTOS choice will probably lead to the abandon of the project). One might invent a common API (like POSIX) but this is not a sufficient condition for a portable design. The behavior behind a system call should also be standardized, and this is certainly not something that will happen "tomorrow".
- It is impossible to do detailed application design without the knowledge of the OS to be used and especially without the knowledge of the OS behavior.

Notice that RTOS vendors do not publish these behavior issues, and even worse, they don't always know the behavior of their product themselves. This is one of the reasons why we launched our RTOS evaluation program in 1995.

2.8.3 The way ahead

Gradually and steadily, multithreading and real-time techniques will be used more and more in general purpose applications for different reasons and the differences between GPOS and RTOS will fade away. However we are then probably in the year 2050.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

3 Comparing some RTOS products

3.1 Introduction

Dedicated Systems Experts evaluates continuously real-time operating systems (RTOS). What RTOS are evaluated for publication depends largely on the goodwill of the vendor giving a special evaluation license including the possibility to publish the results and a budget for doing so. In this short document we will limit ourselves to 3 RTOS:

- The VxWorks RTOS from Wind River Systems
- Windows CE 5.0 from Microsoft Corporation,
- The Montavista Linux Professional edition 2.1 from Montavista

More recent versions of these products might be available today. Especially if we give negative comments, vendors are in a hurry to produce a new version of their RTOS. However we were not yet allowed from these vendors for unknown reasons to evaluate these new versions and we advice the reader to visit our website to verify new evaluations.

Anyhow, the purpose of this document is to make the reader aware of the important issues in choosing an RTOS, and it does not really matter if we don't discuss the "latest" version.

3.2 Summary

Windows CE 5.0, is the successor of CE .NET and CE 3.0. There have been 3 different versions of this RTOS in a very small timeframe of just 7 years. The compatibility between CE 3.0 and .NET was limited. CE 5.0 is totally compatible with .NET. CE 5.0 exhibited real-time behavior during our tests. None of the stress tests exposed any problems concerning stability and robustness either.

VxWorks 5.3 is a very old and stable RTOS. Wind River needed more than 5 years to finally release the 6.0 version end of 2004. More recently the 6.1 version was released. Wind River states to release a new version every 6 months which could only mean minor changes from one release to another. 6.x addresses fundamental lacking features in the older version such as memory protection and improved error handling.

The Montavista Linux Professional edition 2.1 has serious drawbacks if it comes to real-time behavior. It is certainly use-able to a certain extent in embedded systems requiring limited soft real-time behavior. However as we see that Montavista is trying to enhance the product constantly, adding more real-time features, they show themselves a clear need for these. We did not have the opportunity to test the very recent 4.0 release which seems to address some of the issues we discovered during our tests.

Although the Linux kernel is royalty free, it comes with a price: documentation is poor and the API is not always compatible with (POSIX) standards. The learning curve to get the kernel up and running on your custom target platform is steep. Montavista is offering paid support.

Finally, the reader should bear in mind that we tested all OS published in this document on an Intel x86-platform. Except for Linux the behavior on different processors is roughly the same. The vast Linux community has easy access to cheap x86 platforms. On more exotic embedded processors, we discovered serious behavior discrepancies which might be explained by the fact that only a very limited number of people have access to these (sometimes expensive development) platforms.

3.3 VxWorks 5.3.1. – Wind River Systems

"RT-VALIDATED", VxWorks passed all tests without problems.

3.3.1 Ratings (scale in appendix: chapter 6)

Installation and Configuration	0	6	10
RTOS Architecture	0	6	10
API Richness	0	8	10
Internet support	0	8	10
Tools	0	9	10
Documentation and Support	0	7	10
Test Results	0	7	10

3.3.2 Observations

☺ During all of our tests, VxWorks 5.3.1 exhibited fast and predictable behavior.

We did find that the time needed to add a thread to a semaphore's queue (and sort the queue) is proportional with the number of threads already in the queue. VxWorks should have used queue structures that result in a faster and more constant sorting times. However we understand that this requires more memory, which might be a problem for very small embedded systems with a limited memory capacity.

