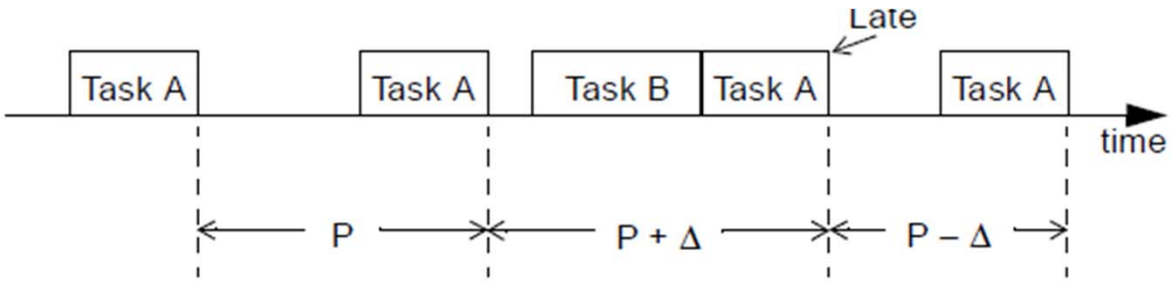
A Real Time Operating Systems (RTOS) Comparison

Abstract. This article presents quantitative and qualitative results obtained from the analysis of real time operating systems (RTOS). The studied systems were Windows CE, QNX Neutrino, VxWorks, Linux and RTAI-Linux, which are largely used in industrial and academic environments.

Windows XP was also analysed, as a reference for conventional non-real-time operating system, since such systems are also commonly and inadvertently used for instrumentation and control purposes.

The evaluations include worst case response times for latency, latency jitter and response time.

RTOS Performance Issues : Real Time Jitter



RTOS can significantly can increase the amount of jitter and reduce the predictability of the real-time system's behavior.

This lack of predictability is due to portions of the RTOS code that are called frequently and with execution times that are quite large or that change with respect to variable system parameters. This reduced predictability may prohibit an application from meeting its design constraints.

HARDWARE SUPPORT FOR REAL-TIME OPERATING SYSTEMS, Paul Kohout, Master of Science, 2002, University of Maryland, College Park

Most Valued RTOS Features

Real Time Operating Systems (RTOS) are specially designed to meet rigorous time constraints. In several situations RTOS are present in embedded systems, and most of the time they are not noticed by the users.

Evans Data Corporation's Surveys reveal the *Top* 5 RTOS Features most valued by developers:

Real-time responsiveness(33.2%)
Royalty-free pricing(14.7%)
Source code availability(10.6%)
Tools integration(IDE) (10.1%)
Microprocessor coverage(7.8%)

Ref: Logic, Inc. Thread-Metric Benchmark Suite

Most Valued RTOS Features

RTOS Performance is the speed with which an RTOS completes services for an application

•RTOS Performance is platform-sensitive
processor-sensitive
clock-speed sensitive
compiler-sensitive
design-sensitive

•Performance also is “context sensitive”

Ref: Logic, Inc. Thread-Metric Benchmark Suite

Most Valued RTOS Features

- Cooperative Context Switching
–Preemptive Context Switching
–Interrupt Processing Without Preemption
–Interrupt Processing With Preemption
–Message Passing
–Semaphore Processing
–Memory Allocation/De-allocation

A semaphore is a system resource used to guarantee a task exclusive access to a critical resource. Semaphores synchronize asynchronous activities

Ref: Logic, Inc. Thread-Metric Benchmark Suite

Most Valued RTOS Features

Applications commonly must allocate memory dynamically to avoid the need to plan for the maximum possible memory needs of all tasks at the same time

Ref: Logic, Inc. Thread-Metric Benchmark Suite

RTOS Usage

A good example of this situation may be observed in the automobile industry, where it is estimated that 33% of the semiconductors used in a car are microcontrollers, being common for a car to have dozens of microcontrollers. As a consequence, the manufacturing costs of vehicles are reaching proportions that existed only in the aerospace industry, where 1/3 of the total cost of a vehicle is spent in the chassis, 1/3 in the power train and 1/3 in electronics.

RTOS Usage Scenario : Automobile Sector

Seeking for improvements on its products and development time reduction, the car manufacturers have been adopting RTOS to control the software that runs in the vehicles.

A good example, is the electronic injection of fuel into the car engine, which must be made with rigorous time constraints.

At each motor cycle, sensors need to measure and analyse the output gases generated by the combustion, and then compute the next mixture combination before the next ignition happens.

Additionally, it is known that nowadays even simple motors, such as the ones used in motorbikes, already uses RTOS in their software.

RTOS Usage Scenario : Avionics Sector

Recently developed avionics control systems use a single computer to cope with several aircraft subsystems, thus requiring an operating system with temporal and spatial partitioning systems.

