http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be
reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Doc. No: EVA-2.9-TST-CE-x86-01
Doc. Version: 1.00 Doc. date: 07 October, 2004

WINDOWS CE 5.0 ON AN X86 PLATFORM

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Disclaimer

Although all care has been taken to obtain correct information and accurate test results, Dedicated Systems Experts and Dedicated Systems Magazine 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 and Dedicated Systems Magazine have been advised of the possibility of such damages.

http://www.dedicated-systems.com

E-mail: info@dedicated-systems.com

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

EVALUATION REPORT LICENSE

This is a legal agreement between XXX and the company DEDICATED SYSTEMS EXPERTS.

1. **GRANT**. Subject to the provisions contained herein, DEDICATED SYSTEMS EXPERTS hereby grants XXX a non-exclusive license to use its accompanying proprietary evaluation report for projects where XXX is involved as major contractor or subcontractor. XXX is not entitled to support or telephone assistance in connection with this license.
2. **PRODUCT**. DEDICATED SYSTEMS EXPERTS shall furnish the evaluation report to XXX electronically via Internet. This license does not grant XXX any right to any enhancement or update to the document.
3. **TITLE**. Title, ownership rights, and intellectual property rights in and to the document shall remain in DEDICATED SYSTEMS EXPERTS and/or its suppliers or evaluated product manufacturers. The copyright laws of Belgium and all international copyright treaties protect the documents.
4. **CONTENT**. Title, ownership rights, and an intellectual property right in and to the content accessed through the document is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives XXX no rights to such content.
5. **XXX CAN NOT**:
   - XXX can not, make (or allow anyone else make) copies, whether digital, printed, photographic or others, except for backup purposes. The number of copies should be limited to 2. The copies should be exact replicates of the original (in paper or electronic format) with all copyright notices and logos.
   - XXX can not, place (or allow anyone else place) the evaluation report on an electronic board or other form of on line service without authorization.
6. **INDEMNIFICATION**. XXX agrees to indemnify and hold harmless DEDICATED SYSTEMS EXPERTS against any damages or liability of any kind arising from any use of this product other than the permitted uses specified in this agreement.
7. **DISCLAIMER OF WARRANTY**. All documents published by DEDICATED SYSTEMS EXPERTS on the World Wide Web Server or by any other means are provided "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. This disclaimer of warranty constitutes an essential part of the agreement.
8. **LIMITATION OF LIABILITY**. Neither DEDICATED SYSTEMS EXPERTS nor any of its directors, employees, partners or agents shall, under any circumstances, be liable to any person for any special, incidental, indirect or consequential damages, including, without limitation, damages resulting from use of OR RELIANCE ON the INFORMATION presented, loss of profits or revenues or costs of replacement goods, even if informed in advance of the possibility of such damages.
9. **ACCURACY OF INFORMATION**. Every effort has been made to ensure the accuracy of the information presented herein. However DEDICATED SYSTEMS EXPERTS assumes no responsibility for the accuracy of the information. Product information is subject to change without notice. Changes, if any, will be incorporated in new editions of these publications. DEDICATED SYSTEMS EXPERTS may make improvements and/or changes in the products and/or the programs described in these publications at any time without notice. Mention of non-DEDICATED SYSTEMS EXPERTS products or services is for information purposes only and constitutes neither an endorsement nor a recommendation.
10. **JURISDICTION**. In case of any problems, the court of BRUSSELS-BELGIUM will have exclusive jurisdiction.

Agreed by downloading the document via the internet.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

1 Introduction ... 6
1.1 Purpose and scope ... 6
1.2 Document issue: the 2.9 framework ... 6
1.3 Related documents ... 6
2 Results summary ... 8
2.1 Product under test ... 8
2.2 Test result ... 8
2.2.1 Positive points ... 8
2.2.2 Negative points ... 8
2.2.3 Ratings ... 8
3 Introduction ... 9
3.1 Product under test ... 9
3.1.1 Software ... 9
3.1.2 Hardware ... 9
3.2 Introduction ... 9
4 Installation and BSP ... 10
4.1 Installation ... 10
4.1.1 Installation on Host ... 10
4.1.2 Installation on target ... 11
4.2 BSP ... 11
5 Test Results ... 12
5.1 Calibration system test (CAL) ... 13
5.1.1 Tracing overhead (CAL-P-TRC) ... 13
5.1.2 CPU power (CAL-P-CPU) ... 14
5.2 Clock tests (CLK) ... 15
5.2.1 Operating system clock setting (CLK-B-CFG) ... 15
5.2.2 Clock tick processing duration (CLK-P-DUR) ... 15
5.3 Thread tests (THR) ... 19
5.3.1 Thread creation behaviour (THR-B-NEW) ... 19
5.3.2 Round robin behaviour (THR-B-RR) ... 20
5.3.3 Thread switch latency between same priority threads (THR-P-SLS) ... 21
5.3.4 Thread creation and deletion time (THR-P-NEW) ... 28
5.4 Semaphore tests (SEM) ... 38
5.4.1 Semaphore locking test mechanism (SEM-B-LCK) ... 38
5.4.2 Semaphore releasing mechanism (SEM-B-REL) ... 39
5.4.3 Time needed to create and delete a semaphore (SEM-P-NEW) ... 39
5.4.4 Test acquire-release timings: contention case (SEM-P-ARN) ... 45
5.4.5 Test acquire-release timings: contention case (SEM-P-ARC) ... 49
5.5 Mutex tests (MUT) ... 54
5.5.1 Priority inversion avoidance mechanism (MUT-B-ARC) ... 54
5.5.2 Mutex acquire-release timings: contention case (MUT-P-ARC) ... 55

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6 Memory tests 63
5.6.1 Memory leak test (MEM_B_LEK) 73
6 Support 74
7 Appendix A: Vendor comments 74
8 Appendix B: Acronyms 76
9 Appendix C: Document revision history 77
9.1 Issue 1.0 (July 12, 2004) 77

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

DOCUMENT CHANGE LOG

Issue No.	Revised Issue Date	Para's / Pages Affected	Reason for Change
1.00	October 7, 2004	All	Initial Issue

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

1 Introduction

1.1 Purpose and scope

This paper presents the test results of the operating system and hardware platform under evaluation.

The layout and the content of this report follow the one depicted in "The evaluation report definition." [Doc. 3] and "The evaluation report template." [Doc. 4]. See section 1.3 of this document for more detailed references. Therefore these documents have to be seen as an integral part of this report.

The results of these tests can not be interpreted by the reader without knowledge of the test methodology used. Therefore the evaluation report definition is attached as appendix to this report. It contains the detailed description of each test performed (using the same test name). The generic code of each test is also downloadable on Dedicated Systems website (http://www.dedicated-systems.com).

Due to the tight coupling between these documents, the framework version of "The evaluation report definition." must match the framework version of this evaluation report (which is 2.9). More information about the document versioning, tests and the relations between both can be found in "The RTOS evaluation framework.", see [Doc. 1] in section 1.3 of this document.

The software (RTOS) and the hardware platform tested in this report are shown in section 3.1 of this document.

1.2 Document issue: the 2.9 framework

This document shows the test results in the scope of the evaluation framework 2.9.

1.3 Related documents

These are documents that are closely related to this document. They can all be downloaded using following link:
http://www.dedicated-systems.com/encyc/buyersguide/rtos/evaluations

Doc. 1 The RTOS evaluation framework.
This document presents the evaluation framework. It also indicates which documents are available, the document naming convention, and how the numbering and versioning of the documents are related. This document is the base document of the evaluation framework.
EVA-2.9-GEN-01 Issue: 1 Date:

Doc. 2 What is a good RTOS?
This document presents the criteria that Dedicated Systems Experts use to give an operating system the label "Real-Time". The evaluation tests are based upon the criteria defined in this document.
EVA-2.9-GEN-02 Issue: 1 Date:

Doc. 3 The evaluation report definition.
This document presents the different tests used for this report together with the

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

flowcharts and the generic pseudo code for each test. Test labels are all defined in this document.
EVA-2.9-GEN-03 Issue: 1 Date:

Doc. 4 The evaluation report template.
This document presents the layout used for all reports within a framework.
EVA-2.9-GEN- Issue: TBD Date: TBD

Doc. 5 The OS evaluation report.
This document presents the quantitative evaluation of the OS being used for the tests where the results are presented here. This document is independent of the platform being used.
EVA-2.9-OS-TBD Issue: TBD Date: TBD

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

2 Results summary

2.1 Product under test

Windows CE version 5.0 from Microsoft Corporation on an x86 platform

2.2 Test result

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

2.2.1 Positive points

- Modular operating system, with a large amount of optional features.
- All protection primitives use priority inheritance.
- Stable real-time results, worst case improved compared with CE 4.0.
- Interfaces easily with other Microsoft Operating systems.

2.2.2 Negative points

- Some limiting factors: like the number of processes and amount of virtual memory.
- Platform builder isn't always reliable.
- Documentation could be improved further, although it is already better than previous version.

2.2.3 Ratings

For a description of the ratings, see [Doc. 3]. The first four ratings are the ones given in the theoretical evaluation (independent of the platform used for the tests) which can be found in [Doc. 5].

Category	Min	Rating	Max
RTOS Architecture	0	8	10
OS Documentation	0	6	10
OS Configuration	0	7	10
Internet Components	0	9	10
Development Tools	0	8	10
Installation and BSP	0	7	10
Test results	0	7	10
Support	0	8	10

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

3 Introduction

3.1 Product under test

3.1.1 Software

Windows CE 5.0 operating system from Microsoft.

For more information on the Windows CE 5.0 OS from Microsoft, see http://msdn.microsoft.com/embedded/default.aspx or http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceintro5/html/wce50oriWelcomeToWindowsCE.asp

The Operating System in this report has been configured as an Enterprise Terminal configuration (Release 15 Mb) and for comparison sake we tested the Tiny Kernel configuration (582Kb) as well.

The qualitative evaluation of this product is done in [Doc. 5].

3.1.2 Hardware

All the tests were executed using the following hardware:

- Motherboard: Chaintech 5TTMT M201 with a 33MHz PCI bus
- BIOS: Award BIOS v4.51PG
- CPU: Intel Pentium 200Mhz MMX Family 5 Model 4 Stepping 3 (with 32KB L1 Cache)
- RAM: 32 Mb
- Hard drive: Western Digital Caviar 22000 , Capacity 20Gb
- Graphic adapter: S3 trio6 TV2/DX
- Network interface card: The Realtek **RTL8139C**(L)
- VMETRO PCI exerciser in PCI slot 3 (PCI interrupt level D, local bus interrupt level 10)
- VMETRO PBT-315 PCI analyser in PCI slot 4.
- External and CPU internal cache was enabled during the tests, unless otherwise specified.

The qualitative evaluation of this platform is done in [Doc. 5].

3.2 Introduction

Microsoft has long been active in the operating systems market as a General Purpose Operating system vendor used for desktop and server systems. It is only recently that it came to the embedded market place as the mobile and embedded systems markets emerged. Indeed the first version of Windows CE was introduced in the fall of 1996.

It was with the introduction of Windows CE version 3.0, however, that the operating system was enhanced to address the needs of real-time applications.

This report is focused on version 5.0 of the operating system.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

4 Installation and BSP

Installation and BSP	0	7	10

Installation of the product toolchain went easy. The platform builder is easy to use for configuring a platform. Although it is not failsafe: when you want to customize your platform a lot it can break apart your configuration.

CE supports a large number of boards and drivers. Drivers are delivered in source code.

4.1 Installation

4.1.1 Installation on Host

The first step to use Windows CE 5.0 is installing the platform builder software. Platform builder 5.0 is the set of tools that is used to create a custom Windows CE 5.0 platform. The platform builder comes on a CD and supports ARM, MIPS, SH or Intel x86 based platforms. For this evaluation, only the Intel x86 component was installed and the configurations used are:

- Enterprise Terminal release configuration : approximately 15 Mb
- Tiny Kernel configuration : 582 Kb
- Enterprise Terminal Debug Configuration : approximately 30 Mb

Installing platform builder is similar to installing any other Microsoft software application, and is pretty straightforward and user-friendly.

The next step is to use the platform builder to create, customize and configure a platform. Configuring the platform to your requirements can be complicated in review to the expectations.

The platform builder integrated development environment (IDE) includes wizards for creating platforms and components. Together with the increased help system, this is a major improvement. Now it is possible to create a system for your device in a minimum of time. But there are still complications if you want to customize a configuration: dependencies can fail or files can get corrupted so that the build process needs to be restarted or even platform builder needs to be re-installed.

We had the impression that Platform Builder 5.0 has made a good progress in friendliness compared to previous versions in terms of ease of normal configurations, however once you leave the predefined configurations things still become difficult. Like for most RTOS, the graphical platform builder is build on-top of a large number of configuration files and scripts. If something doesn't work out like expected it can become hard to get back to a workable platform.

There is dependency checking build-in in the platform builder, but it isn't always correct. Also it doesn't add the needed other modules in the build when needed.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

4.1.2 Installation on target

☺ Installation on the target goes quick and easy. Platform builder 5.0 has a boot loader which works with Ethernet, serial port or an emulator. We used Ethernet for the evaluation. If you want to run an application on the OS you need to make a specific SDK to support your code.

4.2 BSP

☺ It is possible to configure BSPs for the hardware platform, this is easily implemented under the form of a wizard. You can create a new BSP, modify existing ones and even create global drivers.

Windows CE is a componentized operating system (OS) where features and drivers are selectable from a graphical IDE catalog. After the user configures the OS feature set and collection of device drivers from the catalog, a build dependency checker ensures that the image's feature set is self-consistent with all dependencies being met. Microsoft and other 3rd Party BSP vendors install their BSPs into the IDE catalog and thus make them available to customers with similar (or the same) hardware.

In addition to the device drivers, BSP developers will need to create an OEM Adaptation Layer (OAL) to abstract the kernel from the hardware implementation. The OAL is responsible for CPU and system board initialization and is used routinely during system usage. The OAL handles things like cache operations, system timers, device interrupt handling, etc.

Although the concept is good, it is not an easy task. You have to understand very well the hardware software border and how the kernel interacts with it. Remark that this statement is in general true for all embedded systems.

Once you try to build more complex configurations, the task becomes complicated with pit-falls. For instance a dependency checking is not always working correctly, and some changes can make the configuration uncompilable.

It is important to note that the drivers are delivered in source code. This is something that would not be done some years ago. This is a result of the Open Source market pressure on traditional operating system vendors.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5 Test Results

Test Results 0 7 10

No problems were detected, the worst case behavior improved a bit compared with CE 4.0, average performance is about the same.

For the evaluation we did the tests on various configurations, so that comparisons can be made. The different configurations used were:

- Enterprise Terminal (Release build)
- Enterprise Terminal (Debug build, with kernel debugger)
- Tiny Kernel (Release build)

We noticed great differences between Debug and Release configurations, but this is normal due to the extra features like stack checking and so on. The debug configuration also included the kernel debugger that allows you to halt OS execution (break) and step through the kernel (and device-drivers, etc.). Note that the kernel debugger can also be used in a release build.

In this report all tests were done on Release configurations, except some tests to show the differences between debug versions and release versions. We made a rather large configuration of an enterprise terminal (binary of 15 Mb) and a very tiny kernel version (binary of 582Kb).

The behavior tests gave the same results on both configurations so they will only be shown for one version in the report. The different configurations will only be compared in the performance tests.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.1 Calibration system test (CAL)

These tests are used to calibrate the tracing overhead compared with the processing power of the platform. This is important to understand the accuracy of the measurements done in scope of this report.

Also it measures the Processing power of the platform, so the results can be compared with the results on other platforms.

5.1.1 Tracing overhead (CAL-P-TRC)

This test calibrates the tracing system overhead. This is more hardware than OS related, but it is needed to correct the measured times. More details about how these measurement are performed can be found in the "The evaluation report definition." [Doc. 3], a must read for understanding this report.

In the rest of the report, the tracing overhead is subtracted from the results obtained.

Tracing accuracy depends here on the PCI clock (33MHz), as this is the minimum time frame that can be detected. In general, the results in this report are correct to +/- 0.03 µseconds (one PCI clock cycle). Therefore the results shown in the tables are rounded to the nearest 0.1 microsecond.

5.1.1.1 Test results

Test	result
Average tracing overhead	209.1 nsec
minimum tracing overhead	209.1nsec
maximum tracing overhead	209.1 nsec
tracing accuracy	<0.1 µsec
Critical section primitive present?	YES

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.1.2 CPU power (CAL-P-CPU)

This test will calibrate the CPU performance and the memory bandwidth of the platform being used. This test measures the same algorithm when cached (looped) or not cached (un-looped) code and data. As such the effects of the cache can be calculated and performance of platforms can be compared with other platforms. In this test report our standard platform is being used.

Worst case behaviour is caused by caching issues, so this is an important measure to predict worst case delays.

Again, to understand how these tests are run and what exactly they are measuring you will need to read the “The evaluation report definition.” [Doc. 3]. This document has to be considered a part of this report.

5.1.2.1 Test results

The test on our standard platform (Pentium MMX 200 MHz):

Test	no cache	cached	cache effect
CPU test duration	401.9 us	270.8 us	1.48
MEM test duration	5.442 ms	1.512 ms	3.60
Average caching effect (CPU and MEM)			2.54

As for this report the operating system is tested on our standard platform, the same results may be used.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.2 Clock tests (CLK)

The clock test measures the time the operating system needs to handle its clock interrupt. On the tested platform, the clock tick interrupt is set on the highest hardware interrupt level, interrupting any other thread or interrupt handler.

The priority of the clock interrupt depends on the board used and the hardware configuration of these.

In Windows CE the clock interrupt is a very small piece of code that only activates the kernel scheduler when a time-out is detected. More details about the working of the clock interrupt in Windows CE can be found in the architectural review rapport on CE.

5.2.1 Operating system clock setting (CLK-B-CFG)

This tests the period of the clock tick interrupt in the operating system. The test shows the default clock timing as set by the BSP and or the kernel.

5.2.1.1 Test results

Test	result
Test succeeded	YES
Tested clock period	995.10 µs
Clock period adaptable	No

5.2.2 Clock tick processing duration (CLK-P-DUR)

This tests the clock tick processing duration in the kernel. The test results are extremely important, as the clock interrupt will disturb all other measurements done. Windows CE passes this test very well: the clock interrupt takes only 2.9 µs.

These results are as expected taking into account the architectural design of the system clock. The clock timer interrupt will only activate the scheduler when something is timed-out: then a rescheduling will occur and some waiting thread can activate.

When using the kernel debugger, it can be clearly seen that on regular times a rescheduling occurs: a second line is located about 14µs above the clock duration.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.2.2.1 Test results on Enterprise Terminal /Release Build

Test	result
CLOCK_LOOP_COUNTER	10000
Normal busy loop time	248.8 µs
Busy loop time with clock interrupt	251.7 µs
Clock interrupt duration	2.9 µs

5.2.2.2 Test results on Tiny Kernel /Release Build

Test	result
CLOCK_LOOP_COUNTER	10000
Normal busy loop time	248.8 µs
Busy loop time with clock interrupt	251.7 µs
Clock interrupt duration	2.9 µs

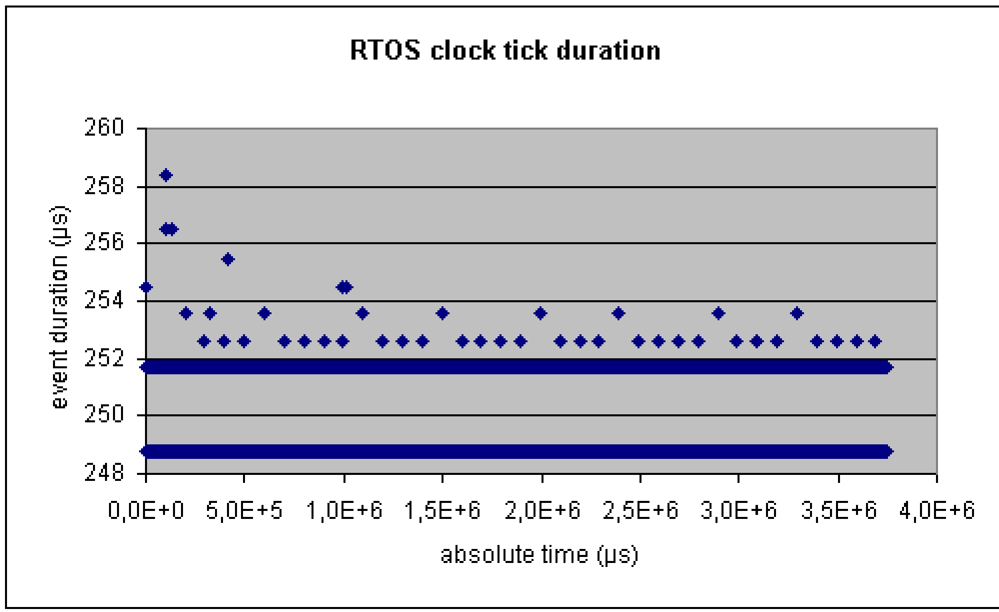
5.2.2.3 Test results on Enterprise Terminal /Debug build /Kernel debugger used

Test	result
CLOCK_LOOP_COUNTER	10000
Normal busy loop time	796.09 µs
Busy loop time with clock interrupt	799.91 µs
Clock interrupt duration	3.82 µs