☺ VxWorks 5.3.1 has an extensive API and has a lot of additional libraries and device drivers for a wide variety of devices.

☺ Tornado 1.0.1 is a good and complete development environment with a very extensive set of tools. Furthermore, Tornado is an "open" environment, allowing for the integration of tools from third party vendors.

☺ Wind River Systems' technical support team is very professional. Furthermore, users with a valid maintenance contract have access to 24 hours/day online technical support, consisting of a large and well-organized database containing thousands of articles.

☺ VxWorks 5.3.1 has"client-server" architecture in the sense that it has a small microkernel which only handles the basic real-time features. All the other functionality is implemented as processes. VxWorks is not a message based operating system.

😐 All the tasks in VxWorks by default share one common address space, without any memory protection. Memory protection is available, but an optional component (VxVMI) needs to be purchased and installed. Version 6.x seems to deal with this issue.

☹ The manuals are not always clear. The documentation is not detailed enough on the system's architecture and the description of some API function calls.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

☹ The clock ISR execution time in VxWorks 5.3.1 can get pretty lengthy, especially when timers are used. We managed to create a situation where the clock ISR took more than 40 µs to execute. This slows the system down considerably, especially in platforms where the clock interrupt has the highest priority (like the x86 PC platform used for this evaluation).

☹ Does not qualify as a fault-tolerant, distributed system. Poor support for inter-processor communication. Again version 6.x wants to deal with this issue.

3.3.3 New product versions

Wind River released very recently a new 6.0 and 6.1 version to deal with some serious shortcomings of the older version. Dedicated Systems negotiates with Wind River to evaluate this new release, but these discussions do not seem to come to conclusions. The real reasons not to go for an evaluation yet are not clear to us.

3.4 Windows CE 5.0 – Microsoft

"RT-VALIDATED", CE 5.0 passed all tests without problems.

3.4.1 Ratings

Installation and Configuration	0	7	10
RTOS Architecture	0	8	10
API Richness	0	7	10
Internet support	0	9	10
Tools	0	8	10
Documentation and Support	0	7	10
Test Results	0	7	10

3.4.2 Observations

☺ Modular operating system, with a large amount of optional features. It can be extended with a lot of features such as a user interface, internet services and so on.

☺ All protection primitives use priority inheritance! It is the only RTOS that we tested that exhibit this behavior. In our opinion all other RTOS should take an example on this. This will limit already the potential design flaws caused by incorrect usage of protection primitives.

☺ Stable real-time results, worst case improved compared with CE 4.0.

☺ Interfaces easily with other Microsoft Operating systems. It supports the COM technology and the .NET (compact) framework with remoting: this makes it easy to interface via the network with any general purpose windows application. As an supplementary advantage, normal windows developers can be used to develop the non real-time part of the system.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

☹ Some limiting factors: like the number of processes (32) and amount of virtual memory (32MB for each process). But having processes already implies memory protection between them.

☹ The graphical BSP configurator, called “Platform builder” makes it easier to add the correct modules in your system. However, platform builder isn’t always reliable.

☹ Documentation could be improved further, although it is already better than in the previous version. As CE is an RTOS with a lot of extensions and a large API, it is a complex system with a lot of possibilities and good documentation is therefore important.

3.4.3 New product versions

CE 5.0 is the latest version of the Windows CE family.

3.5 Montavista Linux Professional Edition 2.1

“NOT RT-Validated”

Why did we test Montavista Linux Professional Edition 2.1 in our RTOS evaluation program, although the Linux community states clearly that Linux as such is not meant to be used in real-time environments? A lot of people are using it for different reasons and Montavista did apply some enhancements (patches) to the basic Linux kernel in order to “approach” a real-time behavior.

3.5.1 Ratings

Installation and Configuration	0	5	10
RTOS Architecture	0	4	10
API Richness	0	5	10
Internet support	0	8	10
Tools	0	8	10
Documentation and Support	0	4	10
Test Results	0	3	10

3.5.2 Observations

☺ The MontaVista Linux Professional Edition 2.1 (Pro 2.1) is easy to install and use. It is shipped with a correct "Linux Support Package (LSP)" for the board under test.