Spatial partitioning refers to tasks isolation in the computer memory, while temporal partitioning refers to tasks scheduling, dividing the processor time properly.

These partitions allow a single processor to execute several tasks simultaneously, without the risk of one task causing interference in the temporal requirements of other tasks.

This approach allows reduction of computers required to fly a plane, making it lighter.

RTOS Evaluation : A Difficult Task?

In the international market, there are more than a hundred options of RTOS to choose from, while additionally there are free similar options as the Linux operating system.

In this way, the decision on which system to use may be a key factor to the success or failure of a project, and the analysis must be made with impartiality and adequate criteria.

RTOS Evaluation : Methodology

Comparing RTOSs is not a trivial task. Besides this, the specialists from Dedicated Systems, an institution that has several projects and publications related to RTOS comparison, states that it is not possible to measure characteristics of a RTOS with reliability without using external hardware

Express Logic, Inc. Thread-Metric Benchmark Suite

RTOS Evaluation : Thread Metric Suite

“Thread-Metric,” is a free-source benchmark suite for measuring RTOS performance

•The Thread-Metric suite is freely available from Express Logic

•Performance of Express Logic’s ThreadX®RTOS is provided for reference

•Thread-Metric is easily adapted to other RTOSes

Express Logic, Inc. Thread-Metric Benchmark Suite
www.expresslogic.com

RTOS Evaluation : Thread Metric Suite

Lots of iterations, 30-second reporting

Execute (service, inverse) in pairs

Send message; get message
Allocate memory; de-allocate memory
Keep counts in local variables
“Printf” results to host every 30 seconds

- Calibration run to establish baseline
- No special hardware required
- Easily ported to new environments
- Coded in “Vanilla” C

Tested with various compilers
RTOS functions identified for adaptation

Express Logic, Inc. Thread-Metric Benchmark Suite

RTOS Evaluation : Important Aspects

The most important factors of real time systems are the worst case response time of a task and worst case response time of an interrupt.

However, it makes no sense to analyze real time operating systems metrics such as interrupt latencies and task switching time without considering different CPU usage scenarios, as it is easier for a system to be more predictable when it is not overloaded.

RTOS Evaluation : Important Aspects

The most important specification of a real time system is the amount of time that interrupts are disabled, because interrupt latency is a component of the system response time.

Additionally, response time measures of external interrupts gives a good idea of the real time capabilities related to a specific system or application.