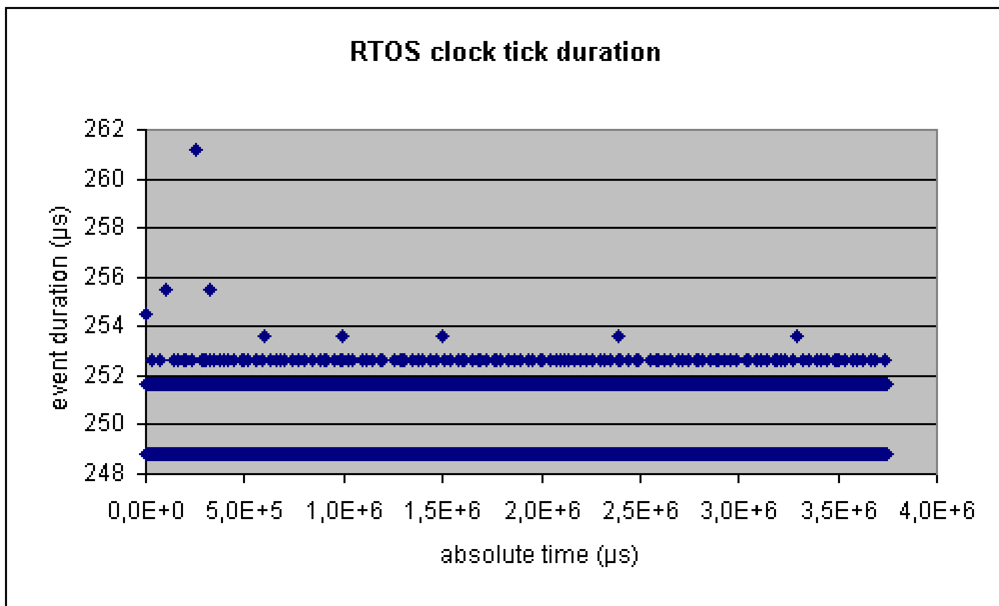
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.2.2.4 Diagram

Enterprise Terminal /Release Build

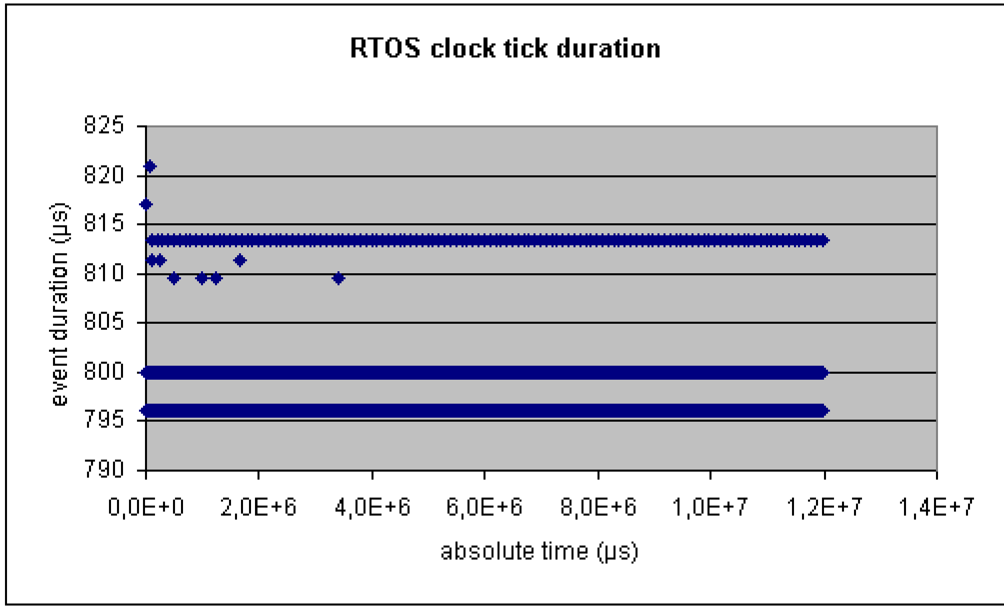


Tiny Kernel /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Enterprise Terminal Debug build /Kernel debugger used



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.3 Thread tests (THR)

These tests are used to measure the performance of the scheduler.

The thread test behaved well and stable. There is one shortcoming that could easily be improved: when you create a thread you can't give a priority with it. So if you want to set the priority from the creating thread you need to perform following steps:

- create the thread in suspended state
- set the priority and the thread quantum
- start the thread.

This mechanism for starting a thread (in suspended state) was used for all the tests in this report.

5.3.1 Thread creation behaviour (THR-B-NEW)

This will test the behavior of creating threads. Does the operating system behave as it should for a real-time operating system?

In our opinion, a thread created with a higher priority than the creating thread should activate immediately. A thread created with a lower priority than the creating thread should surely not activate until any higher priority threads have finished their job. By default Threads are created at priority of 251 and there are 256 priorities. The only thing missing in the CreateThread call is a parameter for the priority, which would be really useful.

This lacking feature can cause problems if you create threads from a thread running at a lower priority than the default one, therefore it's good practice to start a thread in suspended mode first.

5.3.1.1 Test results

Test	result
Test succeeded	YES (only if creating thread in suspended state, setting priority and then starting the thread)
Lower priority not activated?	OK
Same priority at tail?	OK
Yielding works?	YES
Higher priority activated?	OK

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.3.2 Round robin behaviour (THR-B-RR)

This test checks if the scheduler uses a fair round robin mechanism when threads are having the same priority and all are in the ready-to-run state!

In Windows CE, there is round-robin scheduling between processes using the same priority. The test isn't much dependent on the number of threads used in the test. We did the test with 2, 10 and 128 threads.

We detected a problem with the first time scheduling of each thread. By default the round-robin time tick (called thread quantum by Microsoft) is set to 100ms. The OEM may overload this default value and the application programmer can set this value for each thread independently.

So when needed you can change this with the CeSetThreadQuantum() system call. However this only works the second time the thread is activated by the kernel, even if the quantum is set before the thread starts (thread created in suspended state). This can have side-effects when you start-up your real-time application. It seems logical that changing the thread quantum has only an affect on the next scheduling, as the scheduler will calculate the time-out event time. But it isn't logical that it is not taken into account if it set before the thread is even started!

Remark that the thread quantum can also be set to zero: in this case the thread runs until termination (at its priority of course).

5.3.2.1 Test Results

Test	result
Test succeeded	YES
Time slice following this test	1 ms (Default 100ms)

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.3.3 Thread switch latency between same priority threads (THR-P-SLS)

This test measures the time to switch between threads of the same priority. Therefore the voluntary yield processor to other thread system call is used (in Windows CE this is done by the System Call Sleep(0)). The time between the entry of the yield call (thread going to sleep) and the exit of the yield call (activated thread) is measured.

This test behaved differently than other RTOS tested, but it is a valid behavior in a real-time environment. This is due to some design choices in the kernel which are explained in the architectural review of CE (see ???). In Windows CE, no difference is made between being pre-empted by:

– any higher priority thread (for instance caused by an interrupt event),
– or by itself by lowering its priority below any other thread in the "ready-to-run" state.

The first case is correct and even mandatory behavior in a real-time system. A pre-empted thread should not be put back at the tail of its priority FIFO!

As a consequence it is plausible to use the same mechanism in the second case.

In short: this test creates a number of threads (N) with a decreasing ID (N-1, N-2, …, 1, 0). Each created thread lowers its priority below the creating thread when started and all of them set themselves to the same priority. As a result a queue of suspended threads is waiting to be activated.

Most RTOS will put the pre-empted thread by lowering its priority at the tail of the FIFO. CE puts them at the front.

You can see clearly that the switch latency time rises a bit when the number of threads involved in the application becomes larger. This is a normal as caching effects will occur, but it stays within limits (remark that caching can have an influence of a factor 2 to 3). This can be seen more in detail when zooming in on the diagram (shown for the 128 thread test in tiny configuration). There are no dependencies between the number of suspended threads to be activated and the switch latency.

It is a good exercise to compare the results with our previous evaluation of CE: the 4.0 version. Clearly Microsoft did a good job! The thread switch latency has become more stable. The worst case switch time which happened when the thread first started has disappeared.

The clock tick overhead is clearly seen in the test results.

The spikes on the enterprise terminal (of about 14µs) were caused by a network interrupt.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

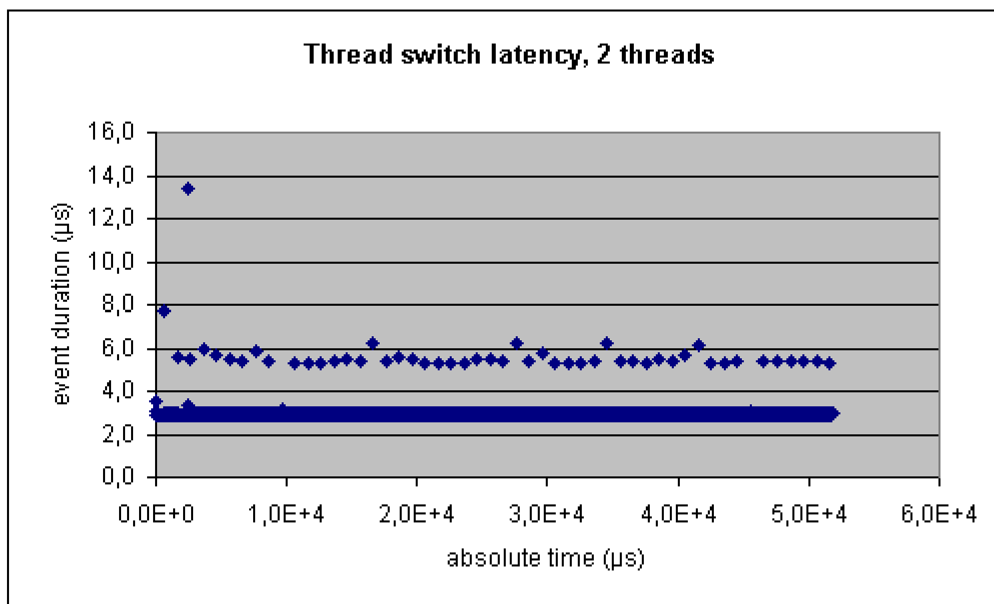
5.3.3.1 Test results on Enterprise Terminal /Release Build

Test	result
Test succeeded	YES

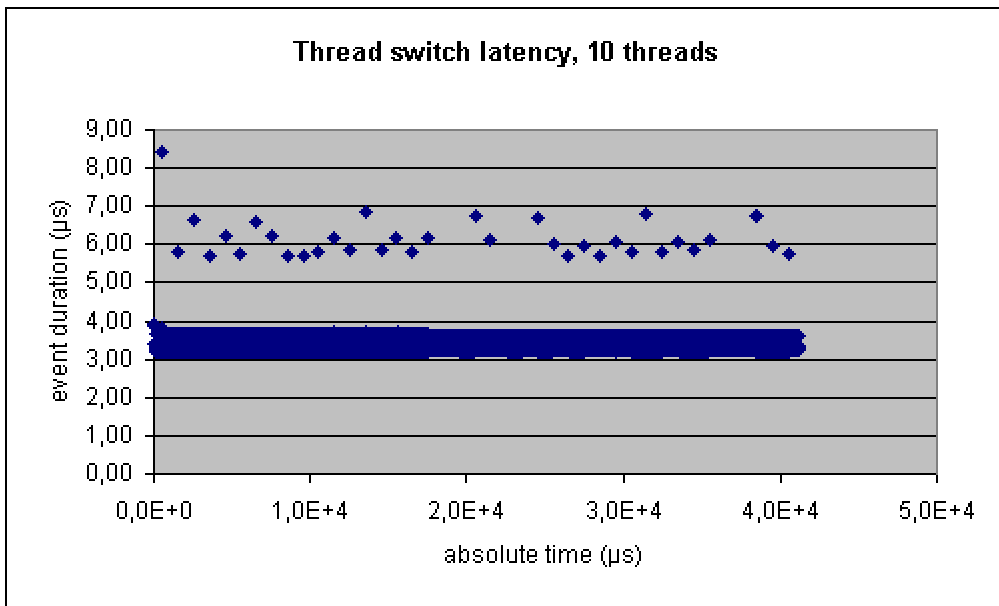
Test	Sample qty	Avg	Max	Min
Thread switch latency, 2 threads	16383	2.9 µs	13.4 µs	2.8 µs
Thread switch latency, 10 threads	10919	3.3 µs	8.4 µs	3.19 µs
Thread switch latency, 128 threads	10880	5.0 µs	19.0 µs	4.3 µs

Diagrams

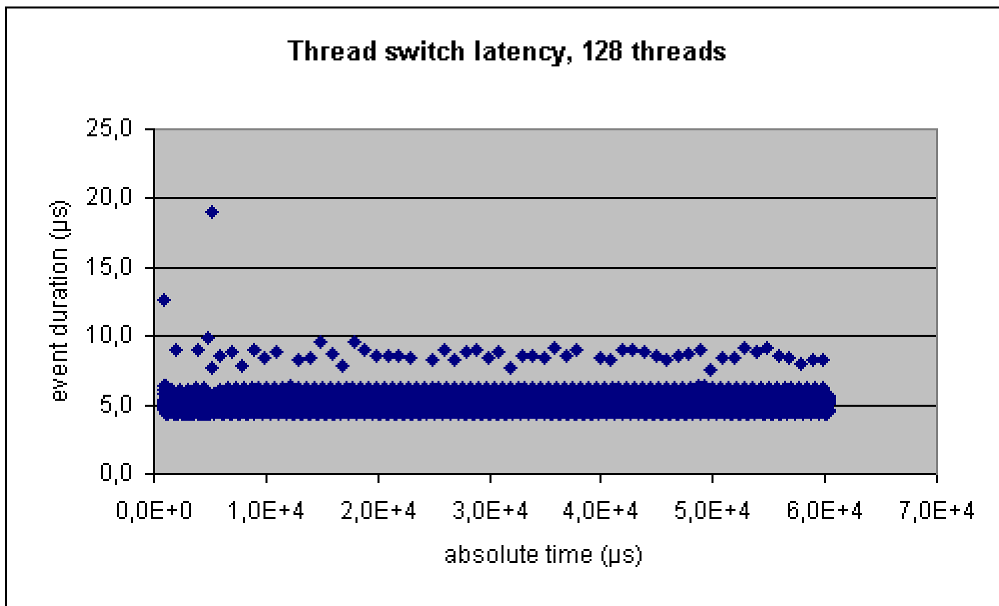
Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

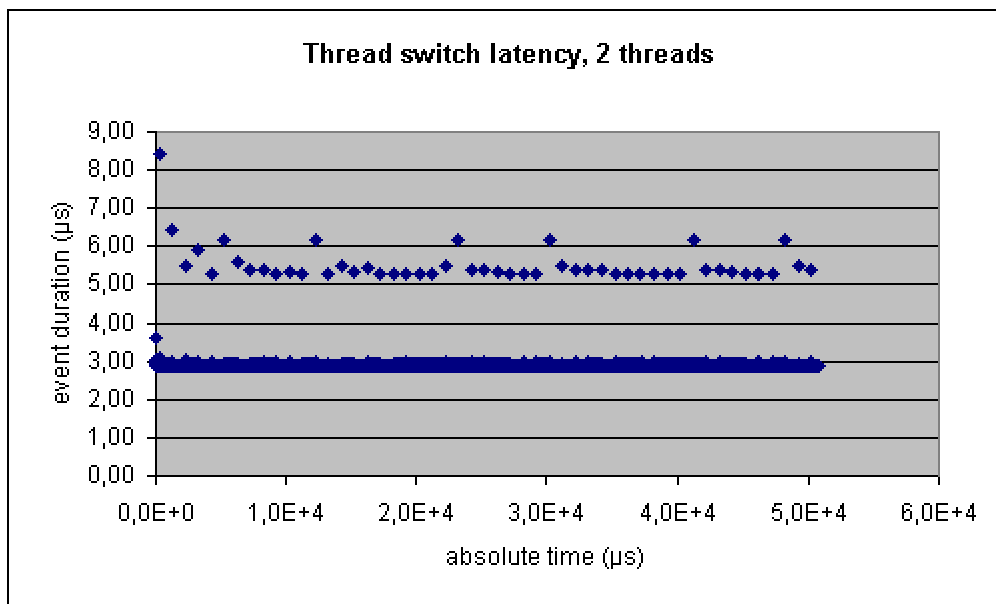
5.3.3.2 Test Results on Tiny Kernel /Release Build

Test	result
Test succeeded	NO: LIFO scheduling used!

Test	Sample qty	Avg	Max	Min
Thread switch latency, 2 threads	16383	2.8 µs	8.4 µs	2.8 µs
Thread switch latency, 10 threads	10919	3.49 µs	11.2 µs	3.15 µs
Thread switch latency, 128 threads	10880	4.9 µs	9.2 µs	3.5 µs

Diagrams

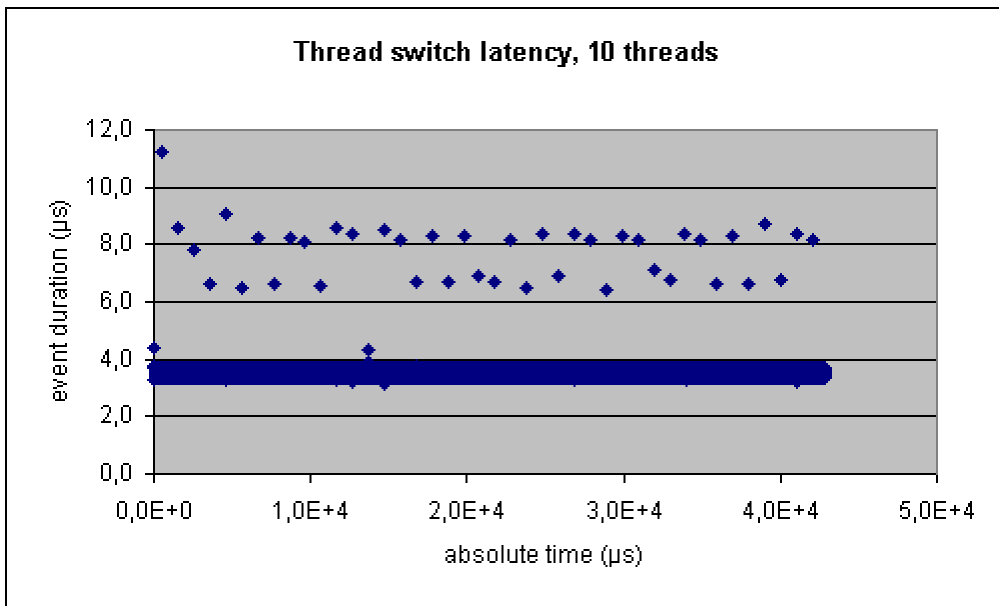
Tiny Kernel /Release Build

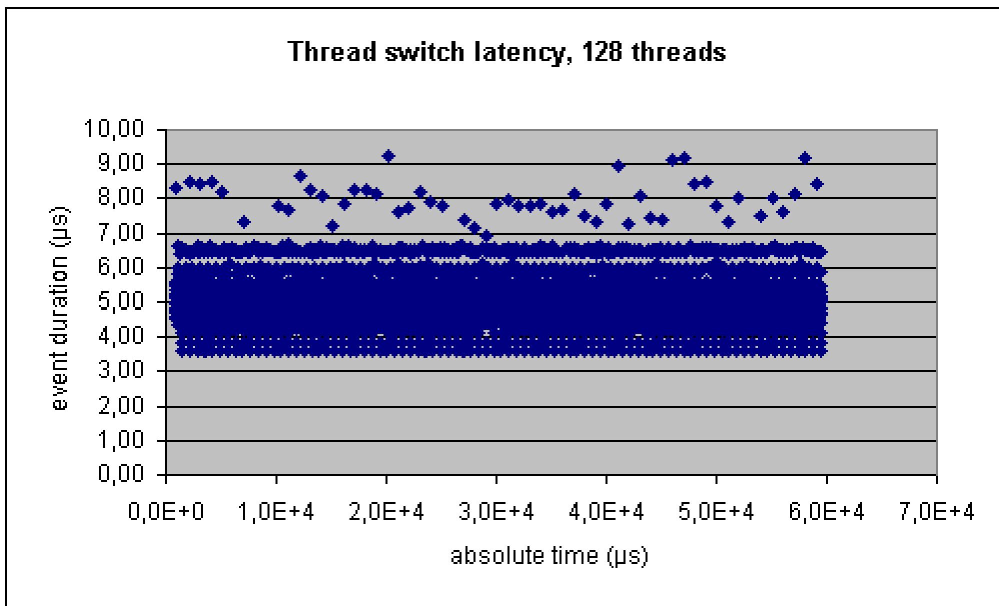


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

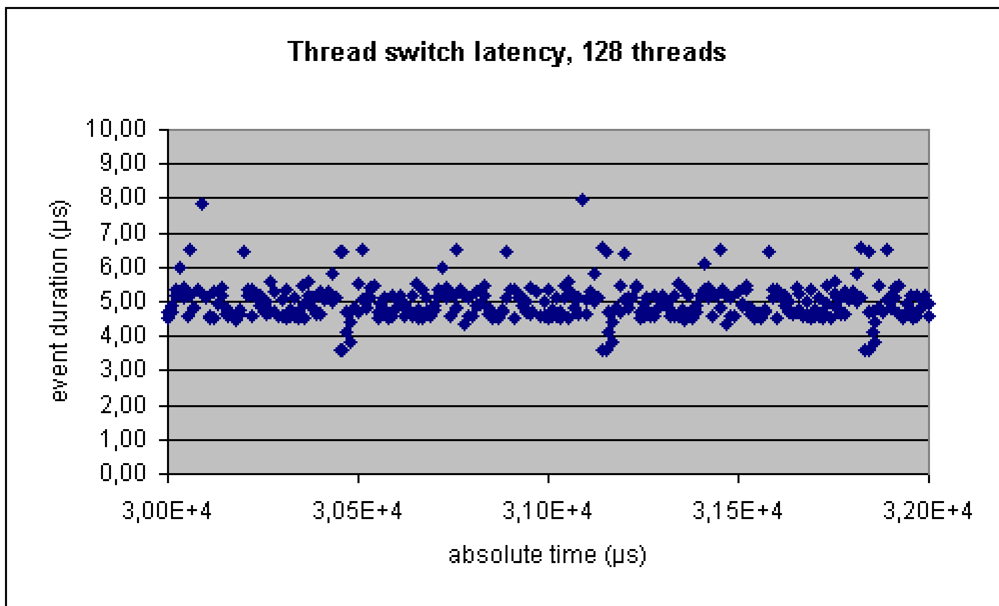
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.





http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



Detailed extract: dependency on number of threads

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

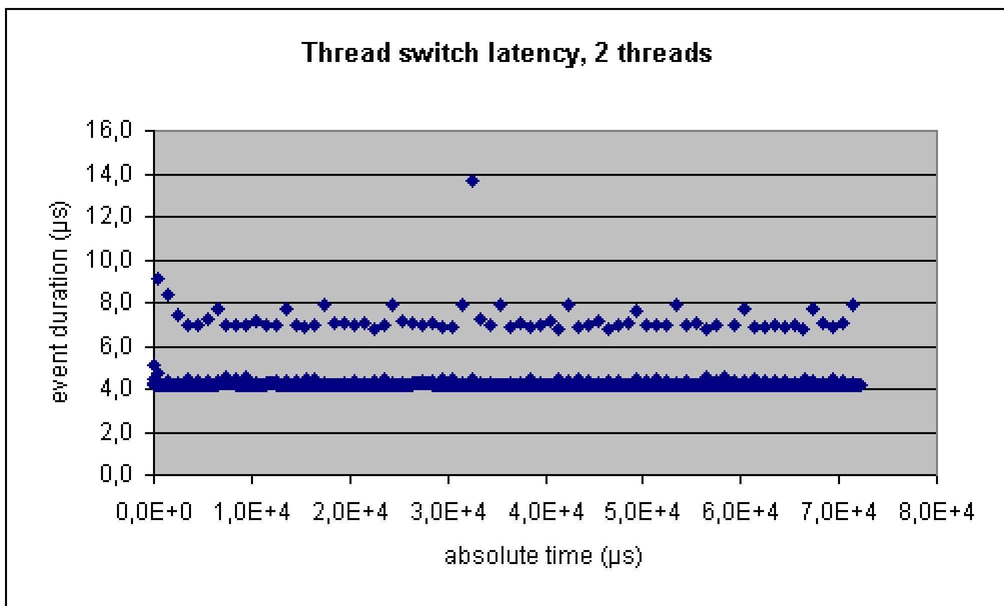
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.3.3.3 Test results on Enterprise Terminal Debug build /Kernel debugger used

Test	result
Test succeeded	NO: LIFO scheduling used!