☺ With Pro 2.1 there are no runtime royalties. This can obviously be a considerable advantage when a large numbers of products are to be built. Furthermore, the source code of the OS is available at no charge. This guarantees the availability of the embedded OS for future developments.

☺ The real-time patches improve slightly the real-time behavior of the Linux kernel.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

☺ A wide range of open source tools and applications are available on the Internet that can be integrated in the embedded system that is to be developed. Especially in the field of computer networks, an important development effort – that is still ongoing – has resulted in a large number of supported application protocols as well as a lot of interesting network security related features.

😐 The Target Configuration Tool (TCT) assists in creating a target file system, yet unfortunately does not provide an initial population of for instance a minimal working file system.

☹ The Pro 2.1 kernel – on the x86 processor - is not a real-time kernel, neither in the default LSP configuration of the kernel, nor in a configuration of the kernel with all the real-time patches applied.

☹ A bug was found in the scheduler that makes it impossible to create more than 244 times a child thread from a given thread.

☹ Our technical evaluation showed serious behavior problems

☹ MontaVista Linux is not equipped with a priority inheritance mechanism.

☹ The GPL that applies to the Linux kernel and a large part of the GNU/Linux system may not be compatible with a company's wish to protect its development investments. Indeed, non-isolatable intellectual property may become locked in the GPL and the source code will need to be available on request. Furthermore, not all utilities and applications in a typical GNU/Linux distribution are licensed under the GPL. Some require separate licensing with possibly royalties for commercial use.

💣 Take care: MVL Professional Edition 2.1 cannot be used for firm real-time.

3.5.3 New product releases

Recently Montavista release the version 4.0 of the professional edition based on the Linux 2.6 kernel. The Linux 2.6 kernel is made to decrease worst time latencies but until now we did not have the opportunity to test this.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

4 Market specifics

4.1 System functions and characteristics

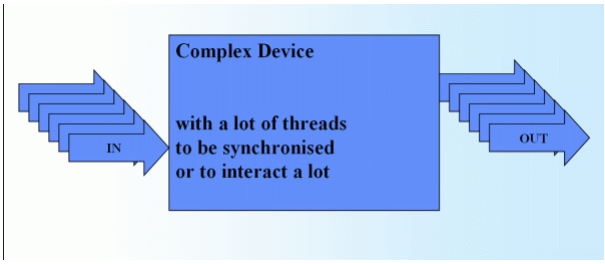
4.1.1 Approach

Before going into the specifics of the 5 specific markets we have chosen, we will describe different types of functionalities and characteristics one can find in embedded systems.

4.1.2 Number of inputs and outputs

The number of inputs and outputs is an important characteristic of an embedded system, because it shows how many simultaneous events should be handled simultaneously.

This Keep It Simple and Stupid (KISS) device has just one input and one output and can be looked at as a "data stream converter". However it is not always that simple which is pictured below: a complex system has multiple inputs and outputs to be dealt with simultaneously.



4.1.3 Events or messages

An input/output can be a single event (ex.: door opens/close valve) or a message or data stream (ex.: internet packet, audio stream) coming in or out. All of these events will be represented in the system by an interrupt to be dealt with. However, data might be involved (message based) or not. Most of the RTOS are good in one or the other handling but not necessarily in both.

4.1.4 Display

An embedded system might have a very limited display with some LED's or have a graphical display. With a graphical display (LCD) it then makes a difference what kind of data will be presented and as a consequence the (software) "refreshing" rate of the display. Of course, in most cases, one will accept a soft real-time approach for this display.