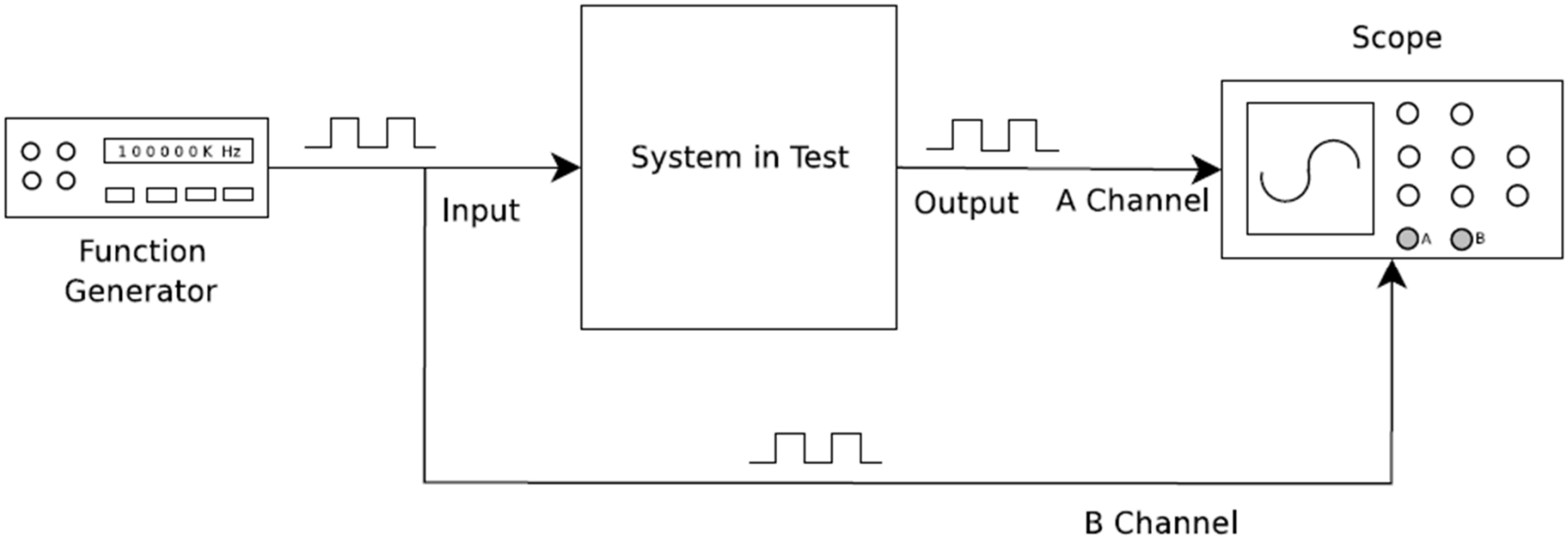
RTOS Evaluation : An Approach

Use the PC parallel port to receive an interrupt and generate a response to this interrupt, allowing testing the system as a black box. Using an external signal generator and an oscilloscope, it is possible to obtain the latency to handle interrupts and jitter (a random variation from one latency measurement to another), as one of the most accurate methods to measure execution time is through output ports

RTOS Evaluation : Approach Used

The tests were conducted generating external stimuli with a signal generator, and analyzing the response for these stimuli with an oscilloscope. To guarantee the results reliability, all the experiments were executed in the same platform (a Pentium II 400MHz PC with 256MBytes of RAM memory) submitted to several different load scenarios (normal and overloaded use).

Experimental Set Up used



Interrupt Latency Measurement

Although interrupt latency measurement through external mechanisms is commonly used, the International Society for Measurement and Control ,through its committee for automation and control (Automation & Control Systems Committee- A&CS) do not recommend using latency as a measure of real time response.

The test proposed by the A&CS consists in setting a system that copies the input signal directly to an output port, and measuring how many pulses were generated in the input and how many were copied to the output. Theoretically, while the system is stable, the accumulated number of pulses in both ports should be equal.

Interrupt Latency Measurement

Later, the input signal frequency should be slowly incremented until the input and output pulses count starts to diverge. In this moment, the frequency should be reduced until the maximum system operation frequency is found. This frequency is the one that does not cause difference in the pulse count from the input to the output port.

Selected Parameters for Measurement : Latency

Latency: Latency is analyzed externally taking the RTOS under test in conjunction with the hardware as a black box. The latency consists of the time difference between the moment that an interrupt is generated and the moment that the associated interrupt handler generates an external response.

The latency was measured in a scenario with low CPU use and with the CPU overloaded. For each scenario 60 independent samples were taken.

Selected Parameters for Measurement : Jitter

Jitter: Jitter is an indirect information obtained from several latency measures, consisting of a random variation between each latency value.

In a RTOS, the jitter impact could be notorious while controlling step motors.

For example, the pulses duration controls the motor rotation, but the jitter induce the torque to vary, causing step losses in the motor.

To compute jitter, the time difference between two consecutive interrupt latency measures is calculated. Finally, the greatest encountered difference is selected as the worst jitter of this system

Selected Parameters for Measurement : Response Time

Worst Case Response Time: Worst Case Response Time is obtained using the method of analyzing the maximum interrupts frequency that is handled by the RTOS with reliability.

The worst case response time is the inverse of the maximum frequency obtained. The test was made in a low CPU usage scenario and in an overloaded CPU scenario. For each scenario, 60 independent samples were taken.

MicrosoftWindows CE Embedded 6.0

The studied version of Windows CE is the Embedded 6.0. It supports ARM, MIPS, SH4 and x86 architectures. The tests were performed in the x86 architecture.

Microsoft Windows CE Embedded 6.0

The clock interrupt is the “heartbeat” of the system. In most systems, this is a constant rate interrupt generated by a hardware clock to trigger system’s house keeping routines.

Windows CE introduces an interesting innovation in this aspect: the variable clock tick, to reduce the overhead that the clock tick could cause in the operating system.

For example, the variable clock tick system verifies that in a certain moment it is not necessary to generate clock tick interrupts at each 1ms, but only at each 100ms, changing the clock tick interrupt frequency.

This allows the system to adjust the tick rate according to each situation. This also implies in energy saving and more computing power.

QNX Neutrino

QNX Neutrino is one of the most traditional RTOS in the market. Based in the microkernel architecture, it is completely compatible with the POSIX standards and has been certified by FAA DO-278 and MIL-STD-1553 standards.

In the microkernel architecture, the kernel implements only four basic services: task scheduling, inter task communication, low level network communication and interrupt handling.

All the remaining parts of the system (device drivers included) are implemented as user tasks, making the kernel fast, reliable and small. The tested version of Neutrino is 6.1.

QNX Neutrino

One of the advantages of the microkernel over other types of kernels is that even when a critical error happens, such as in the file system for example, all the remaining parts of the system are not influenced.