Test	Sample qty	Avg	Max	Min
Thread switch latency, 2 threads	16383	4.2 µs	13.6 µs	4.1 µs

Enterprise Terminal Debug build /Kernel debugger used



Remark that for the OS to be predictable, the number of threads in the ready queue may not have an impact on the switch latency measured. In this case there is indeed no impact.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.3.4 Thread creation and deletion time (THR-P-NEW)

This tests the time to create a thread and the time to delete a thread in different scenarios:

- Scenario “never run”: The created thread has a lower priority than the creating thread and is deleted before it had any chance to run: in this test no thread switch occurs.
- Scenario “run and terminate”: The created thread has a higher priority than the creating thread and activates. The created thread immediately terminates itself (thread does nothing).
- Scenario “run and pre-empt”: The same scenario as the second case (above), but the created thread does not terminate (it lowers its priority when it is activated).

In the scenarios where the thread actually runs, the creation time is the duration from the system call creating the thread to the time when the created thread activates. For the “never run” scenario the creation time is the duration of the system call. The deletion time is the time of the system call duration that terminates a thread.

Remark that in a well designed real-time application, threads are created once at startup and they will live until the end of the application. Just like any resource allocation and freeing, no one can guarantee when and if resources will be available within a certain timeframe. Therefore creation/deletion tests are less important as usage of system objects in a real-time environment.

However we do these test to try to detect anomalies in the OS.

In this case we clearly detect a spike now and then where the creation/deletion takes about an extra millisecond to complete.

5.3.4.1 Test results on Enterprise Terminal /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg (µs)	Max (µs)	Min (µs)
Thread creation, never run	7500	91.8	1190.0	89.2
Thread deletion, never run	7500	388.1	1493.0	356.0
Thread creation, run and terminate	7500	229.5	1343.0	225.8
Thread deletion, run and terminate	7500	5.5	11.2	5.2
Thread creation, run and block	7500	235.7	1263.0	232.1
Thread deletion, run and block	7500	383.0	1470.0	316.7

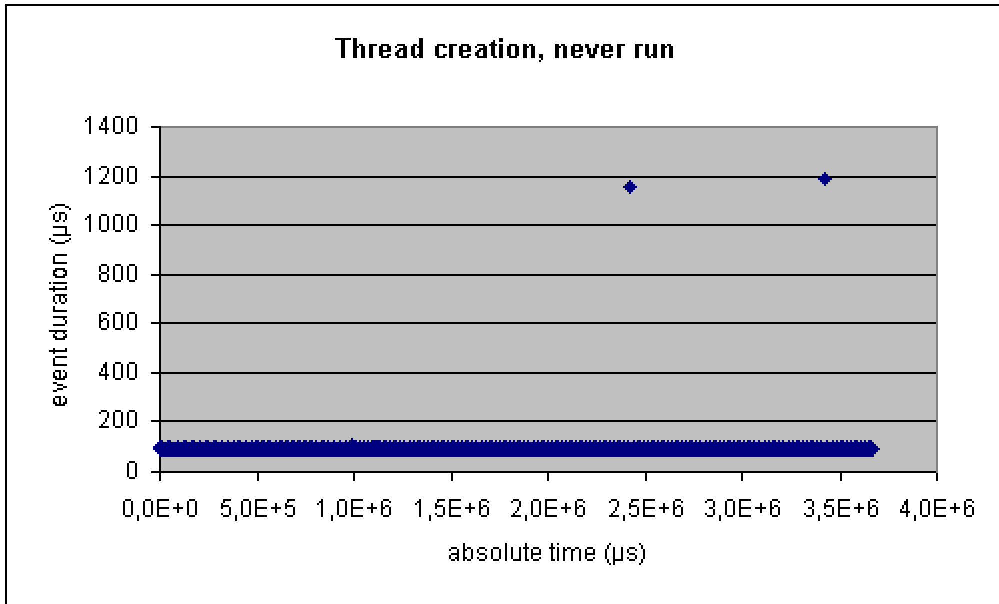
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

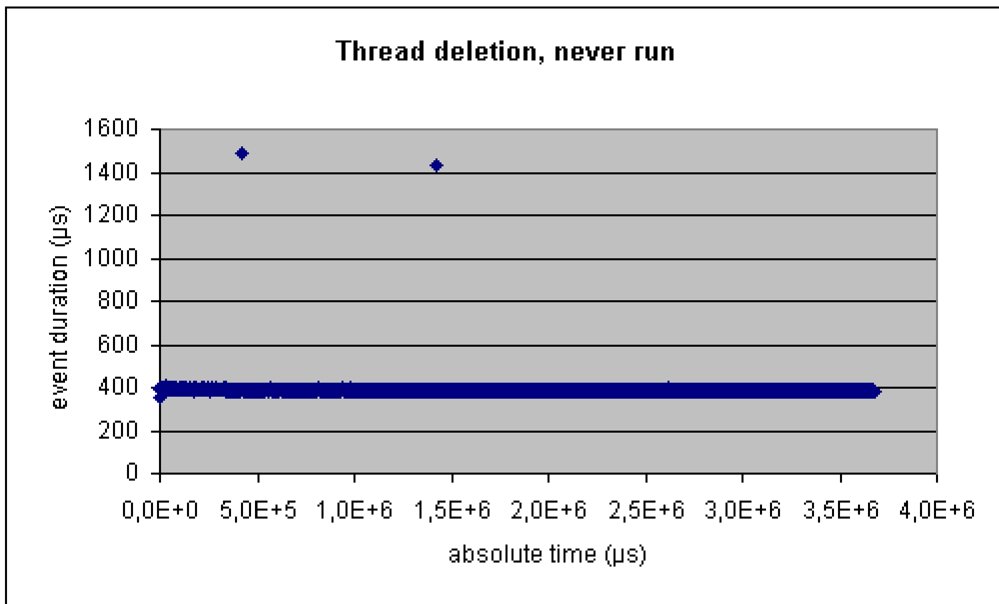
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagrams:

– Thread creation time in the "NER" scenario (duration of system call). (**Enterprise Terminal** /Release Build)



– Thread deletion time in the "NER" scenario. (**Enterprise Terminal** /Release Build**)**

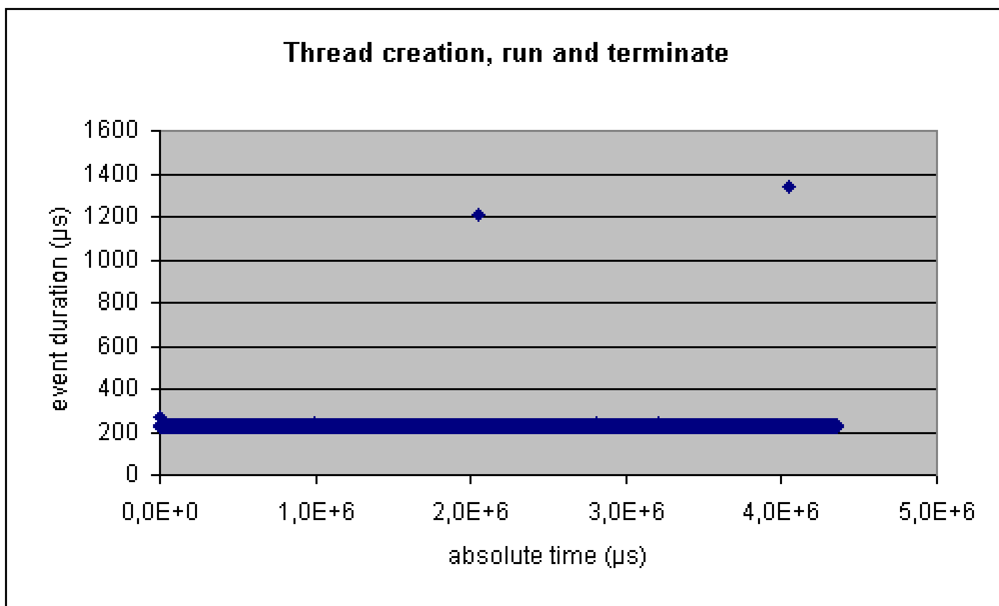


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

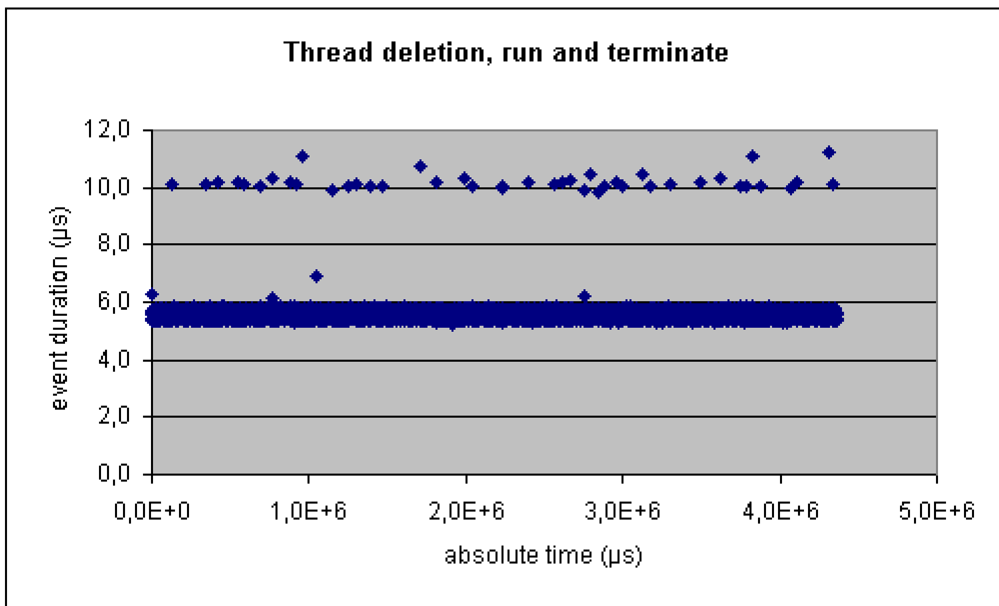
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

– Thread creation time in the "RTE" scenario (duration of system call start to activated thread). (**Enterprise Terminal** /Release Build**)**

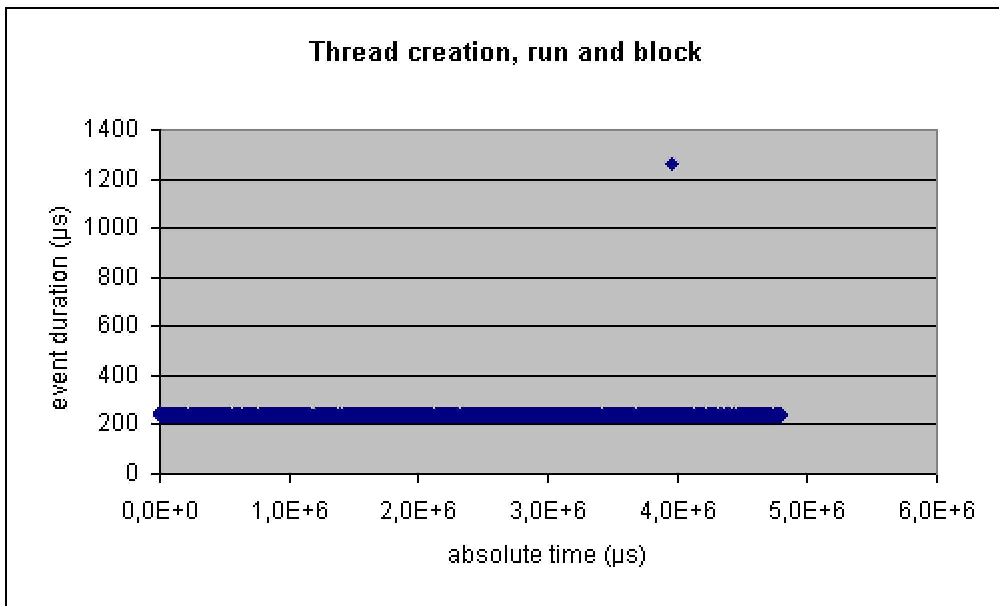


– Thread deletion time in the "RTE" scenario. (**Enterprise Terminal** /Release Build**)**

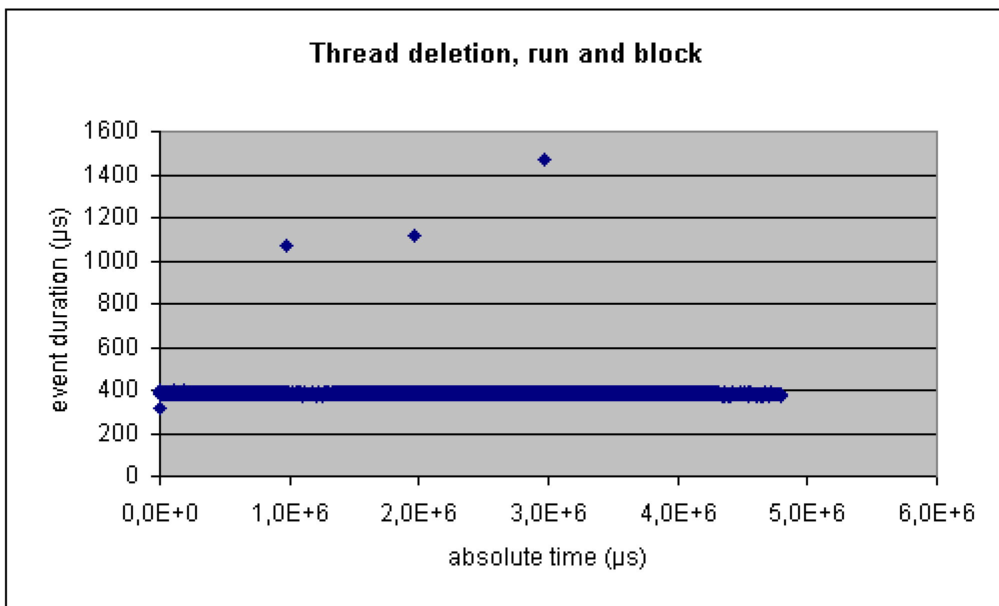


– Thread creation time in the "RNT" scenario (duration of system call start to activated thread). (Enterprise Terminal /Release Build)

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



– Thread deletion time in the “RNT” scenario. (**Enterprise Terminal** /Release Build**)**



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

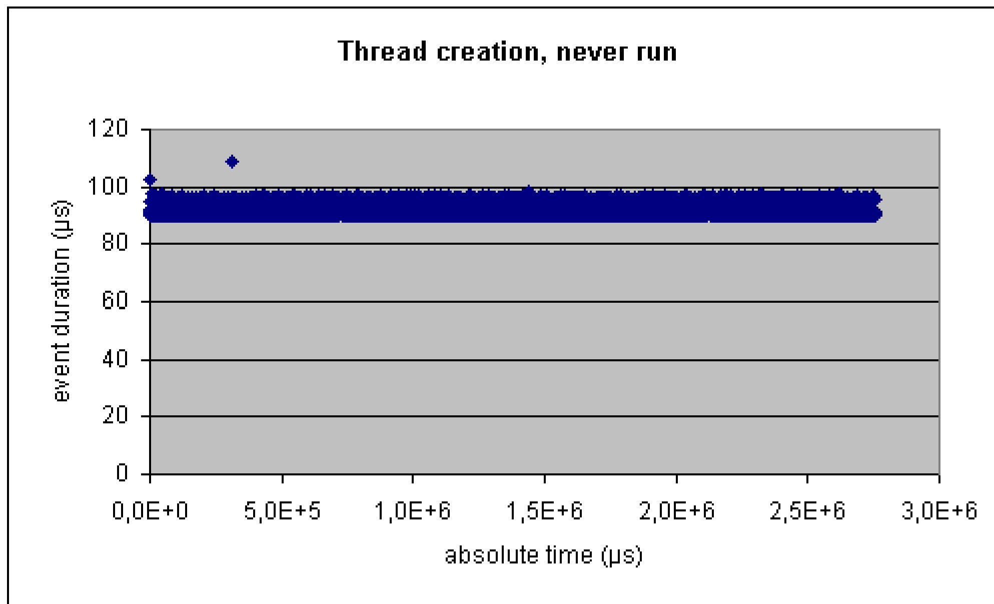
5.3.4.2 Test results on Tiny Kernel /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg (µs)	Max (µs)	Min (µs)
Thread creation, never run	7500	88.5	108.6	86.1
Thread deletion, never run	7500	261.1	1282.0	211.0
Thread creation, run and terminate	7500	113.0	1156.0	111.0
Thread deletion, run and terminate	7500	4.9	10.0	4.6
Thread creation, run and block	7500	116.3	126.8	114.1
Thread deletion, run and block	7500	256.8	1194.0	194.7

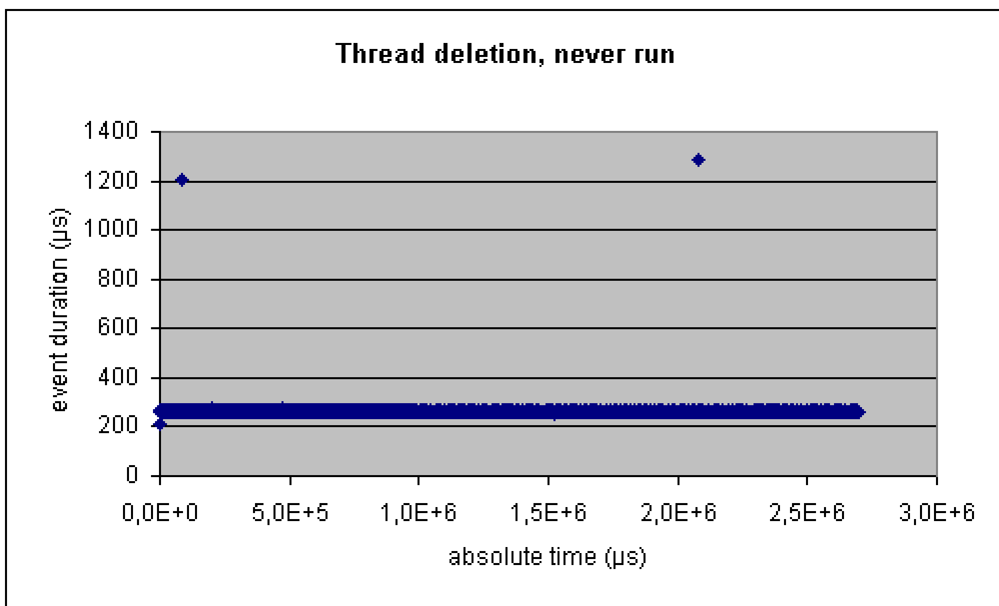
Diagrams:

– Thread creation time in the “NER” scenario (duration of system call). (**Tiny Kernel** /Release Build)