4.1.5 Subsystems subdivided in subsystems

As we have seen in 2.6, real-time is a matter of deadlines to meet and these deadlines may vary over different magnitudes (from nanoseconds to hours). Most systems therefore are subdivided in subsystems which might be implemented in hardware OR a processor + one program OR a processor using an RTOS with a multitasking approach.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

4.1.6 Intelligent sensors

An immediate consequence of the previous paragraph (4.1.5) is the approach where people start using intelligent sensors. Indeed, previously, most embedded systems did have some measurements to do like temperature, pressure etc. Sensors were used producing analogue signals and an analogue to digital converter sampled these analogue signals with the aid of a microprocessor. Such a program was very common. This approach has also a serious drawback because you are mixing analogue and digital signals in the same systems, which is not an easy thing to deal with.

Today, sensors are made as subsystems with all electronics (and optional software) to deliver digital signals (or messages) to the embedded system. The sensor in itself is then a small embedded systems with possibly an analogue to digital converter and some software running on a 4, 8, 16 and exceptionally a 32 bit processor delivering data in one or another way (on an industrial network for example), mostly without RTOS. However this could change in the future if the sensors become smarter and smarter and 32 bit processors become cheaper.

4.1.7 Predictable networks

As subsystems are now smart devices they communicate with each other via a network. Different networks can be used for that, but if the overall characteristic of the system needs to be real-time (= predictable) then the subsystems should be connected with predictable networks. This is typical in industrial systems where one can find industrial networks having real-time characteristics like Profibus, CAN etc... It is also interesting to know that USB and firewire have these characteristics too. Ethernet has not.

4.1.8 Power consumption

Power is a major concern in embedded systems, especially for portable devices. Long up-time requirements combined with low weight (battery) and enough CPU power is not an easy task.

4.1.9 Smart, smarter, smartest

Systems are today becoming more and more complex or smarter. Indeed, instead of making simple control decision, you can add more decision making intelligence today and add also much more monitoring functions to make the operation of the device safer. People only start to see, now that smart systems are becoming available, that they might become smarter and smarter. This imposes supplementary constrains to the (RT) behavior of the system and to the operating system.

4.1.10 Memory protection

20 years ago, when people started using software systems to control and supervise nuclear installations, one of the typical requirements for these systems was the use of a memory management unit to “protect” the data to be corrupted in a way or another. This was realized with the hard and software on the DEC PDP and VAX machines. Nobody saw the interest to have memory protection in embedded systems that time. We have always seen memory protection as an essential option - at least during the development cycle of the new device - because it helps you debugging the system much faster. Today however, people largely accept the need for this feature even in the production versions of the embedded system for multiple reasons (anti-hacking measure, data protection in billing functions, etc..) . Of course you pay a performance price for this feature.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

4.2 Markets and their systems

4.2.1 Quotations

In the next paragraphs we will describe different devices or systems used in different markets. We give advice about the use of this (RT) OS for these devices. Advice is given for VxWorks 5.3 and also VxWorks 6.1 although the advice of the latter is based on published new features, which we were not able to test on their behavior and performance for time being. A similar remark is valid for MVL Pro2.1 and the 4.0 version. Again the latter has not been tested.

For the sake of completeness in this advice sheets we added Embedded XP and Linux. The OS can be used for non RT embedded systems. With Linux we mean embedded or not embedded Linux based on the 2.4 or 2.6 kernel without any patches or additions.

The scale we use is:

++	Very well suited – excellent behavior and API for such an application. The product as shipped is ready to go for such a device.
+	Well suited, but designer might need to adapt his design to the available API.
0	Can be used but is not really indicated. Designer will need a lot of design workarounds before the system will work correctly. A lot of time might be lost during the design phase.
-	The use is not indicated. Too many tricks and workarounds will be necessary to make the system work correctly if ever possible.
--	The use in this case will lead to a non functional device or project failure.
+/- +/NA	Combination of 2 scales due to certain circumstances
NA	Not Applicable: If the OS can not be used for this purpose. For instance, none of the discussed OS supports 8-bit processors.
(?)	One might expect this rate but we are not sure – because we did not test this OS

We indicate what we believe is possible with the particular OS. Designers might have preferences for no particular reason, except that it is “for free”.

4.2.2 Industrial automation and manufacturing

Industrial automation and manufacturing uses a vast range of embedded (and less embedded) systems. It goes from robots to vast plant control and monitoring systems. The number of devices or subsystems involved is great. These subsystems are very different in nature.