This means that the microkernel architecture offers a more robust environment than the ones used in other operating systems, although its problem is the overhead caused by the memory protection which have to be used very frequently, as all the parts of the system are isolated.

QNX Neutrino

Neutrino supports ARM, MIPS, PowerPC, SH4 and the PC architecture. A recent feature of this operating system is called adaptive partitioning, which enables the creation of processor constraints by tasks.

One example, is limiting the processor use of task
A to no more than 30% of CPU time, and task B to no more than 10% of CPU time.

Micrium µC/OS-II

µC/OS-II stands for Microcontroller Operating System Version II consisting of a highly simplified, yet powerful real time kernel developed by Jean Labrosse, an embedded systems designer with more than 20 years of experience in the area. Labrosse is one of the most known authorities in the RTOS subject

the decision of writing a new real time kernel from the beginning occurred because other commercial solutions were not meeting the quality requirements for the products he was developing. Today he owns a company called Micrium, which supplies this kernel.

Linux OS

Linux is a free operating system, with a modular monolithic kernel where all the important parts of the operating systems are in kernel space, such as memory management, task scheduler, file system and device drivers.

It is possible to dynamically add or remove parts and functions of the kernel using Linux Kernel Modules (LKMs). Linux kernel implements memory protection with the MMU aid. The evaluated Linux kernel was 2.6.18.

Linux OS

Regarding real time systems, Linux is not a real time operating system, although, there is a low latency kernel patch called low-pre-empt patch that can be applied to the mainstream Linux to add soft real time capacity to the system.

Linus Torvalds himself(Linux's author) affirmed in 2006 that he would use Linux to control a laser cutting machine using this patch.

Adding more rigorous real time constraints, is not an easy task. Including hard real time guarantees in a kernel with millions of lines of code is very complex and could lead to errors.

Real Time Linux

Real Time Linux has been recently offering a good cost/benefit relation in the real time area, being used in small applications, tests, and even in medical and state of the art scientific equipments

Real Time Linux is so well consolidated that well known institutions with RTOS needs have been using Real Time Linux in some applications.

Examples are the American National Institute for Standards and Time (NIST) and the National Aeronautics and Space Administration (NASA).

NASA itself, already used Real Time Linux in production to build a radar that collects data from tornados

RTAI Features

RTAI approach consists in isolating the operating system into domains, making the real time domain as the one with highest priority.

Whenever a real time task needs to be executed, a tiny scheduler, also called a nanokernel, schedules this task, freezing the whole remaining system.

When there are no pending real time jobs to be done, the other parts of the system, such as GUI and user interface are executed.

This enables the use of a single computer to control both real time critical functions and user interface, networking or others features that can be added at any time without changing the real time performance.

Real Time Application Interface (RTAI)

Although there are several different versions of Real Time Linux, the chosen version for analysis is RTAI.

RTAI has real time performance comparable to the best RTOS such as VxWorks and QNX.

However it does not offer any certification or guarantee to its users, as it is a free software.

Windriver VxWorks

VxWorks is the most used RTOS in the embedded systems industry.

VxWorks is used is the international space station and in the famous NASA rover robots Spirit and Opportunity.

This RTOS had been certified by several agencies and international standards for real time systems, reliability and security-critical applications.

The evaluated version used was VxWorks 6.2 which takes advantage of the system's MMU to isolate and protect the tasks.

Comparison of RTOSes

Table 1. Worst times measured during the experiments. A: Response Time (1/maximum sustained frequency), B: Latency, C: Latency Jitter

	Win XP	Win CE	Neutrino	μC/OS-II	Linux	RTAI	VxWorks
A	200μs	20μs	20μs	1,92μs	13,89μs	5μs	3,85μs
B	848μs	99μs	35,2μs	3,2μs	98μs	11,4μs	13,4μs
C	700μs	88,8μs	32μs	2,32μs	77,6μs	7.01μs	10,4μs

Conclusion

The well consolidated systems QNX Neutrino and VxWorks from Wind River, did really show determinism and reliability, although two newer systems that are not widespread also showed promising characteristics: Windows CE and RTAI Linux.

Windows CE Embedded was written from scratch especially for critical applications, and during the tests it behaved as a robust, powerful and flexible system.

In the free open source domain, RTAI offers the opportunity of implementing reliable real time systems with free software, having all the advantages of the Linux community and already available free software that could be used together.

Reference

A Real Time Operating Systems (RTOS) Comparison
Rafael V. Aroca1, Glauco Caurin1 1Laborat´orio de Mecatrˆonica Escola de Engenharia de S˜ao Carlos (EESC) – Universidade de S˜ao Paulo (USP) Av. Trabalhador S˜ao-Carlense, 400 – CEP 13566-590 – Caixa Postal 359 S˜ao Carlos – SP – Brasil