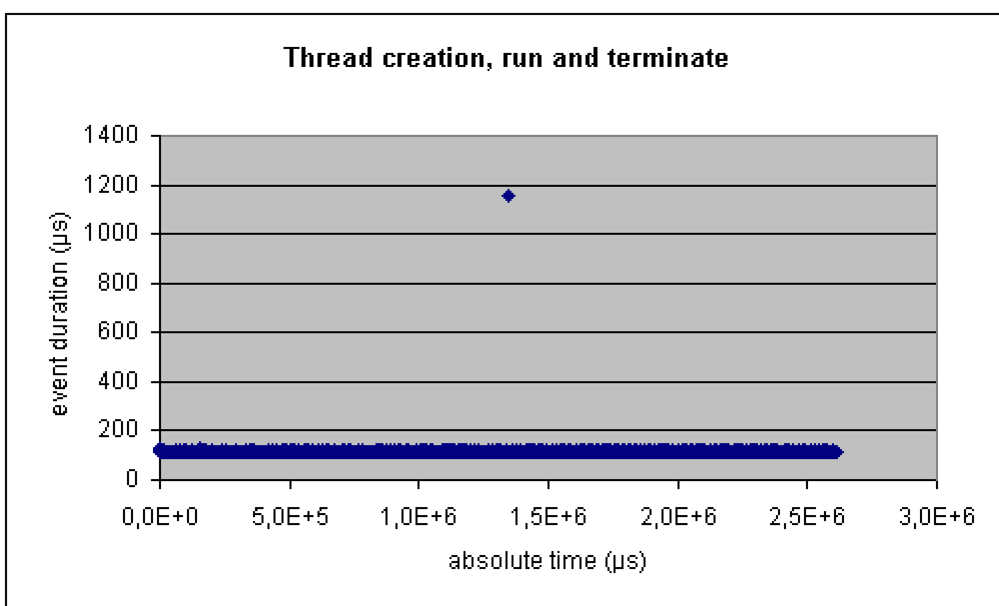


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

– Thread deletion time in the "NER" scenario. (**Tiny Kernel** /Release Build**)**



– Thread creation time in the "RTE" scenario (duration of system call start to activated thread). (**Tiny Kernel** /Release Build**)**

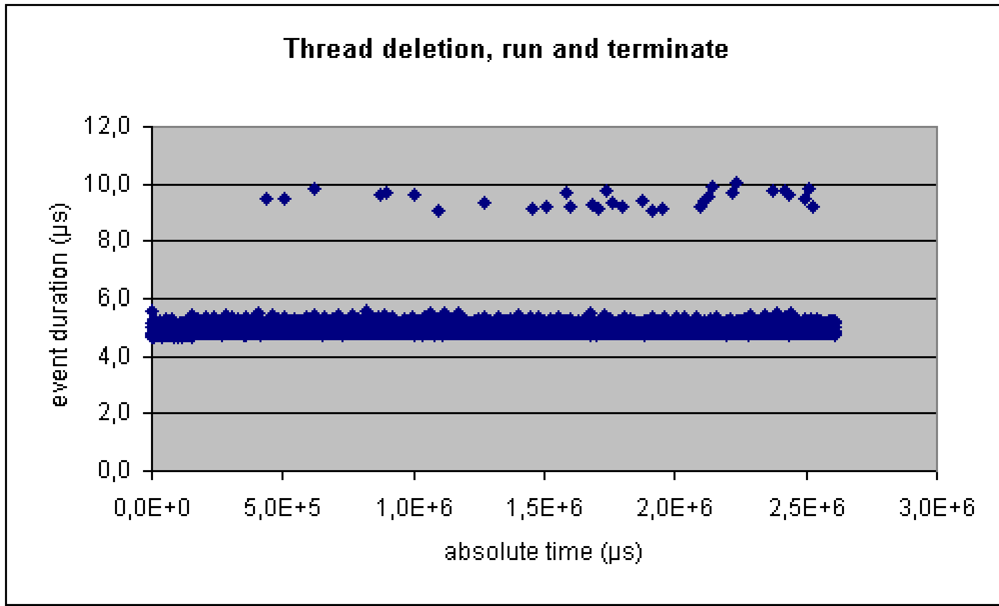


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

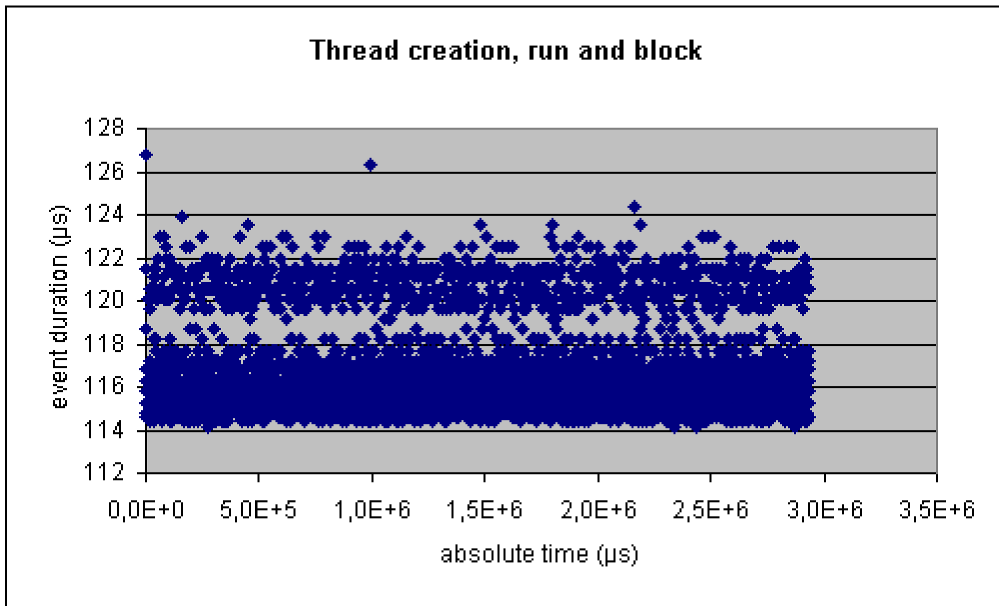
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

– Thread deletion time in the "RTE" scenario. (**Tiny Kernel** /Release Build**)**



– Thread creation time in the "RNT" scenario (duration of system call start to activated thread). (**Tiny Kernel** /Release Build**)**

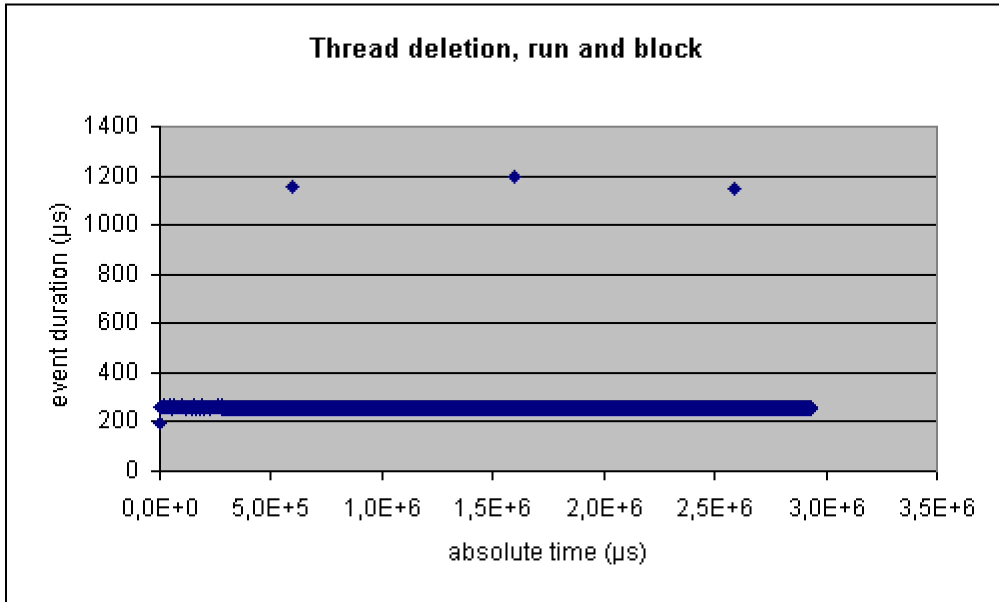


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

– Thread deletion time in the "RNT" scenario. (**Tiny Kernel** /Release Build**)**



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

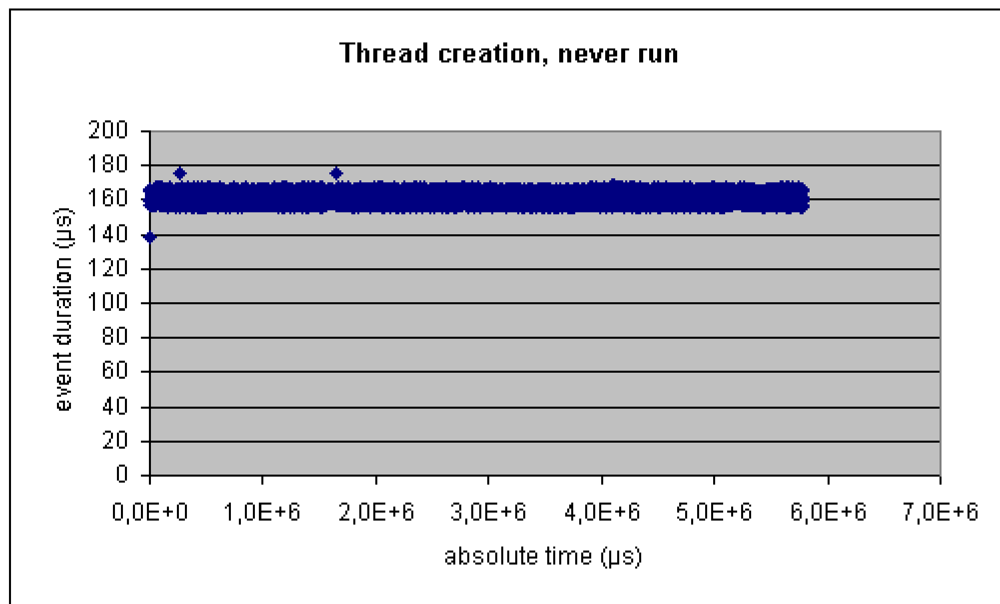
5.3.4.3 Test results on Enterprise Terminal \DEBUG

Test	result
Test succeeded	YES

Test	Sample qty	Avg (µs)	Max (µs)	Min (µs)
Thread creation, never run	7500	159.7	176.1	138.8
Thread deletion, never run	7500	586.6	1573.0	528.2
Thread creation, run and terminate	7500	358.3	1408.0	354.0
Thread deletion, run and terminate	7500	10.7	20.8	10.4
Thread creation, run and block	7500	369.3	1451.0	356.0
Thread deletion, run and block	7500	580.5	1646.0	480.3

Diagrams: We won't show all the diagrams here because the debug results are not very different than the release configurations.

– Thread creation time in the "NER" scenario (duration of system call). (**Enterprise Terminal** Debug build /Kernel debugger used)

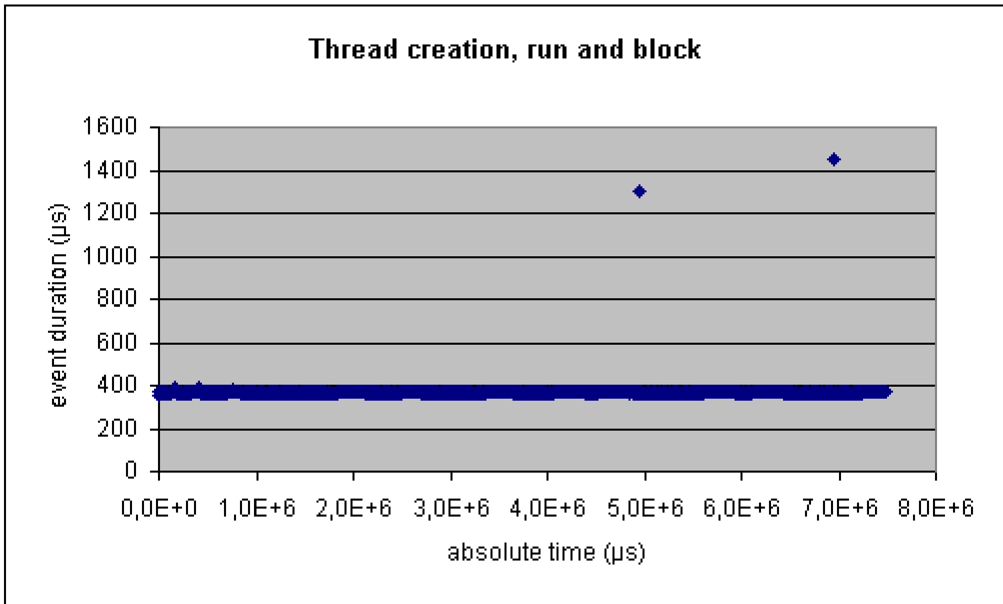


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

– Thread creation time in the "RNT" scenario (duration of system call start to activated thread). (**Enterprise Terminal** Debug build /Kernel debugger used**)**



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.4 Semaphore tests (SEM)

Here the performance and the behavior of the counting semaphore are tested. The counting semaphore is a system object that protects for simultaneous accesses to some device.

We use the well known acronyms from Dijkstra, the Dutch mathematician who invented the semaphore:

- P() : “Probeer”, the dutch word for “Try”, thus trying to take the semaphore
- V(): “Vrij”, the dutch word for “Free”, thus releasing the semaphore.

It is important to remark that in Windows CE the semaphore uses a priority inheritance to avoid priority inversion. This is exceptional: most RTOS use only priority inheritance with the mutex object.

This is a very nice feature for making real-time systems reliable. Some will argue that this makes the OS slower. This is true indeed, but only in the average case: in real-time systems the worst case behavior is far more important, this is why priority inheritance is an important feature!

5.4.1 Semaphore locking test mechanism (SEM-B-LCK)

This will test if the counting semaphore locking mechanism works as it is expected. The P() call should block only when the count is zero. The V() call should increment the semaphore counter. In the case the semaphore counter is zero, the V() call should cause a rescheduling in the kernel: indeed blocked threads may be activated.

The Windows CE semaphore does behave as expected.

5.4.1.1 Test results

Test	result
Test succeeded	YES
Maximum semaphore value?	0x7FFFFFF
Rescheduling on free?	OK

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.4.2 Semaphore releasing mechanism (SEM-B-REL)

This test verifies that the highest priority thread being blocked on a semaphore will be released by the release operation. This should be independent of the order of the acquisitions taking place.

Windows CE passed this test without problems.

5.4.2.1 Test results

Test	result
Test succeeded	YES

5.4.3 Time needed to create and delete a semaphore (SEM-P-NEW)

This will test the time needed to create a semaphore and the time to delete it. The deletion time is checked in two cases:

- Where the semaphore is used between the creation and deletion.
- Where the semaphore is not used between the creation and deletion.

For a good real-time operating system it is expected that there is no difference between the two scenarios. If a difference is detected, then this probably means that the operating system handles some initializations on the semaphore on its first use (making the first use slower). In a good real-time application design, all operating system objects will be allocated and initialized at start of the application and never release until the application terminates. The application developer expects however that these objects will be predictable when they are used, even if it is the first time.

In Windows CE the difference between the two scenarios are minimal which is good.

Remark that sometimes the clock interrupt occurs during an interval measurement, the 2.9 µs clock tick spike is found back in these graphics.

The first time to create/delete the semaphore takes longer due to caching mechanisms.

5.4.3.1 Test results on Enterprise Terminal /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore creation time, used	7500	4.3 µs	13.1 µs	4.0 µs
Semaphore deletion time, used	7500	4.8 µs	9.4 µs	4.4 µs
Semaphore creation time, never used	7500	4.0 µs	11.2 µs	3.9 µs
Semaphore deletion time, never used	7500	4.1 µs	10.7 µs	4.0 µs

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagrams

Enterprise Terminal /Release Build

Semaphore creation time, used

event duration (µs)

absolute time (µs)

Enterprise Terminal /Release Build

Semaphore deletion time, used

event duration (µs)

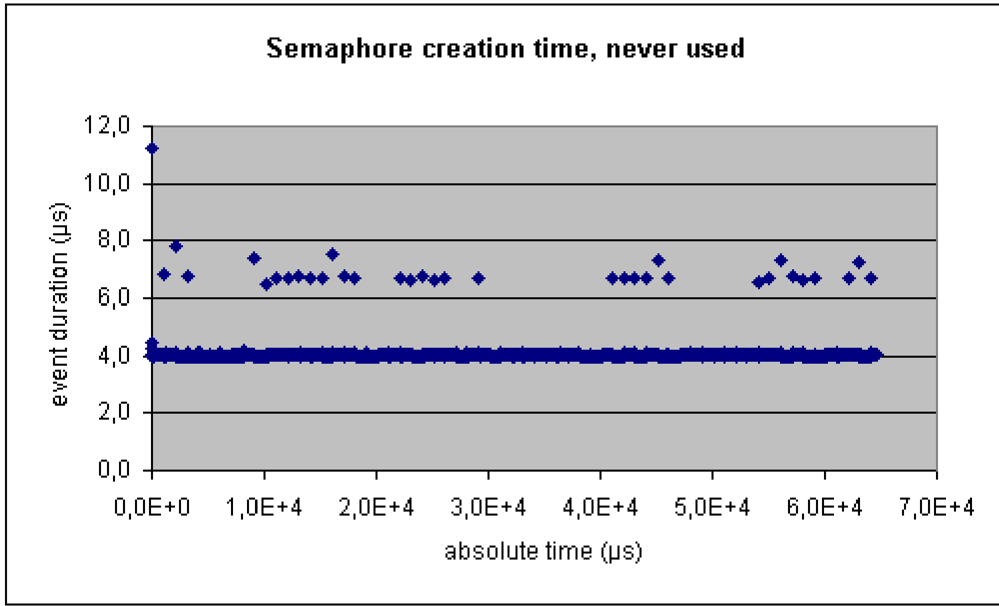
absolute time (µs)

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

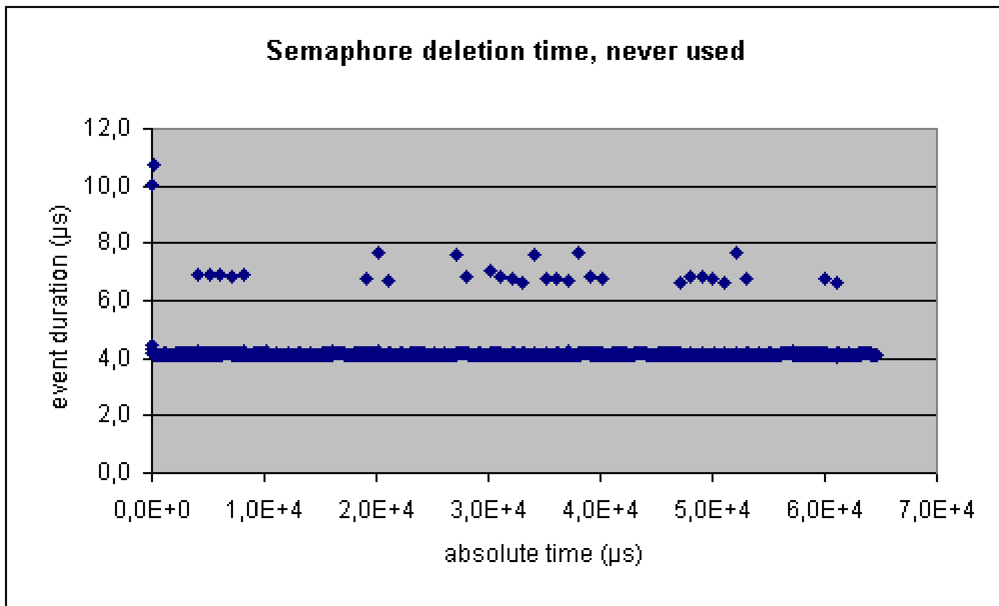
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Enterprise Terminal /Release Build



Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

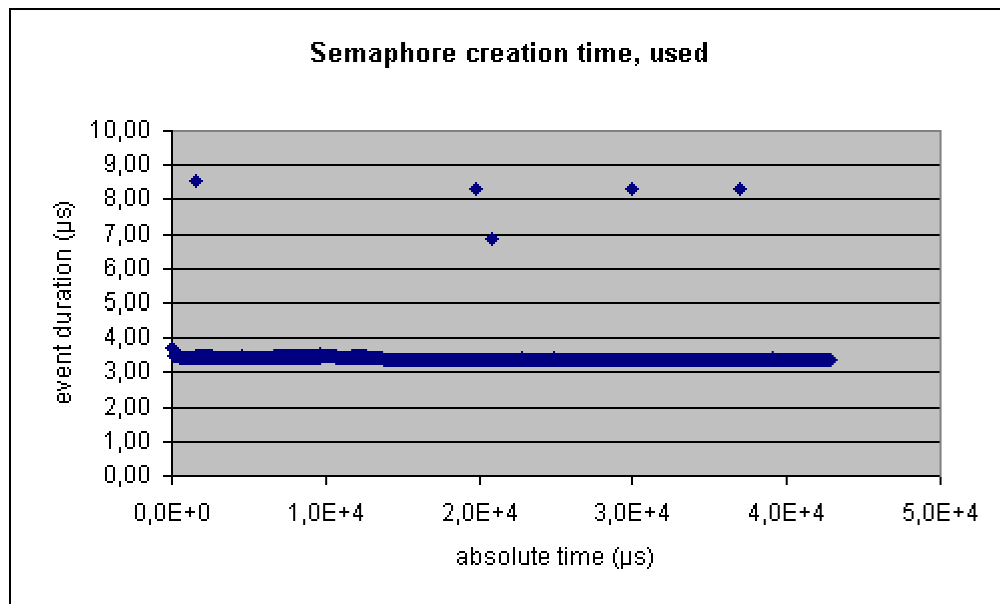
5.4.3.2 Test results on Tiny Kernel /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore creation time, used	7500	3.4 µs	8.5 µs	3.3 µs
Semaphore deletion time, used	7500	4.7 µs	9.4 µs	4.5 µs
Semaphore creation time, never used	7500	4.3 µs	9.8 µs	4.2 µs
Semaphore deletion time, never used	7500	4.2 µs	10.1 µs	4.0 µs