- **An intelligent sensor** on an industrial network: there is no need for a display, the device falls in the category of KISS devices and most RTOS will be capable of handling the requirements as long as there is one input (the sensor) and one output (the data going on the network). A small footprint might be interested in this case and the fact you run on an 8 bit or 16-bit processor could be major requirements. Power consumption is not an issue except if the sensor is connected in a wireless way. Remark that most RTOS do not support 8-bit processors nor 16-bit processors anymore.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+/NA	+/NA	+/NA	--	--	--
+ for 32 bit processors, NA for 16 or 8 bit					

- **An intelligent actor** falls in the same category as the intelligent sensor
- **An industrial network control and monitoring device**: this device is monitoring the intelligent sensors and controlling the intelligent actors. This device has in most cases a display and an industrial keyboard. However another approach is to control the device with a PC connected via an Ethernet to the device. This device needs serious RT control, because it has simultaneous multiple inputs and outputs (on the RT network). We would not use an RTOS that did not qualify as RT-validated in our tests. We consider therefore that Linux is not an option.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++ (?)	++	-	-	--

- **PLC: Programmable Logic Controllers** are versions of the previous described devices where you permit the user to program the device in a special way. Some sensors and actors might be controlled directly by the PLC.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
++	++ (?)	++	--	--	--

- **Plant control and production planning**: all previous devices are connected through a non RT network (mostly Ethernet) together. The system has to keep track of a lot of devices working “in the field”. However, as the predictable (RT) job is done by the industrial network controller the network and handling requirements of the system are not RT. Linux and (embedded) XP will certainly do.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
--	-- (?)	0	+	++	++

4.2.3 Medical devices

Like in the previous case, there is a broad spectrum of devices used in this field.

- There is a whole range of **small portable monitoring and measurement** devices medical personnel can use to monitor patients or do chemical tests. The equipment has a small display and might even have a small printer included. It should be connectable to a notebook in order to recover the collected data and file it in the patient’s files. The price of the device should be low and sometimes very low because some industries are offering the device for free, in order to sell the chemical supports one need to do the measurements of a blood sample. The device almost falls in the category of consumer devices.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
++	+ (?)	+	--	--	--

- **Complex monitoring and alarming devices** used in hospitals in urgency and recovery units. These units have a display, a serious amount of sensors and need to be smart in order to take alarm decisions. They are connected to a network in order to signal the alarm function in a remote place. Precise and timely action is required. The system should be safe. There might be a need for fault tolerance. Budget is not an issue (as long as it remains reasonable)

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++ (?)	++	--	--	--

- **Scanners and beam irradiators** are complex and vast devices. There are a lot of sensors, a moving table with the patient and a control unit with very high resolution display. These devices are that complex that they need to be subdivided in different subsystems. Functionally it is like controlling the plant as in industrial automation, however, in this case a lot of processor power might be needed to produce a view of the scan and the system should be safe and secure. The image processing function is NOT a RT function. Controlling the scan and insuring all data is collected is a serious RT function where no misbehavior is allowed. Budget is not an issue at all; the number of produced machines is very limited. The table below is focused on the control and data acquisition part of the machine.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++ (?)	++	-	--	--

4.2.4 Consumer electronics

Consumer electronics is nowadays also a market with a great variety of devices (and requirements). It is especially characterized by the high number of devices produced which should be low cost for competitive reasons.

- **Small portable (smart) devices** such as cell phones, PDA's, watches, VOIP phone: some have a limited display but there is a tendency to have more resolution. A lot will depend from what functionality is made in hardware. As you need to invent and produce a new model almost every 6 months, time to market is important. Therefore the number of functions readily available and offered by the OS vendor is of high importance in this market.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	+	++	+	0	0

- **Audio and video equipment**: digital photo camera, digital video camera, wma-mp3 players, (tape and DVD video equipment), television sets: the same remarks as in the previous point are valid for this market.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	+	++	+	0	0

- **Computer peripherals**: personal printer, scanners, office automation equipment (large multifunctionals): inkjet printers are never a big issue, nor in stand alone, nor in a multifunctional because you can stop the printing function if you have something more urgent to do. Laser printers

are a more difficult challenge because, once you decided to print, you have to go for a full page at once and keep the data stream going. Personal low cost small format (A4 or letter) laser printers will probably try to do with one processor and one RTOS which then needs serious RT requirements. The larger laser based systems and multifunctionals will have different subsystems and the user interface, network and storage functions have simple or no RT-requirements.

Low cost simple laser printer or printing subsystem for a larger device					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
++	++	++	--	--	--
Interface, storage and networking subsystem in a larger device					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	0	+	++	++	+
Both parts together on one processor in a low cost laser multifunctional with storage and graphical interface					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	0	++	--	--	--

- **Gambling and gaming machines**: as user graphical interface's are important CE and XP embedded are good choices in the graphical subsystem. Another subsystem has the machine's control function with very requiring RT capabilities. If indeed you need to deliver exactly a number of (money) pieces after a win, you better count these exactly (firm real-time)

The graphical subsystem					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
-	0	++	0	++	0
The controller part (if done with 32 bit processor)					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
++	++	++	-	-	--
Both parts together on one processor (may be less costly than a 2 subsystem approach)					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
-	0	++	-	-	--

- **Home router/gateways**: typical functions are a firewall, VPN etc.. The device is also controllable from a PC using a web browser. This means that the device should have and embedded web server included. The device is connected to the network with a limited bandwidth. The number of devices build is high and some functionality is implemented in hardware.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	++	0	++

- **Professional network device** (switches, routers gateways): The switching function is realized in hardware, no single processor with a OS could give a solution here. However other functionalities such as management and statistical functions can be done with a processor and (RT) OS. It largely depends on the functionality if RT is required for this function or not. Anything will probably so if no RT is required. Some RT requirements will exclude plain Linux and Embedded XP. Serious RT requirements will exclude MVL.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	++	0	+

- **Set top box:** it is a bridge between your television set and the internet. It contains protocol stacks, a web browser etc. Here it is important that the operating system has these Internet modules readily available. The box will certainly do with a soft real-time approach.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	++	++	++	++	+

4.2.5 Retail (Thin Clients/POS)

- **Thin clients – smart displays**: these devices are embedded versions of some PC functionality. Windows like OS will very well do here, other OS may be used but much more design work will be needed to arrive to a similar result.

VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
--	0	++	0	++	0

- **Point of Sales (POS)**: a point of sales can be considered as an embedded PC like device. It has a (special) keyboard (or touch screen) as input, an (optional) money drawer, a small printer, a display (small screen or long LCD) and a network connection to a central computer. It should accept card payment modules in different ways. As long as it does not control anything else, there is no need for serious RT functionality. However, if the POS is controlling simultaneously some other elements, that's another story.

Simple POS					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	+	+	++	++	+
POS with complementary functions like barcode scanner, CC reader, etc..					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	+	++	+	+	0

-

- **Vending machine**, which becomes more and more networked to send status information to the provider. The machine can mostly be split into 2 functions: a industrial controller function to produce something (coffee, deliver cans, etc..) and a network function (gateway) to communicate via Internet with the provider. The controller part

The controller part					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	+	++	0	--	--
Gateway part					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	++	0	++
Both parts together on one processor (less costly than a 2 subsystem approach)					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	+	++	0	-	0

- **Kiosk**: this may range from a simple device displaying information in a hotel lobby to a more complex device delivering tickets in a train station or clearing parking tickets were payment with cash and CC cards are allowed.