Diagrams:

Tiny Kernel /Release Build

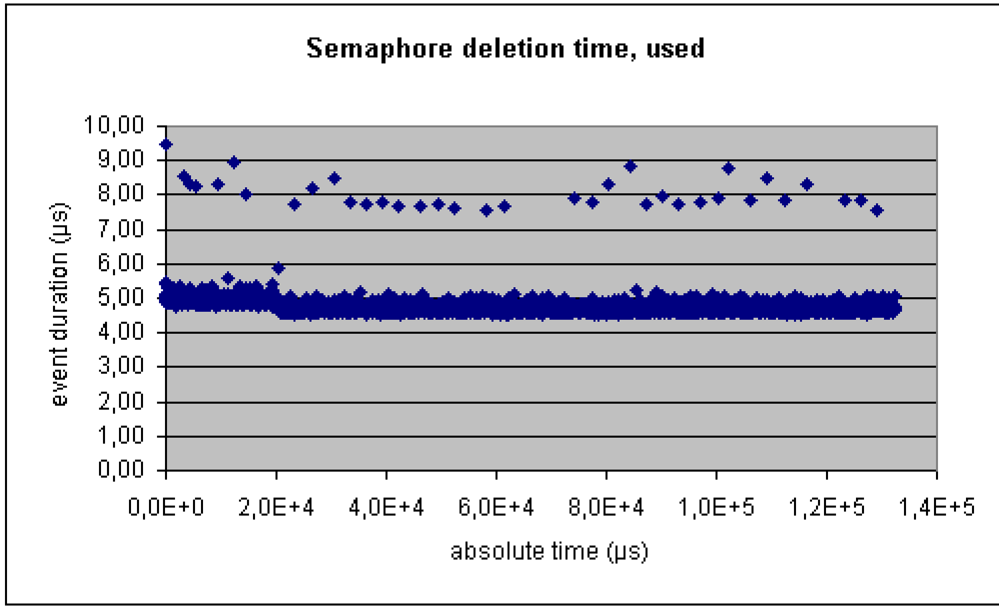


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

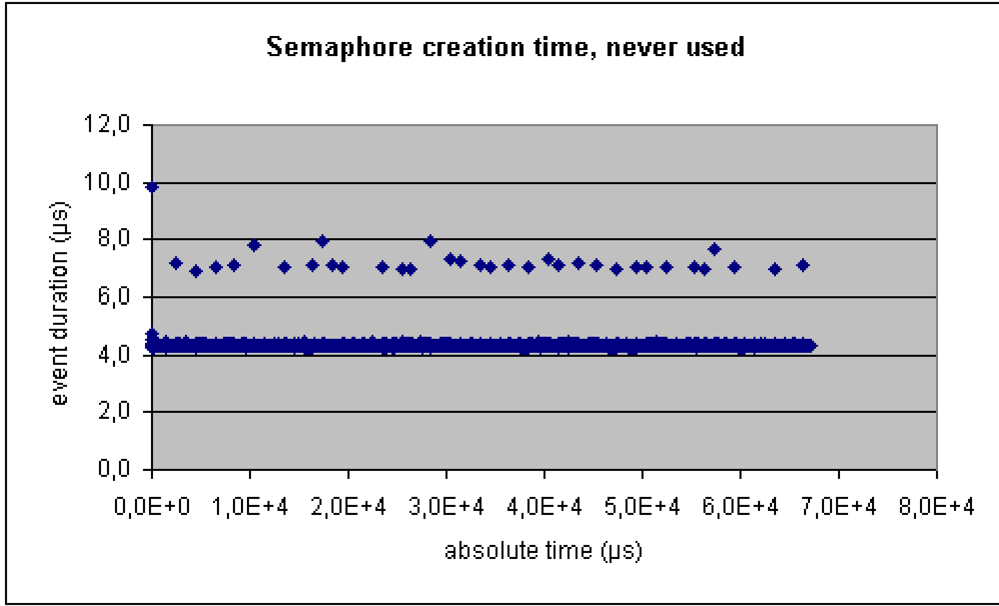
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Tiny Kernel /Release Build

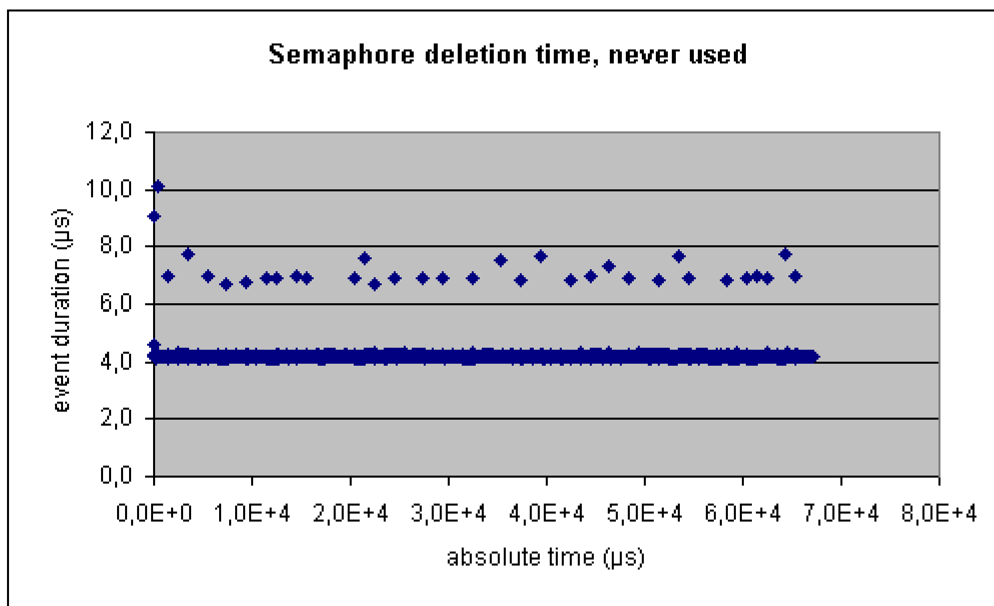


Tiny Kernel /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Tiny Kernel /Release Build



5.4.3.3 Test Results on Enterprise Terminal Debug build /Kernel debugger used

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore creation time, used	7500	8.7 µs	19.9 µs	8.1 µs
Semaphore deletion time, used	7500	12.5 µs	26.8 µs	11.6 µs
Semaphore creation time, never used	7500	7.5 µs	23.2 µs	7.3 µs
Semaphore deletion time, never used	7500	9.6 µs	21.6 µs	9.4 µs

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.4.4 Test acquire-release timings: no-contention case (SEM-P-ARN)

This tests the acquisition and release time in the no-contention case. As in this test case the semaphore does not block nor cause any rescheduling (thread switch), the duration of the system call should be very short.

In fact, the OS will only need to increase or decrease the semaphore counter in an atomic way. However as a semaphore can be used between processes, the semaphore data is probably located in the kernel. Therefore a system call is needed to the kernel which takes more time than just incrementing/decrementing the counter.

Again the 2.9 µs spike caused by the clock interrupt can be seen on the diagrams.

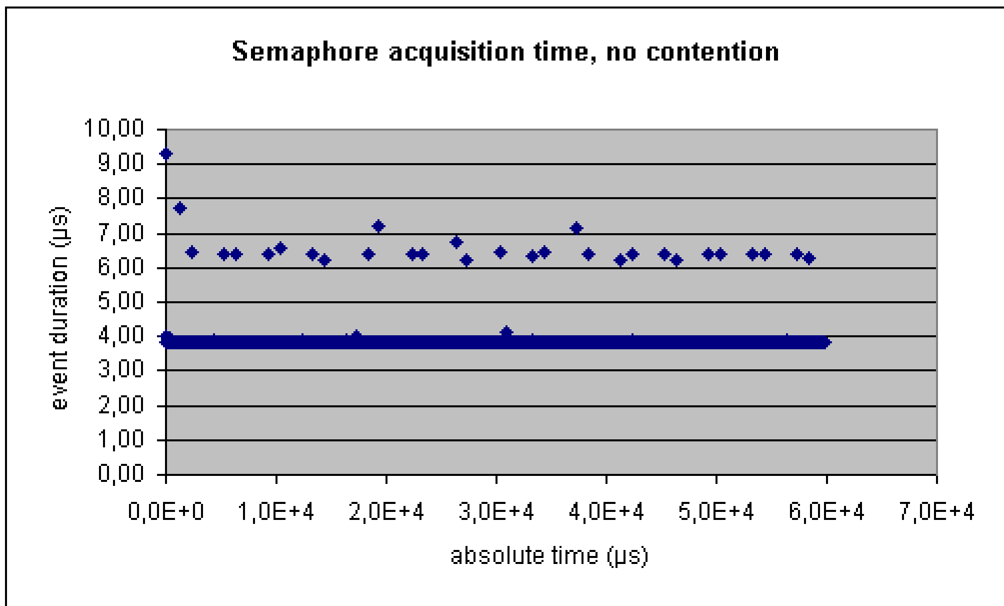
5.4.4.1 Test results on Enterprise Terminal /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, no-contention	7500	3.8 µs	9.2 µs	3.8 µs
Semaphore release time, no-contention	7500	3.7 µs	16.0 µs	3.6 µs

Diagrams:

Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Enterprise Terminal /Release Build

Semaphore release time, no contention

event duration (µs)

absolute time (µs)

5.4.4.2 Test results on Tiny Kernel /Release Build

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, no-contention	7500	3.8 µs	8.1 µs	3.8 µs
Semaphore release time, no-contention	7500	3.7 µs	10.2 µs	3.7 µs

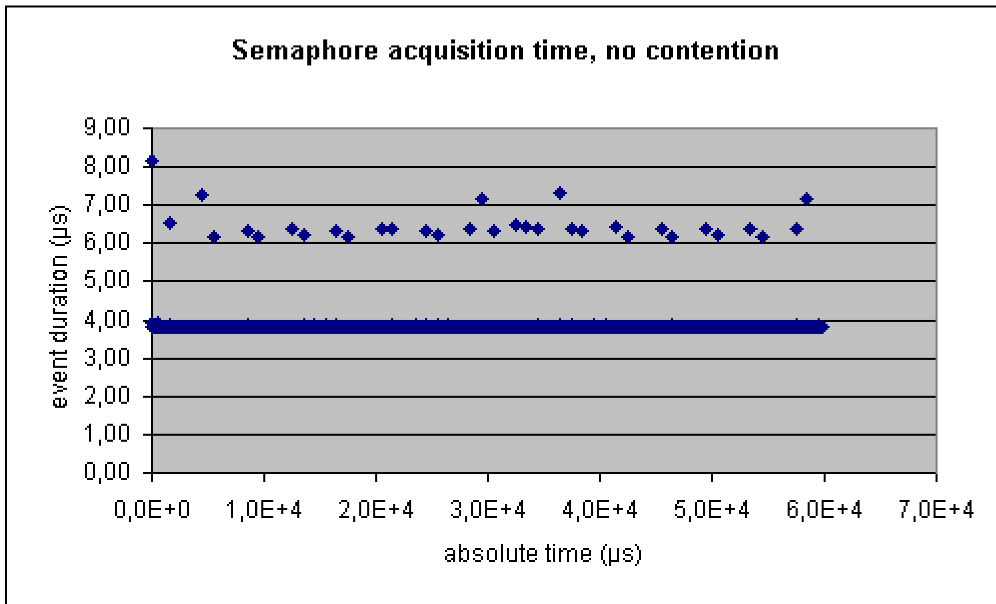
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

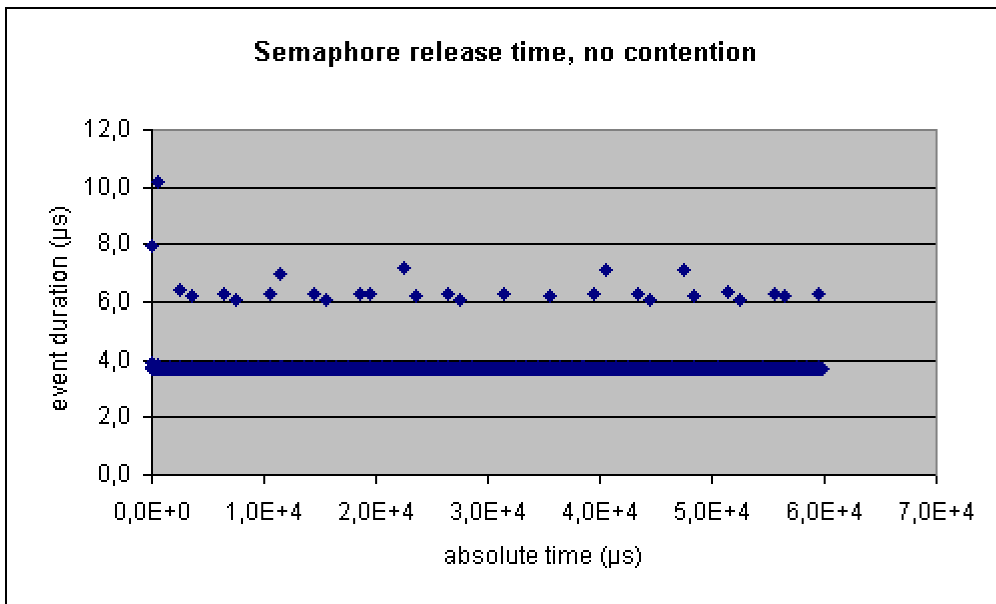
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagrams

Tiny Kernel /Release Build



Tiny Kernel /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

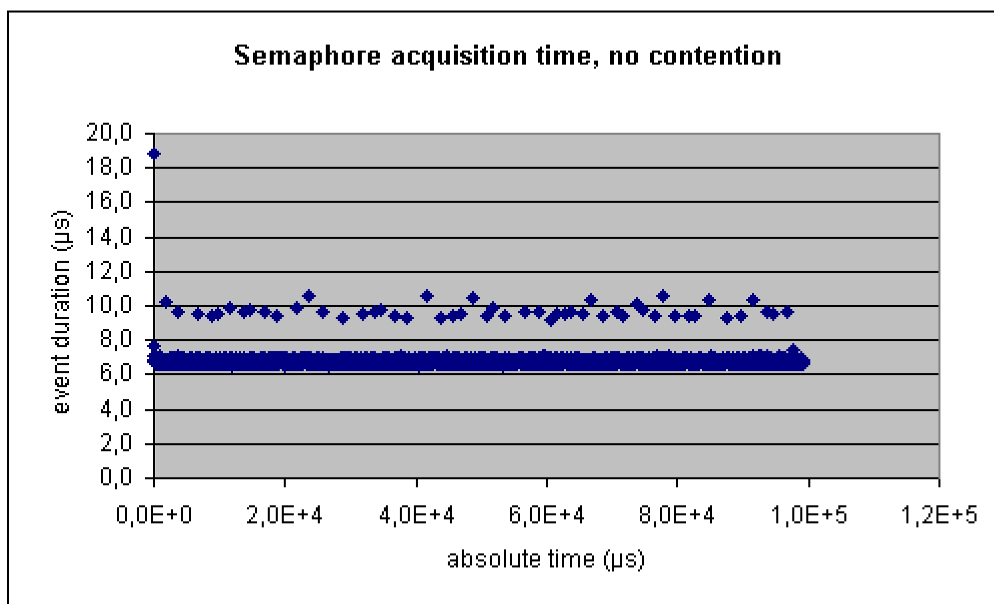
5.4.4.3 Test results on Enterprise Terminal Debug build /Kernel debugger used

Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, no-contention	7500	6.7 µs	18.8 µs	6.5 µs
Semaphore release time, no-contention	7500	6.0 µs	16.7 µs	5.8 µs

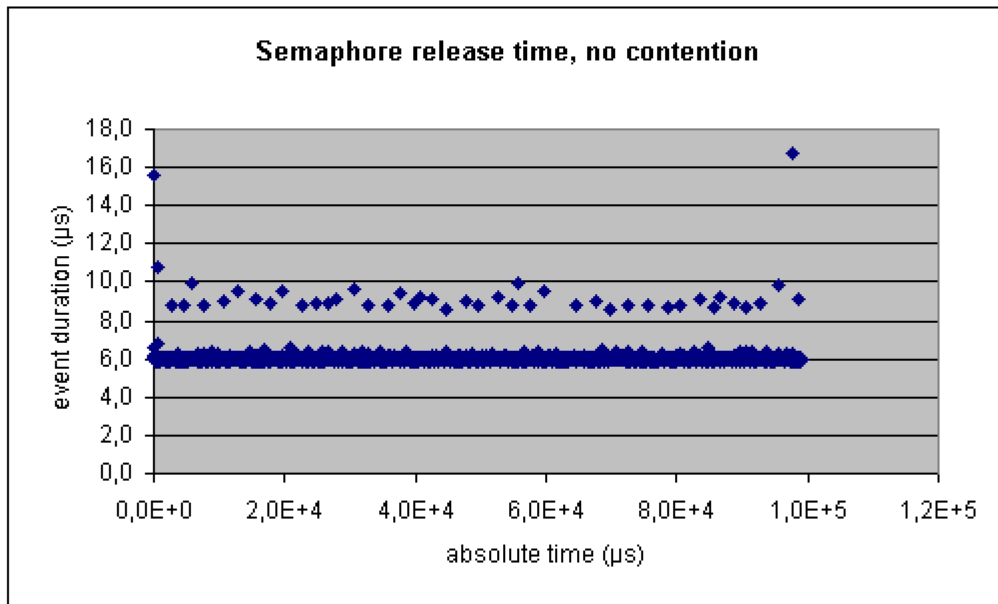
Diagrams:

Enterprise Terminal Debug build /Kernel debugger used



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Enterprise Terminal Debug build /Kernel debugger used



5.4.4.4 Test acquire-release timings: contention case (SEM-P-ARC)

This is used to test the time needed to acquire and release a semaphore depending on the number of threads blocked on the semaphore. It measures the time in the contention case: so when the acquisition and release system call causes a rescheduling to occur.

The aim of this test is to verify if the number of blocked threads has an impact on these timings. So this will answer the question: “how much time the operating system needs to find out the next thread to schedule”.

As can be seen on the detailed extract of the release timings (tiny configuration), the number of pending threads does not affect the release time. The first release is quicker due to caching issues: in the test 128 threads of different priorities are waiting on the semaphore. They grab the semaphore from low priority to high priority (threads created in this order). So when releasing the semaphore the highest priority thread activates again. As the highest priority thread was also the last one taking the semaphore, it will still be cached.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

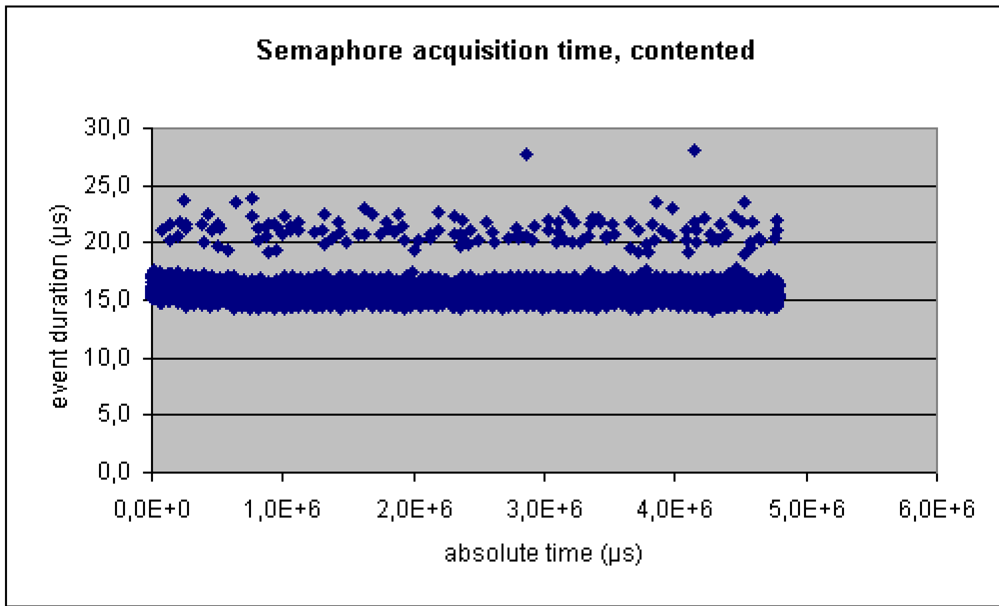
5.4.4.5 Test results on Enterprise Terminal

Test	result
Test succeeded	YES
Max number of threads pending	as much threads as memory resources allows.

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, contended	7500	15.6 µs	28.1 µs	14.2 µs
Semaphore release time, contended	7500	19.4 µs	27.0 µs	12.8 µs

Diagrams

Enterprise Terminal /Release Build

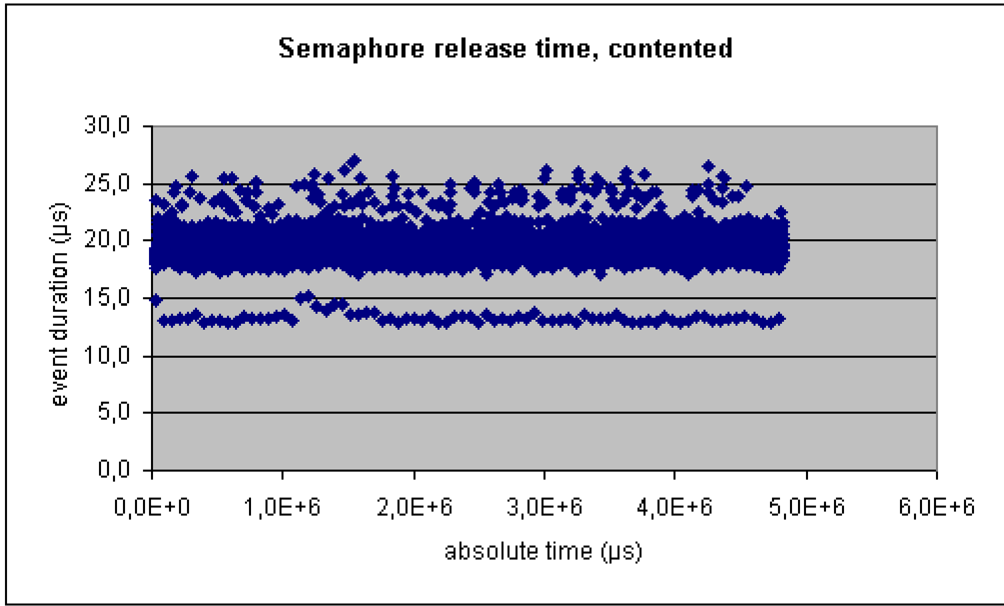


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Enterprise Terminal /Release Build



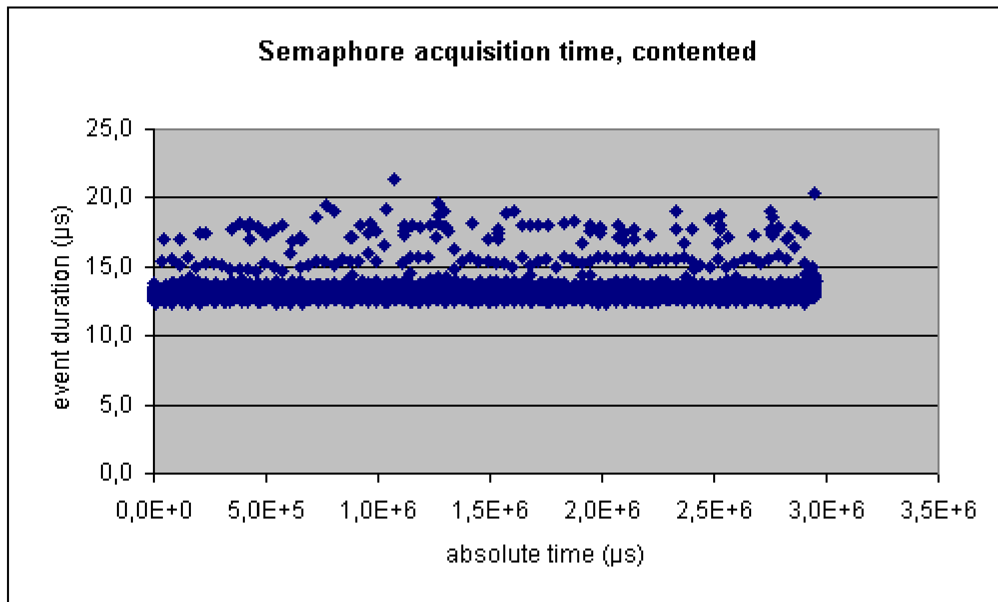
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.4.4.6 Test results on Tiny Kernel

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, contended	7500	13.1 µs	21.3 µs	12.2 µs
Semaphore release time, contended	7500	19.0 µs	25.6 µs	13.0 µs

Diagrams:

Tiny Kernel /Release Build

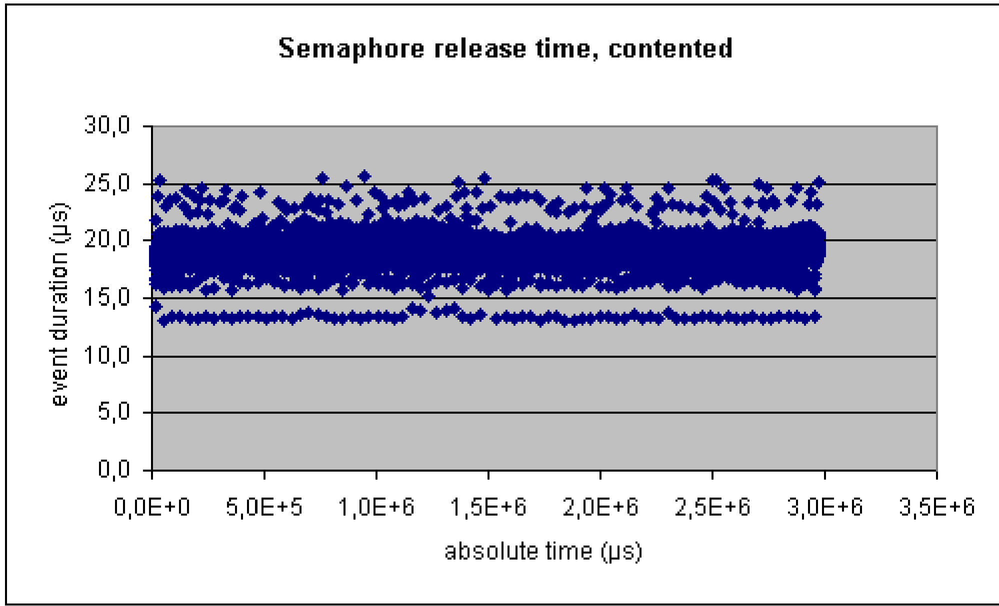


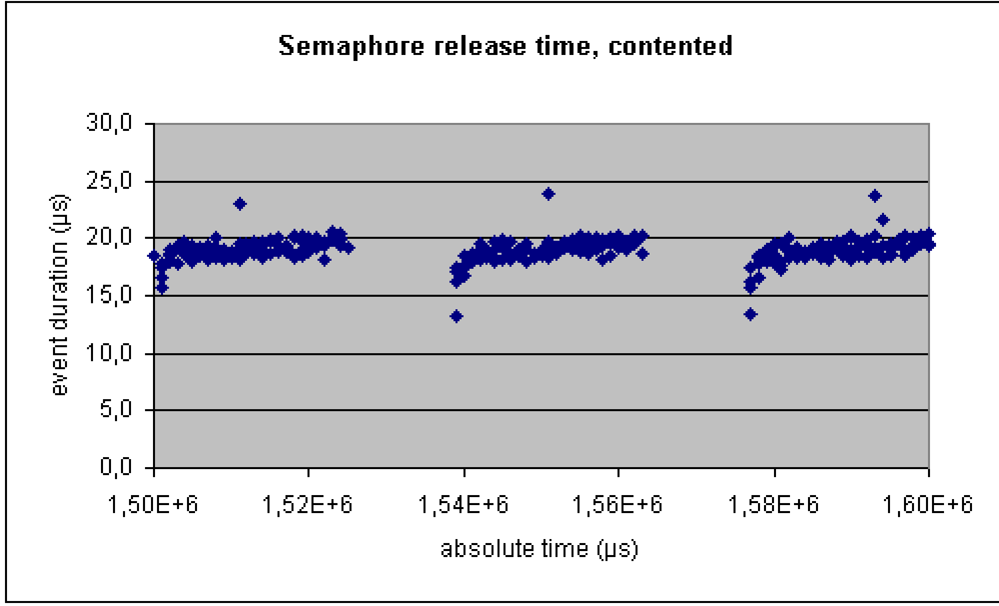
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Tiny Kernel /Release Build





Detailed extract from previous diagram

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.5 Mutex tests (MUT)

Here the performance and the behavior of the mutual exclusive semaphore are tested.

Although the mutual exclusive semaphore (further called mutex) could be the same as the counting semaphore where the count is one, this is not the aim of this test. In scope of the framework, this test will look into detail of a mutex system object that avoids priority inversion.

Windows CE implements an inheritance technique to avoid priority inversion it does this for three synchronization objects:

- Semaphores
- Mutexes
- Critical sections.

All three are verified if they indeed implement the priority inheritance. Mutexes and critical sections are also tested on performance. Semaphores were already tested in the previous section of this document.

As Microsoft claims that their counting semaphores uses priority inheritance as well, this behavior is tested in the section concerning mutex. Following our test definitions we define a mutex as “any protection object that provides mechanisms for avoiding priority inversions”. In case of Windows CE, a semaphore also has this ability.

For other operating systems, critical sections are normally not tested. We do this for Windows CE as CE uses also priority inheritance with the critical section primitive. The aim here is to test if all three mechanisms provide priority inheritance as Microsoft claims. And to test which one of the three mechanisms has the best performance.

5.5.1 Priority inversion avoidance mechanism (MUT-B-ARC)

This test will determine if the system call under test prevents the priority inversion case. Therefore the test will artificially create a priority inversion and verify if indeed the system raises the lower priority thread’s priority temporarily when that thread owns a synchronization object required by a higher priority thread.

5.5.1.1 Test results

Semaphore:

Test	result
Priority inversion avoidance system call present	YES
System call used	WaitForMultipleObjects / ReleaseSemaphore
Test succeeded	YES
Priority inversion avoided	YES
Mechanism used if any?	Temporary priority Inheritance

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Mutex:

Test	result
Priority inversion avoidance system call present	YES
System call used	WaitForMultipleObjects / ReleaseMutex
Test succeeded	YES
Priority inversion avoided	YES
Mechanism used if any?	Temporary priority Inheritance

Critical section:

Test	result
Priority inversion avoidance system call present	YES
System call used	EnterCriticalSection / LeaveCriticalSection
Test succeeded	YES
Priority inversion avoided	YES
Mechanism used if any?	Temporary priority inheritance

5.5.2 Mutex acquire- release timings: contention case (MUT-P-ARC)

This is the same test as above, but performed in a loop. In this case, the time is measured to acquire and release the mutex (and critical section) in the priority inversion case.

The acquisition time is the time for:

- the acquisition,
- activating the thread which has the lock
- raising the priority of this thread to the priority of the acquiring thread

The release time is the reverse:

- the release,
- lowering the thread that hat the lock
- activating the thread which took the lock.

Some remarkable findings in these tests in respect with the semaphore test:

- Critical sections are (a little bit) slower than mutexes!

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

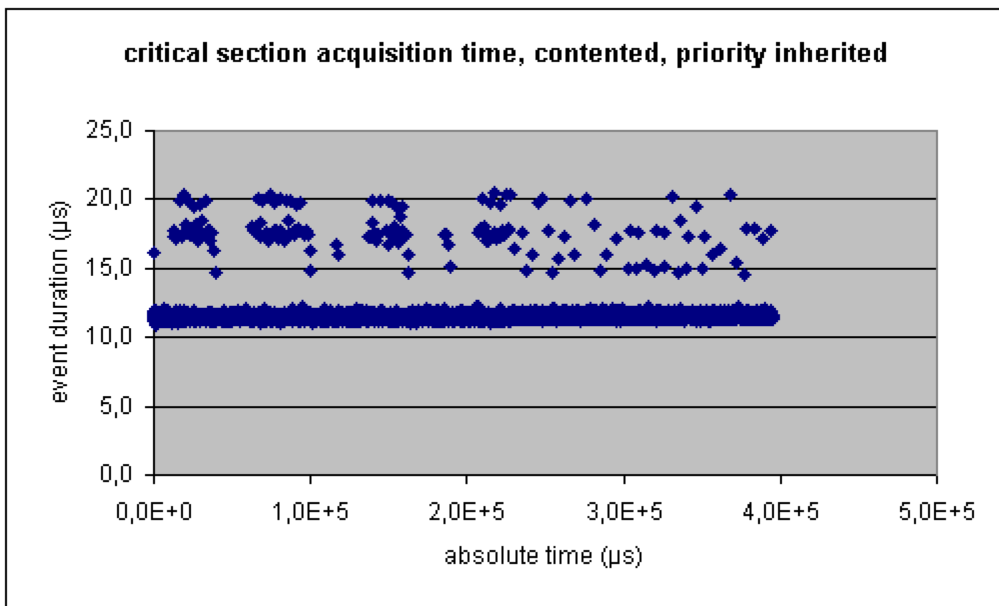
This is very strange: we would expect that critical sections would be faster as they can not be used between processes! This should make the handling simpler than the mutex/semaphore.

5.5.2.1 Test results on Enterprise Terminal /Release Build

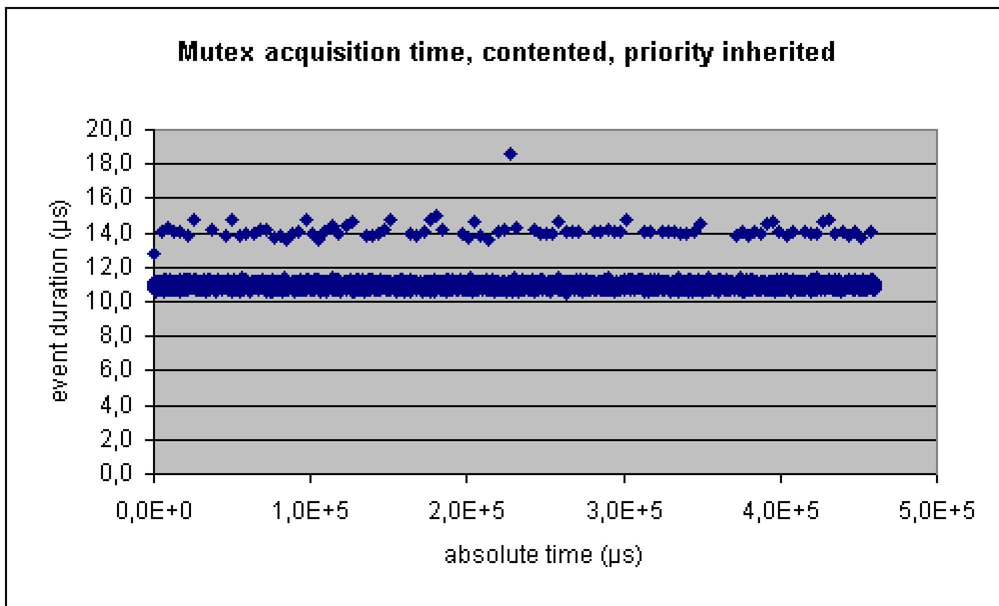
Test	result
Test succeeded	YES

Test	Sample qty	Avg	Max	Min
Critical sections acquire timings, contended	7500	11.6 µs	20.5 µs	10.9 µs
Mutex acquire timings, contended	7500	10.9 µs	18.5 µs	10.4 µs

Diagrams

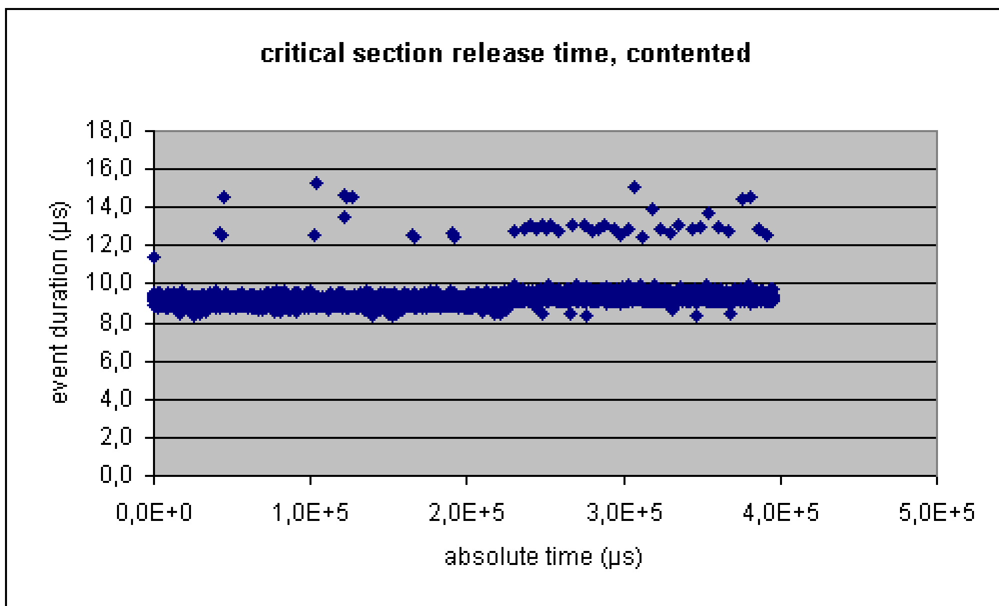


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

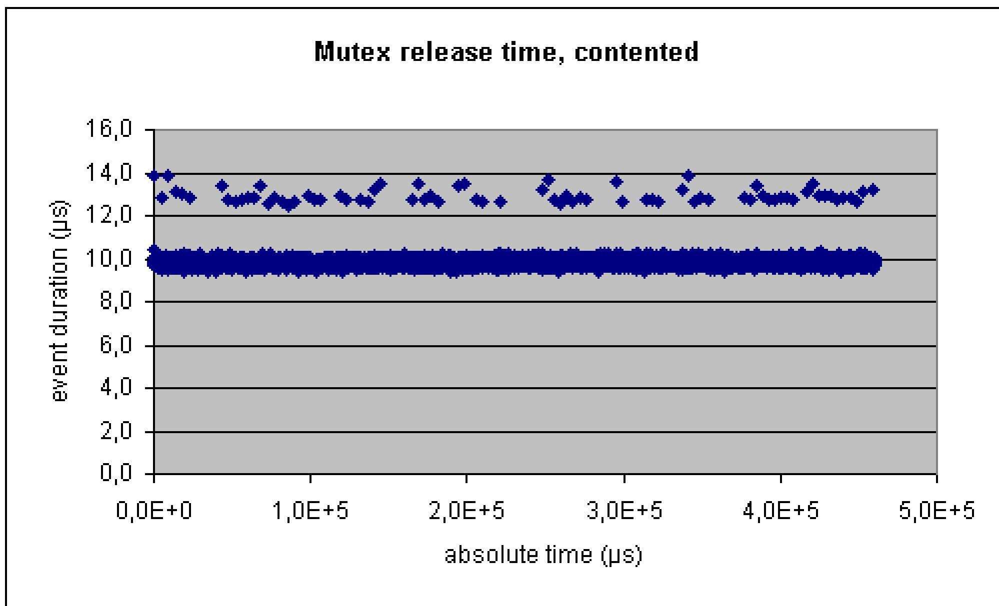


Test	Sample qty	Avg	Max	Min
Critical sections release timings, contended	7500	9.21 µs	15.28 µs	8.34 µs
Mutex release timings, contedted	7500	9.85 µs	13.84 µs	9.35 µs

Diagrams



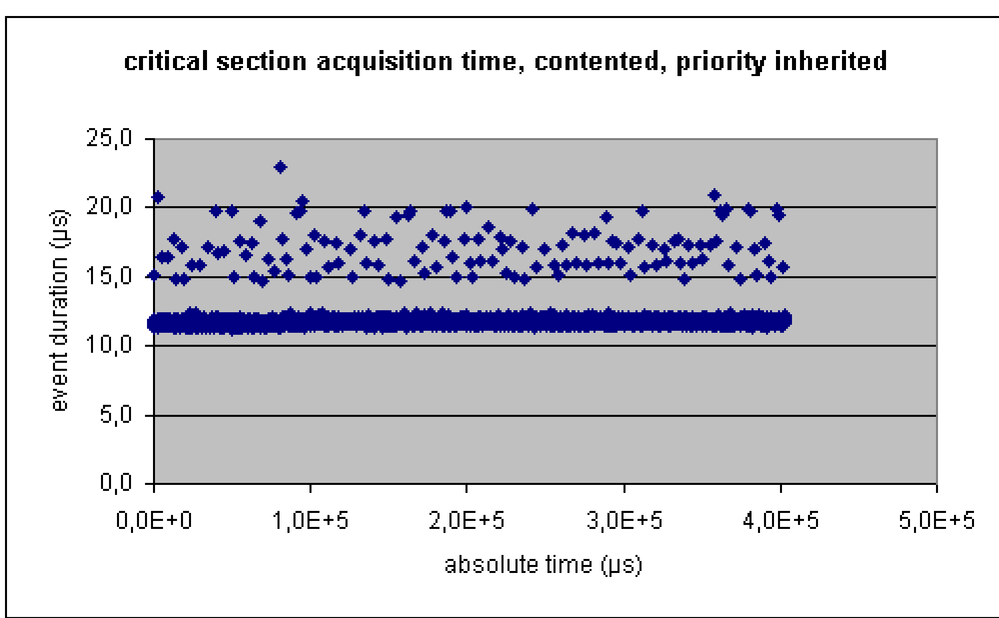
http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



5.5.2.2 Test results on Tiny Kernel /Release Build

Test	Sample qty	Avg	Max	Min
Critical sections acquire timings, contented	7500	11.8 µs	22.9 µs	11.2 µs
Mutex acquire timings, contented	7500	11.2 µs	14.9 µs	10.5 µs