Simple kiosk with information function					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
-	-	+	+	++	+
Sole ATM function as a subsystem					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	0	-	-
Both parts together on one processor (less costly than a 2 subsystem approach)					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	+	++	0	-	-

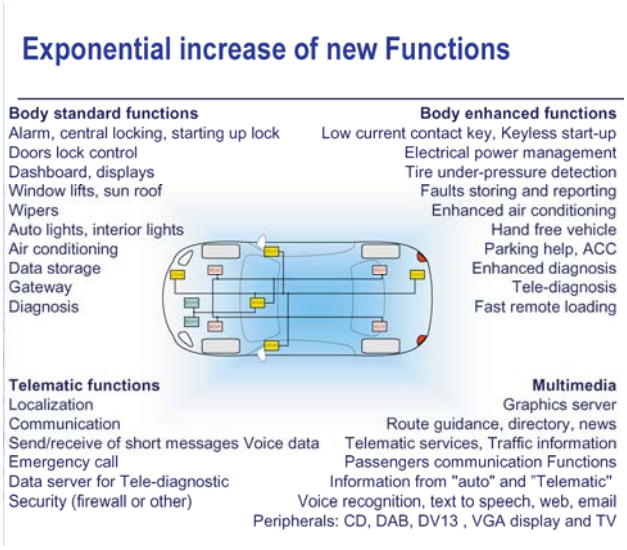
http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

4.2.6 Automotive

A car could me more and more compared to a factory in the coming decennium. As shown in the picture, a tremendous number of different functions will be implemented going from total drive by wire where RT functions are the top requirement to entertainment functions were the RT requirements are less stringent.

The discussed RTOS are not usable for the first category and a timed triggered operating mode is used. The discussion of this falls outside the scope of this paper.

However for most of the less RT requiring functions the studied RTOS can be used.



Body standard functions					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
0	+	+	++	++	++
Telematic functions					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	++	+	+
Body enhanced functions					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	++	++	+	-	-
Multimedia					
VxWorks 5.3	VxWorks 6.1	CE 5.0	MVL 2.1/4.0	Embedded XP	Linux
+	+	++	++	+	+

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

5 To finish with: yesterday, today, tomorrow

As the building block of an RT (embedded) system, an RTOS should in all cases behave predictably. This means always the same behavior and one bounded in time, independent of the system's load, the length of the queues in the system and the number of simultaneous interrupts or event detections.

Software real-time technology (we mean software real-time and not soft real-time) using an operating system has been introduced in the late 60ties with the PDP systems from DEC. However these systems were rapidly used for business applications. The use in real-time factory automation was only known by a limited number of engineers. This real-time minicomputer "track" joined the microprocessor "track" only later when microprocessors started to have sufficient processor power.

Early RT systems were built around discrete mechanics, electro-mechanics and electronics (in that order). Programmable systems were introduced with the microprocessor in the 80ties, but the importance of software in such systems was limited. Most systems were written in assembler and the most efficient "tool" at that time was a C-compiler.

Only since 1990 software real-time science became a reality (when both the minicomputer and microprocessor tracks joined). Engineers showed the great potential and flexibility of a maximal use of software in such systems, but also gradually they became aware of the big challenges.

This document showed the actual state of the art of commercial products and shows indirectly that we are dealing with technology which is far from mature. This also explains the purpose of our evaluation project. Indeed it helps understand the requirements of actual embedded systems, the way vendors try to respond to that with new (releases of) RTOS products which are not necessarily without flaws.

From the market study we learn that a lot of devices except the small ones are subdivided in subsystems. This situation is sometimes misused in promotion campaigns of one or another RTOS vendor. You may for example promote the fact a particular (RT) OS is used in a large professional multifunctional, but probably everybody can claim this. It depends on what subsystem you are talking about.

It will need many more years of research to put some structure and order in this field. However the gap between research and industry is (too) great today. Industry is continuing with too much trial and error (this is called customer feedback). However the customer does not necessarily know himself what he really needs. Education is indeed lacking and everybody has his personal definitions and desires.

Only recently, the tremendous possibilities of smart embedded (RT) systems are discovered and one might expect that this will stimulate research and industry to better work together.

If people start requiring better quality of service for a particular device, RT will be a must. Some have it now, others are fighting to get it work, but we don't believe that "patching" is the solution.