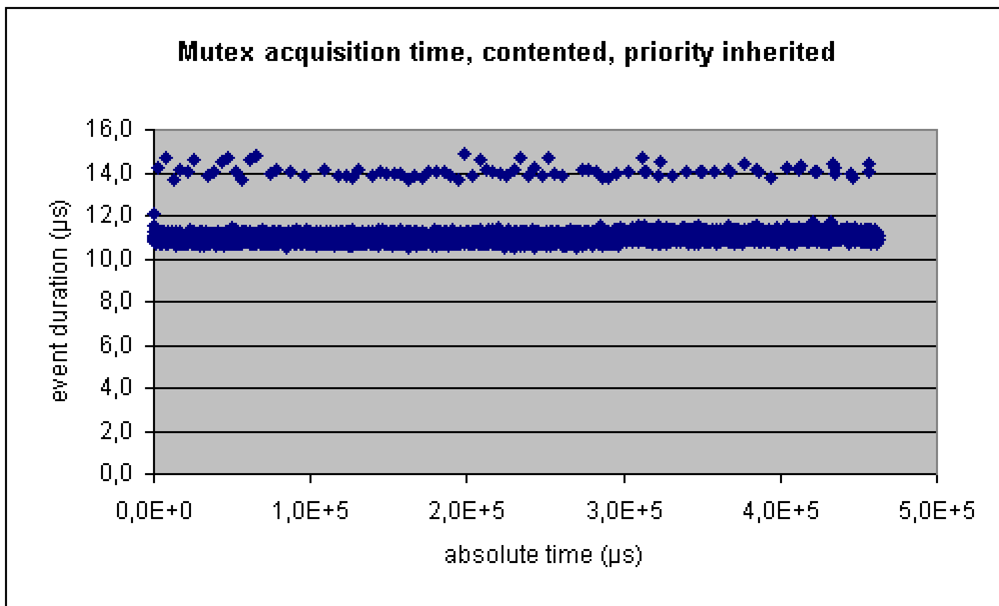
Diagrams



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

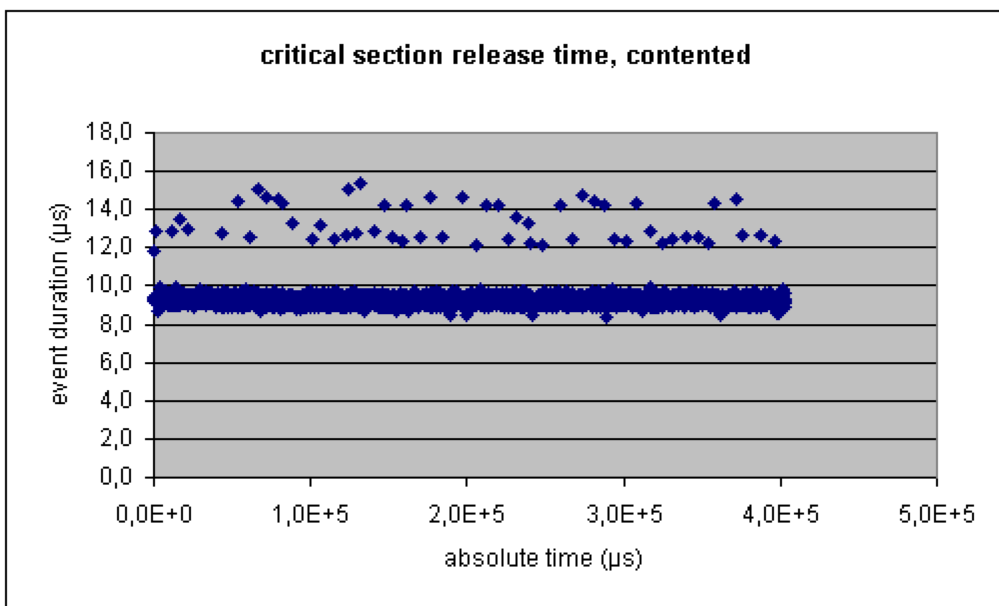
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



Test	Sample qty	Avg	Max	Min
Critical sections release timings, contended	7500	9.23 µs	15.34 µs	8.4 µs
Mutex release timings, contended	7500	10.0 µs	14.08 µs	9.59 µs

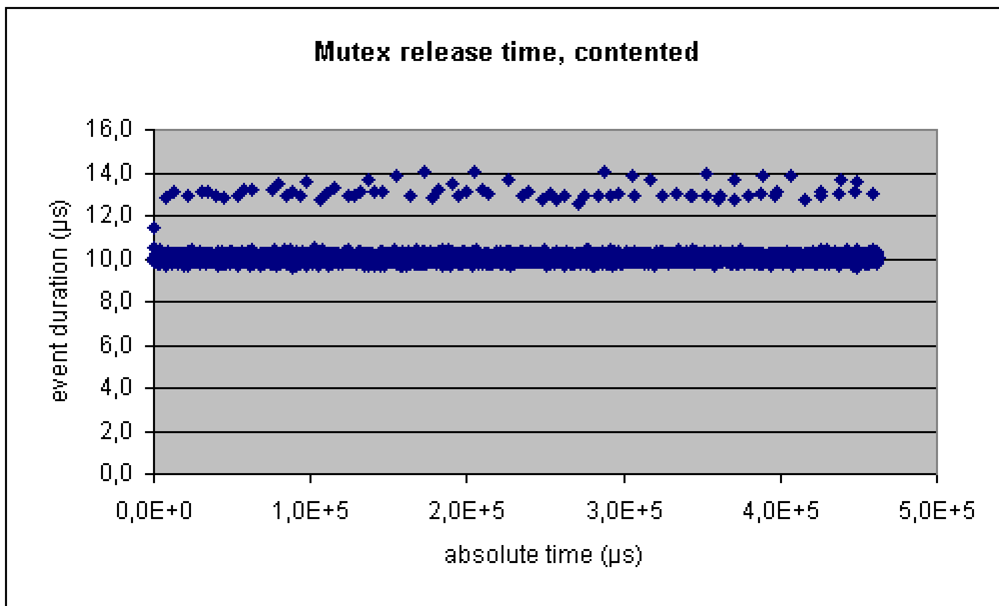
Diagrams



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

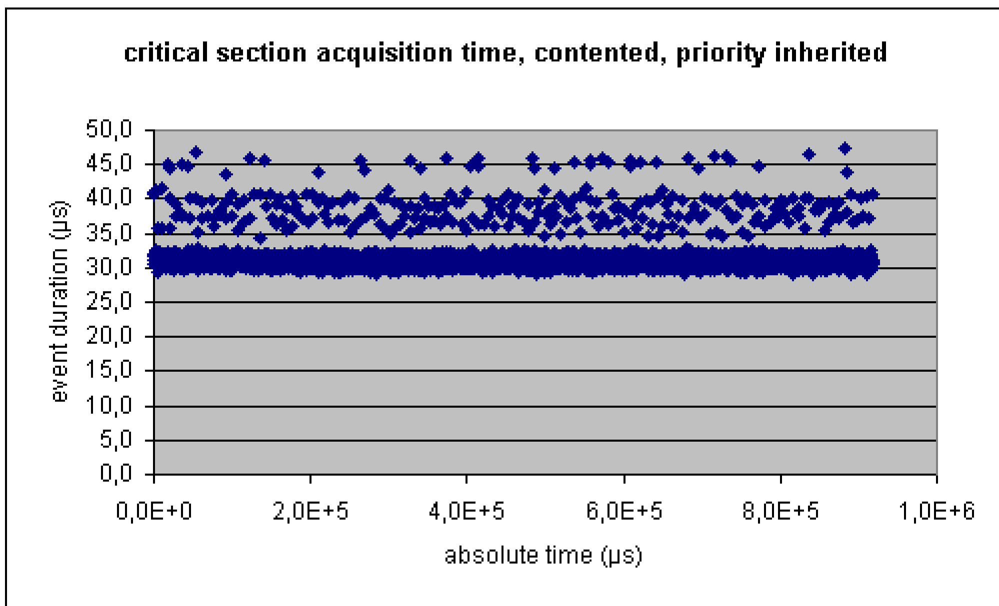
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



5.5.2.3 Test Results on Enterprise Terminal Debug build /Kernel debugger used

Test	Sample qty	Avg	Max	Min
Critical sections acquire timings, contended	7500	31.2 µs	47.5 µs	28.9 µs
Mutex acquire timings, contended	7500	20.3 µs	26.9 µs	18.9 µs

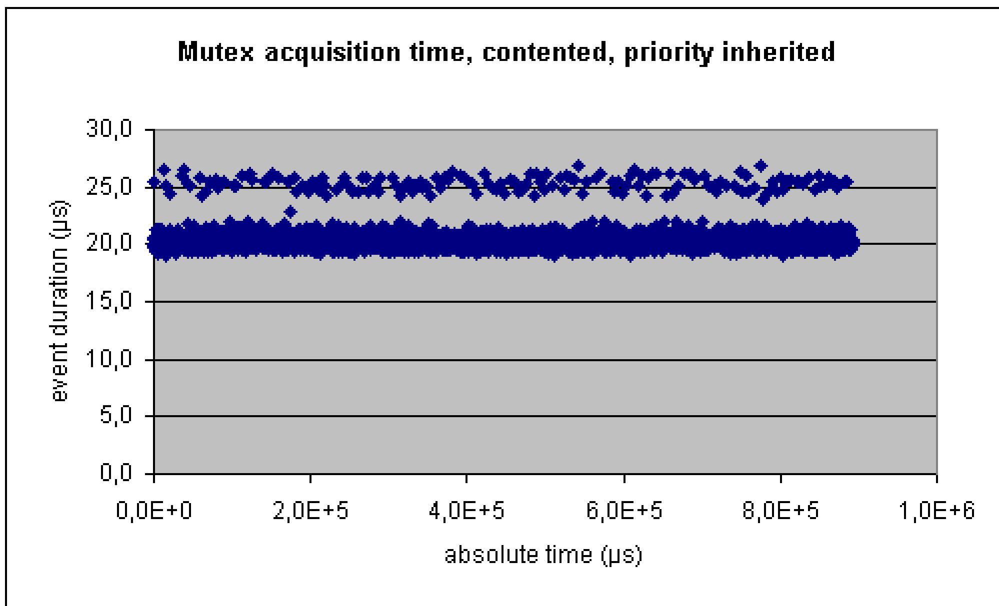
Diagrams



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

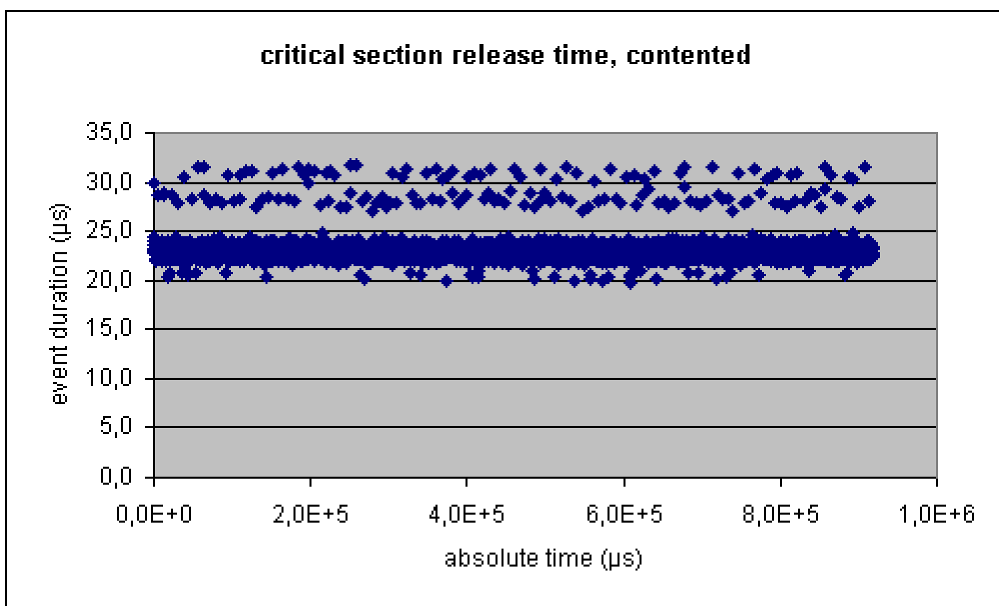
This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



Test	Sample qty	Avg	Max	Min
Critical sections release timings, contended	7500	22.95 µs	31.84 µs	19.7 µs
Mutex release timings, contended	7500	22.82 µs	31.72 µs	21.32 µs

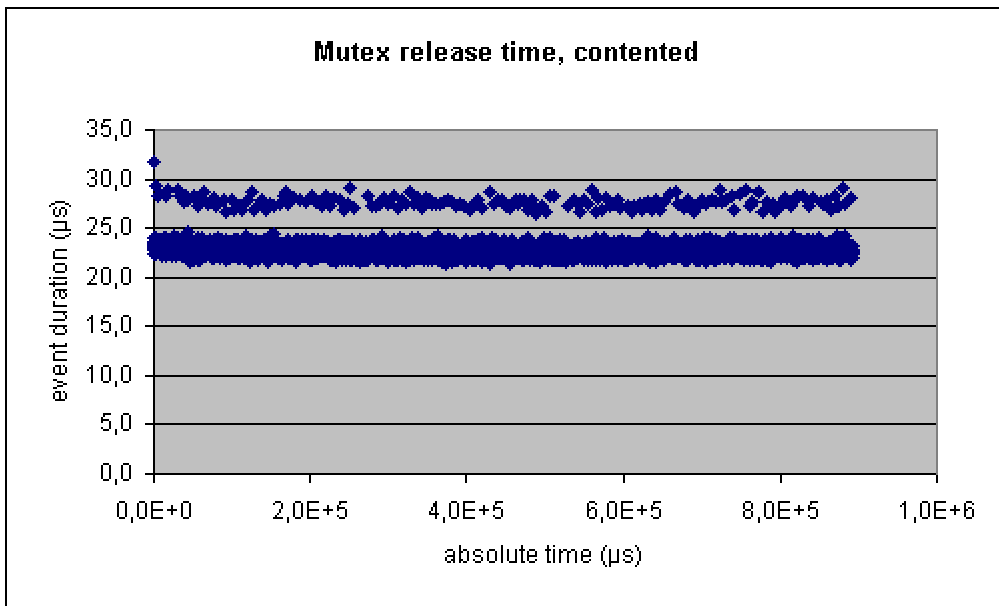
Diagrams



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6 Interrupt Tests (IRQ)

Here the performance of the interrupt handling in the operating system and hardware is tested.

In a real-time system, interrupt handling is a major part of the system: indeed such systems are typically event driven.

For these tests, our standard tracing system is adapted. Interrupts are generated by a plugged in PCI related card (can be PMC/PCI or CPCI). This card has a complete independent processor on board, with custom-made software. As such we can guarantee an independent interrupt source compared with the platform under test.

Microsoft uses a different way of handling interrupts in CE: The low level hardware interrupt (Interrupt Servicing Routine) is handled in the OAL and isn't meant to be used by drivers. Drivers will normally use an Interrupt Servicing Thread (IST) running as a normal thread at a certain thread priority! The low level hardware interrupt is handled in the OAL and if it is needed, it is possible to do specific interrupt handling at the OAL for handling critical real-time constraints, but than you need to adapt the OAL (which is not an easy job).

The advantages of the interrupt service routine (ISR) being handled at driver level as an interrupt service thread (IST) are:

- All system calls are available.
- You have the possibility to set the relative priorities of the ISTs so that the highest-priority interrupt handler receives the most reliable response (independent of the hardware interrupt level).

This gives you a great flexibility but increases the interrupt latency.

Therefore only the IST tests were run in scope of this report.

As can be seen in the test results shown further, no problems were detected and the results were predictable.

5.6.1 Simultaneous interrupt priority handling (IRQ_B_SIM)

This test verifies if simultaneous interrupts are handled prioritized. It answers the question if a lower priority interrupt can be pre-empted by a higher level interrupt.

This is done by starting the interrupt generation of one device in the interrupt handler of the other device.

As interrupts are handled at prioritised thread levels we didn't find any difficulties here (remember we are testing the IST).

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6.1.1 Test results on Tiny Kernel

Test	result
Test succeeded	YES
Interrupt pre-emption existing following the documentation?	YES
Lower level interrupt pre-empted by higher level interrupt?	YES
Higher level interrupt not pre-empted by lower level interrupt?	YES

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6.2 Interrupt latency (IRQ_P_LAT)

This measures the time it takes to switch from a running thread to an interrupt service thread (running at the highest priority level).

This is different compared with most other RTOS where you have a real interrupt servicing routine. So the results are difficult to compare. You should add the IRQ_P_LAT with IRQ_P_TLT results of traditional RTOS to compare with the results we got here.

The clock interrupt is detected again. Also the first sample is slower caused by caching issues. In fact: under normal circumstances the interrupt latency will be more likely the un-cached result, except if the interrupt occurs a lot. Anyhow, any designer should use the "un-cached" result as a real value.

5.6.2.1 Test results on Enterprise Terminal /Release Build

Test	Sample qty	Avg	Max	Min
Interrupt dispatch latency	676	8.8 µs	14.8 µs	8.4 µs

5.6.2.2 Test results on Tiny Kernel /Release Build

Test	Sample qty	Avg	Max	Min
Interrupt dispatch latency	263	8.8 µs	13.31 µs	8.4 µs

5.6.2.3 Test results on Enterprise Terminal /Debug build /Kernel debugger used

Test	Sample qty	Avg	Max	Min
Interrupt dispatch latency	258	13.54 µs	24.43 µs	12.29 µs

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagram : Enterprise Terminal /Release Build

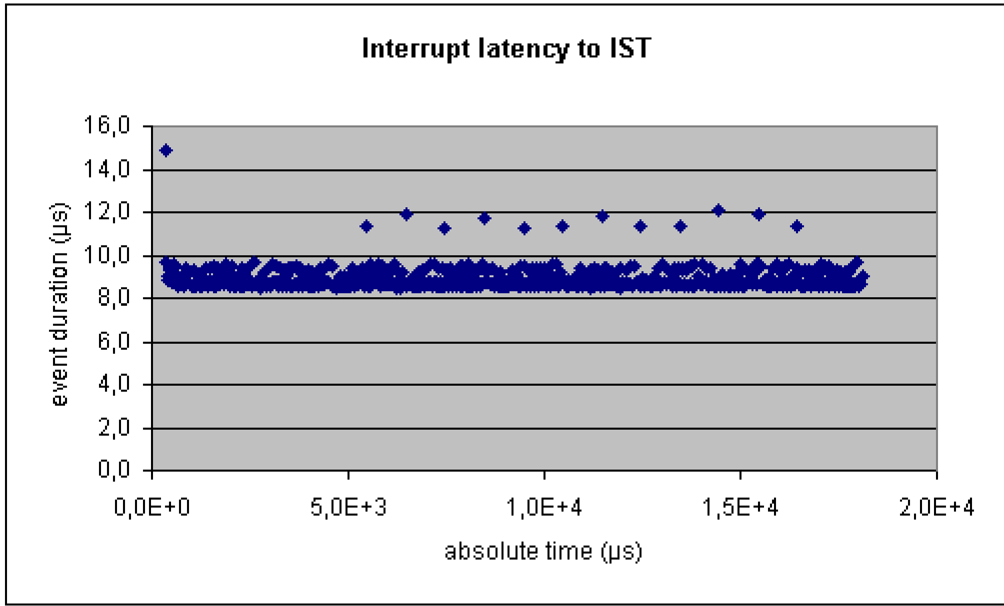
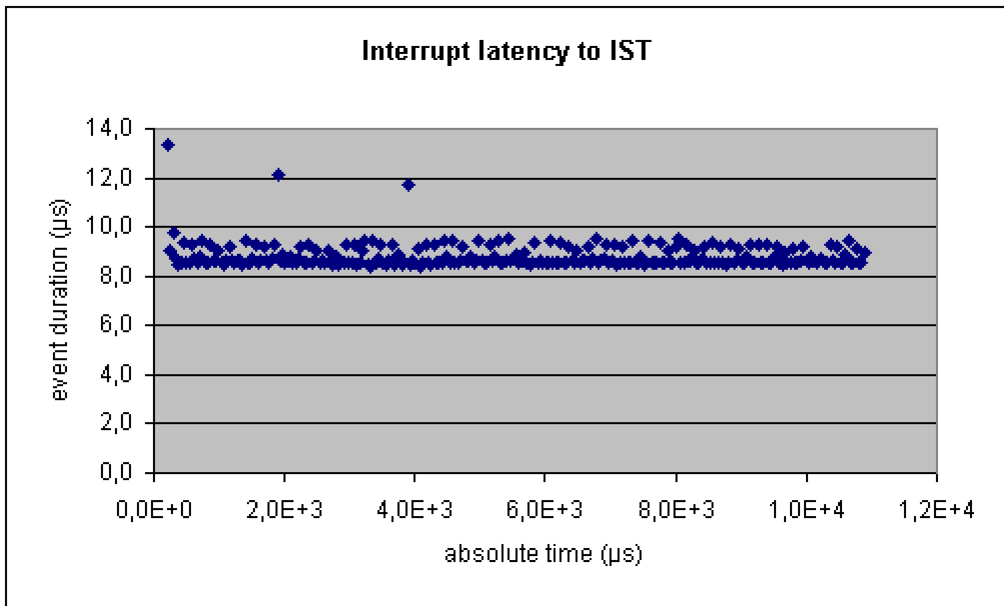


Diagram : Tiny Kernel /Release Build

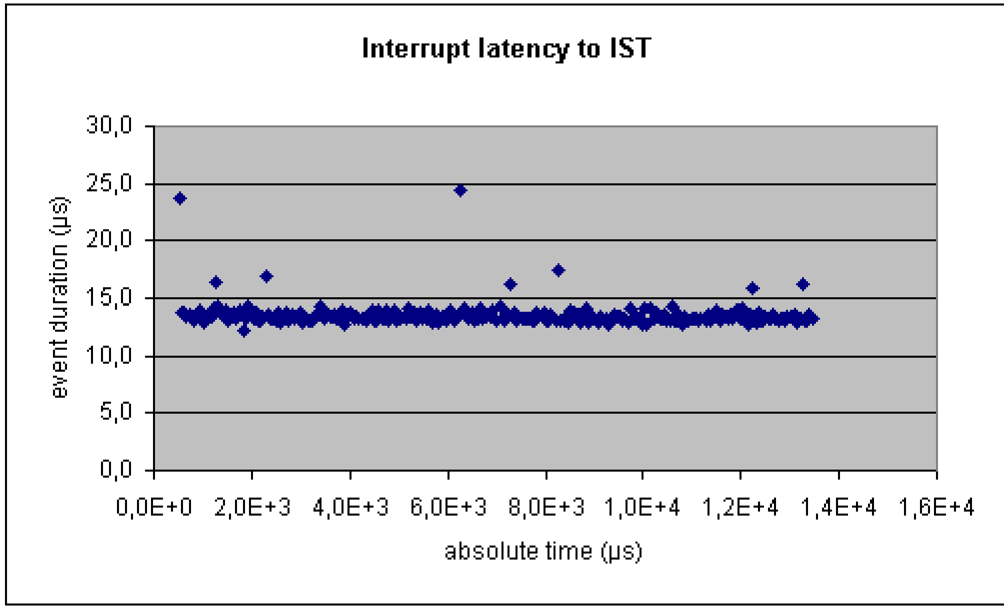


http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagram : Enterprise Terminal /Debug build /Kernel debugger used



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6.3 Interrupt dispatch latency (IRQ_P_DLT)

This measures the time it takes to switch from the interrupt service thread back to the interrupted thread.