6 Appendix: The Rating base

For completeness, we include here the ratings we apply to evaluate an RTOS. These ratings are given at the end of a complete evaluation which includes installation of the product WITHOUT support (to check the documentation). Afterwards, during the tests, incidents happen and support is asked for from the vendor permitting us to see how the vendor reacts on the sometimes complex questions asked.

	0-3	4-6	7-8	9-10
Installation and Configuration	*Very difficult*, Unable to install the product without the help from technical support.	*Average*, Manageable, but unexpected problems still occur.	*Easy*, The product can be installed without any problems. Still some basic knowledge is needed.	*Very easy*, There is really nothing to it.
RTOS Architecture	*Inappropriate*, Totally inappropriate for real-time systems.	*Acceptable*, Workable architecture, but may have some shortcomings.	*Appropriate*, An appropriate architecture for an RTOS.	*Excellent*, Flexible and extensible architecture.
API Richness	*Extremely poor*, Several essential features for an RTOS are missing.	*Poor*, Usable, but still some features missing.	*Adequate*, Complete enough for general use.	*Comprehensive*, Everything you will ever need is in there. Makes programming easier.
Internet support	*Insufficient*, Offers almost no support.	*Adequate*, Some internet capabilities are provided for.	*Good*, Enough internet support for most applications.	*Extensive*, Plenty of support available.
Tools	*Not sufficient*, Almost no tools are provided.	*Adequate*, Manageable, but some basic tools are missing.	*Good*, All the important tools are present.	*Extensive*, Plenty of tools available.
Documentation and Support	*Poor*, Not complete, chaotic and hard to read.	*Reasonable*, The documentation is usable, but some parts are incomplete or unclear.	*Good*, The documentation is clear and complete enough for most users	*Excellent*, The documentation is very clear and comprehensive.
Test Results	*Unacceptable*, Does not exhibit sufficient predictable behavior for an RTOS.	*Fair*, Predictable in most cases but still some problematic issues	*Good*, What you would expect from a good RTOS.	*Excellent*, Completely predictable and fast.

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com

7 Content table

1 Preface 2

2 Real-time systems & RTOS 3

2.1 What is a real-time system? 3
2.2 Real-time system components 3
2.3 Real-time system types 3
2.4 Modern characteristics for embedded systems 4
2.5 Multitasking or multithreading 4
2.6 Deadline spectrum 5
2.7 Hardware freedom 5
2.8 This chapter's conclusion 6
2.8.1 Why is RT important? 6
2.8.2 Which aspects are important when evaluating a real-time solution? 6
2.8.3 The way ahead 6

3 Comparing some RTOS products 7

3.1 Introduction 7
3.2 Summary 7
3.3 VxWorks 5.3.1. – Wind River Systems 8
3.3.1 Ratings (scale in appendix: chapter 6) 8
3.3.2 Observations 8
3.3.3 New product versions 9
3.4 Windows CE 5.0 – Microsoft 9
3.4.1 Ratings 9
3.4.2 Observations 9
3.4.3 New product versions 10
3.5 Montavista Linux Professional Edition 2.1 10
3.5.1 Ratings 10
3.5.2 Observations 10
3.5.3 New product releases 11

4 Market specifics 12

4.1 System functions and characteristics 12
4.1.1 Approach 12
4.1.2 Number of inputs and outputs 12
4.1.3 Events or messages 12
4.1.4 Display 12
4.1.5 Subsystems subdivided in subsystems 12
4.1.6 Intelligent sensors 13
4.1.7 Predictable networks 13
4.1.8 Power consumption 13
4.1.9 Smart, smarter, smartest 13
4.1.10 Memory protection 13

4.2 Markets and their systems 14
4.2.1 Quotations 14
4.2.2 Industrial automation and manufacturing 14
4.2.3 Medical devices 15
4.2.4 Consumer electronics 16
4.2.5 Retail (Thin Clients/POS) 18
4.2.6 Automotive 20
5 To finish with: yesterday, today, tomorrow 21
6 Appendix: The Rating base 22
7 Content table 23

http://www.dedicated-systems.com
Email: m.timmerman@dedicated-systems.com