The same remark concerning the clock interrupt and caching issues is valid here.

5.6.3.1 Test results on Enterprise Terminal /Release Build

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	672	7.5 µs	10.6 µs	7.2 µs

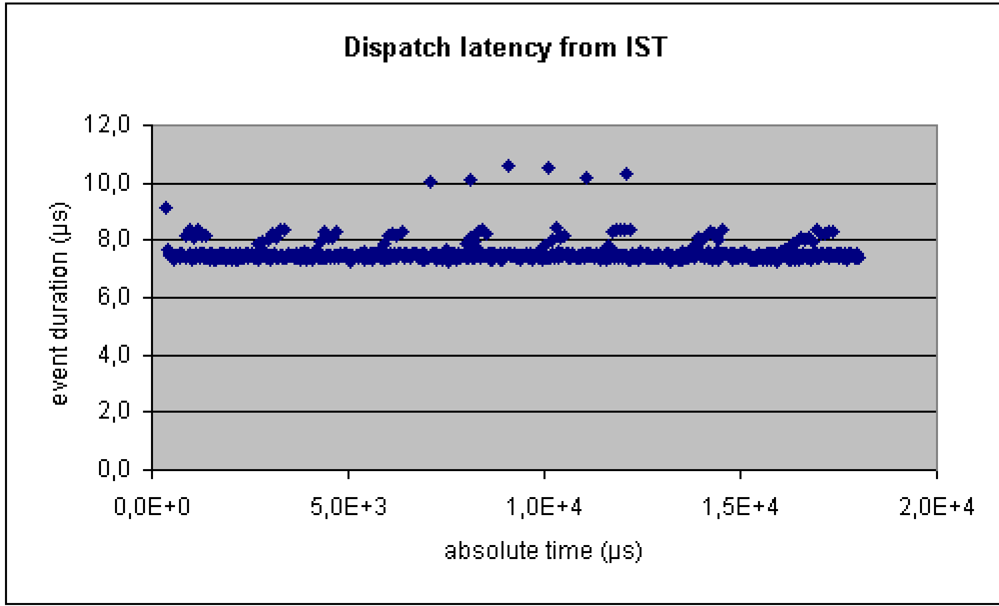
5.6.3.2 Test results on Tiny Kernel /Release Build

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	262	7.45 µs	9.17 µs	7.17 µs

5.6.3.3 Test results on Enterprise Terminal /Debug build /Kernel debugger used

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	774	11.92 µs	16.30 µs	11.15 µs

Diagram : Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagram : Tiny Kernel /Release Build

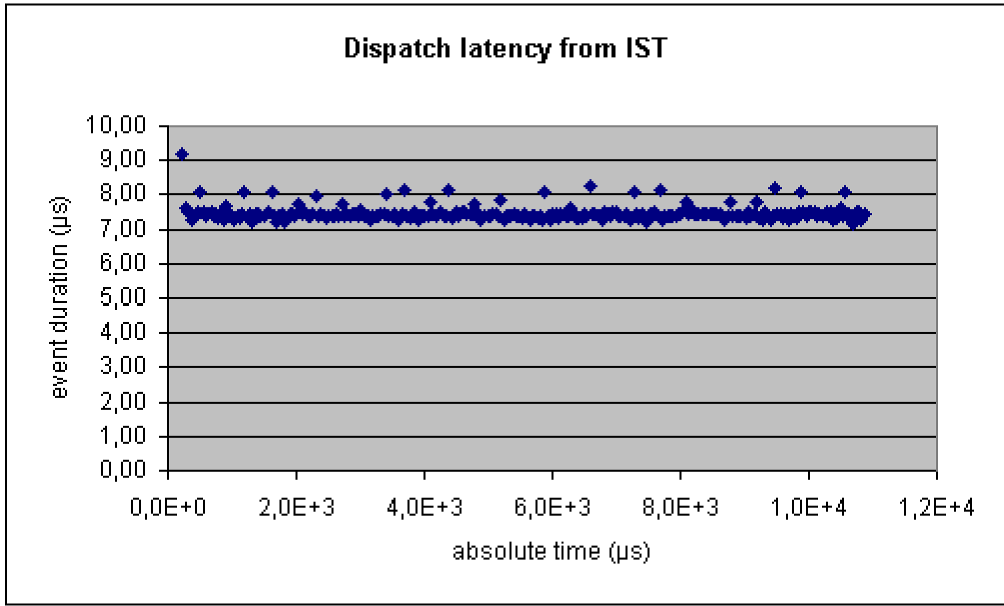
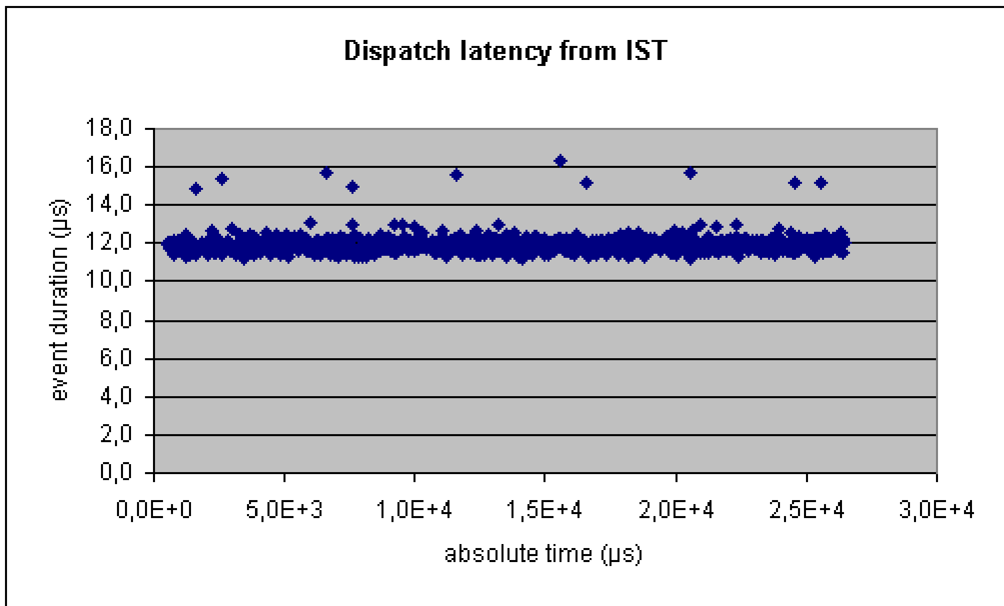


Diagram : Enterprise Terminal /Debug build /Kernel debugger used



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6.4 Interrupt to thread latency (IRQ_P_TLT)

This measures the time it takes to switch from the interrupt service thread to the thread that is activated (by using a semaphore) from the interrupt service thread.

As the interrupt is already running at thread level, this test is in fact about the same of the semaphore acquisition/release test!

The same remark concerning the clock interrupt and caching issues is valid here.

5.6.4.1 Test results on Enterprise Terminal /Release Build

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	10000	11.4 µs	20.4 µs	10.4 µs

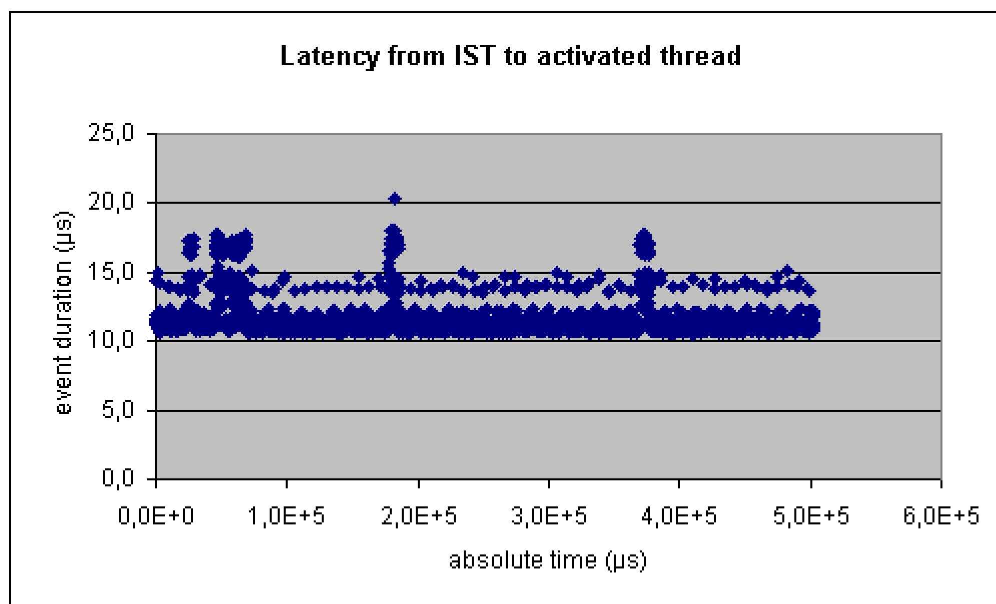
5.6.4.2 Test results on Tiny Kernel /Release Build

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	10000	11.16 µs	18.61 µs	10.02 µs

5.6.4.3 Test results on Enterprise Terminal /Debug build /Kernel debugger used

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler	9516	19.38 µs	25.39 µs	15.34 µs

Diagram on Enterprise Terminal /Release Build



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Diagram: Tiny Kernel /Release Build

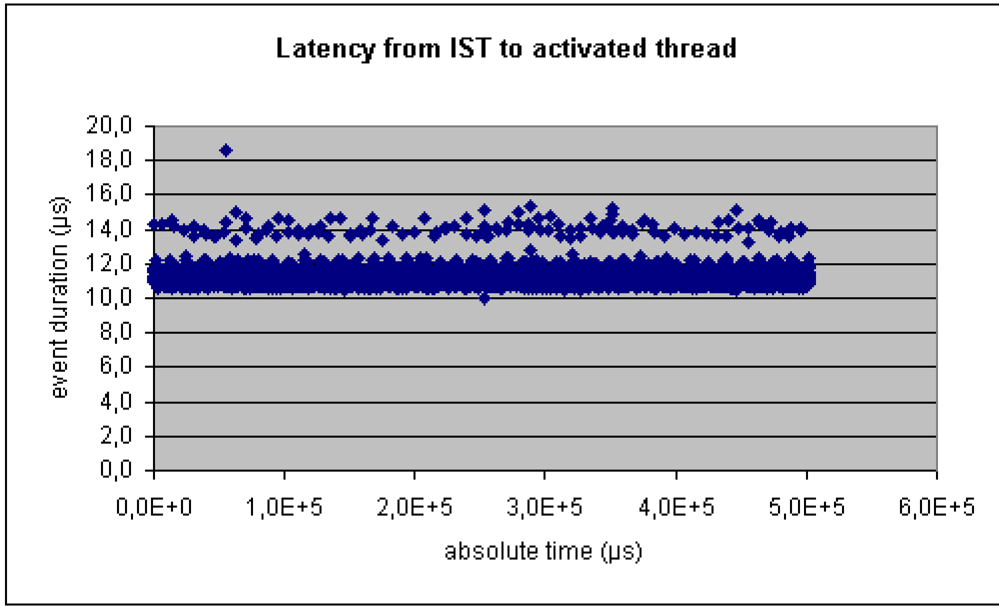
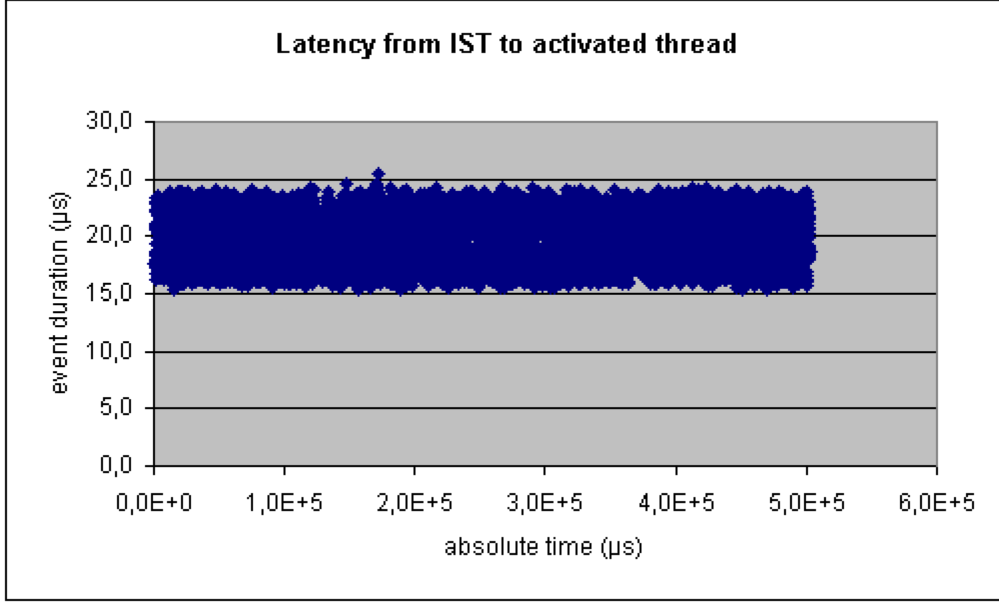


Diagram on Enterprise Terminal /Debug build /Kernel debugger used



http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.6.5 Maximum sustained interrupt frequency (IRQ_S_SUS)

This test measures the probability an interrupt is missed: is the interrupt handling duration stable and predictable?

The test is done on three levels:

– 100 000 interrupts, initial phase: each test takes only some seconds.

– 100 000 000 interrupts, second phase based on the results from the first phase. This test takes less than two hours and gives already accurate results.

– 1 000 000 000: third phase (one billion interrupts), based on the results of previous phase. The probability something goes wrong is ten times higher. Disadvantage is the test duration which takes multiple hours!

Windows CE passes this test. The minimum sustained interrupt period is 25 µs. This changes to 35 µs in longer test runs! These results seem long for an RTOS but you have to keep in mind that these are the timings from the IST and not from the ISR.

5.6.5.1 Test results

Interrupt period	#interrupts generated	#interrupts serviced	#interrupts lost
20 µs	100 000	99 972	28
23 µs	100 000	99 997	3
25 µs	100 000	100 000	0
28 µs	100 000	100 000	0
26 µs	100 000 000	99 999 999	1
28 µs	100 000 000	100 000 000	0
30 µs	100 000 000	100 000 000	0
27 µs	1 000 000 000	1 000 000 000	0

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5.7 Memory tests

This tests the OS for memory leaks.

5.7.1 Memory leak test (MEM_B_LEK)

This test continuously create/remove objects in the operating system (threads, semaphores, mutexes, …).

Ce5.0 passed this test, no memory leaks were detected.

Test	result
Test succeeded	YES
Test duration (how long we let the endless loop run)	>20h
Number of main test loops done	>1.000.000

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

6 Support

Support	0	8	10

Support is done well. Via email you receive a response within a couple of days.

Most problems were solved within a couple of days.

Most of the information can be found on the internet. The best documentation can be found on MSDN, with howtos and other technical support.

If you can't find your support on the web, you can contact the Microsoft team via email. We encountered a bug which caused a memory leak (not caused by the operating system), but their suggestion solved the problem quickly.

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

7 Appendix A: Vendor comments

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info
This licensed copy is owned by Microsoft Corporation, Inc.
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

8 Appendix B: Acronyms

Acronym	Explanation
API	Application Programmers Interface: calls used to call code from a library or system.
BSP	Board Support Package: all code and device drivers to get the OS running on a certain board
DSP	Digital Signal Processor
FIFO	First In First Out: a queuing rule
GPOS	General Purpose Operating System
GUI	Graphical User Interface
IDE	Integrated Development Environment (GUI tool used to develop and debug applications)
IRQ	Interrupt Request (hardware interrupt line)
ISR	Interrupt Servicing Routine
MMU	Memory Management Unit
OS	Operating System
PCI	Peripheral Component Interconnect: bus to connect devices, used in all PCs!
PIC	Programmable Interrupt Controller
PMC	PCI Mezzanine Card
PrPMC	Processor PMC: a PMC with the processor
RTOS	Real-Time Operating System
SDK	Software Development Kit
SoC	System on a Chip

http://www.dedicated-systems.com
E-mail: info@dedicated-systems.info

This licensed copy is owned by Microsoft Corporation, Inc.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

9 Appendix C: Document revision history

9.1 Issue 1.0 (October 8, 2004)

First official version of this report.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

EVALUATION REPORT DEFINITION

© Copyright Dedicated Systems Experts NV. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts NV, Bergensesteenweg 421 B12, B-1600 St-Pieters-Leeuw, Belgium.

Disclaimer

Although all care has been taken to obtain correct information and accurate test results, Dedicated Systems Experts and Dedicated Systems Magazine 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 and Dedicated Systems Magazine have been advised of the possibility of such damages.

http://www.dedicated-systems.com

Email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

EVALUATION REPORT LICENSE

This is a legal agreement between you (the downloader of this document) and/or your company and the company DEDICATED SYSTEMS EXPERTS NV, Bergensesteenweg 421 B12, B-1600 St-Pieters-Leeuw, Belgium.
It is not possible to download this document without registering and accepting this agreement on-line.

1. **GRANT**. Subject to the provisions contained herein, Dedicated Systems Experts hereby grants you a non-exclusive license to use its accompanying proprietary evaluation report for projects where you or your company are involved as major contractor or subcontractor. You are not entitled to support or telephone assistance in connection with this license.
2. **PRODUCT**. Dedicated Systems Experts shall furnish the evaluation report to you electronically via Internet. This license does not grant you any right to any enhancement or update to the document.
3. **TITLE**. Title, ownership rights, and intellectual property rights in and to the document shall remain in Dedicated Systems Experts and/or its suppliers or evaluated product manufacturers. The copyright laws of Belgium and all international copyright treaties protect the documents.
4. **CONTENT**. Title, ownership rights, and an intellectual property right in and to the content accessed through the document is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives you no rights to such content.
5. **YOU CAN NOT**:
   - You cannot, make (or allow anyone else make) copies, whether digital, printed, photographic or others, except for backup reasons. The number of copies should be limited to 2. The copies should be exact replicates of the original (in paper or electronic format) with all copyright notices and logos.
   - You cannot, place (or allow anyone else place) the evaluation report on an electronic board or other form of on line service without authorization.
6. **INDEMNIFICATION**. You agree to indemnify and hold harmless Dedicated Systems Experts against any damages or liability of any kind arising from any use of this product other than the permitted uses specified in this agreement.
7. **DISCLAIMER OF WARRANTY**. All documents published by Dedicated Systems Experts on the World Wide Web Server or by any other means are provided "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. This disclaimer of warranty constitutes an essential part of the agreement.
8. **LIMITATION OF LIABILITY**. Neither Dedicated Systems Experts nor any of its directors, employees, partners or agents shall, under any circumstances, be liable to any person for any special, incidental, indirect or consequential damages, including, without limitation, damages resulting from use of OR RELIANCE ON the INFORMATION presented, loss of profits or revenues or costs of replacement goods, even if informed in advance of the possibility of such damages.
9. **ACCURACY OF INFORMATION**. Every effort has been made to ensure the accuracy of the information presented herein. However Dedicated Systems Experts assumes no responsibility for the accuracy of the information. Product information is subject to change without notice. Changes, if any, will be incorporated in new editions of these publications. Dedicated Systems Experts may make improvements and/or changes in the products and/or the programs described in these publications at any time without notice. Mention of non-Dedicated Systems Experts products or services is for information purposes only and constitutes neither an endorsement nor a recommendation.
10. **JURISDICTION**. In case of any problems, the court of BRUSSELS-BELGIUM will have exclusive jurisdiction.

Agreed by downloading the document via the Internet.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

1 Introduction 6
1.1 Purpose and scope 6
1.2 Document issue: the 2.9 framework 6
1.3 Related documents 7
2 The evaluation report 8
2.1 Introduction 8
2.1.1 The evaluation framework 8
2.1.2 The two evaluation tracks 8
2.2 The evaluation report layout 9
2.3 Measurement method 11
2.3.1 Tracing PCI access cycles 11
2.3.2 State analysis 12
2.3.3 Statistical analysis 12
2.3.4 Generating interrupts 12
2.4 System configuration parameters 13
2.4.1 Memory Protection model 13
3 Testing overview 14
3.1 Naming of the test series 14
3.1.1 The Backus-Naur Form 14
3.1.2 Test identifiers 14
3.1.3 Diagram identifiers 15
3.1.4 Source code identifier 15
3.2 Coding style 18
3.2.1 Identifiers 18
3.2.2 Bracing styles 19
3.2.3 Indenting 19
3.2.4 Code blocks and comments 20
3.3 Libraries 21
3.3.1 Tracing API 21
3.3.2 Interrupt generating API 22
3.3.3 Generic operating system API 22
4 The tests described 28
4.1 Calibration system test (CAL) 28
4.1.1 Tracing overhead (CAL_P_TRC) 28
4.1.2 CPU power (CAL_P_CPU) 29
4.2 Clock tests (CLK) 30
4.2.1 Operating system clock setting (CLK_B_CFG) 30
4.2.2 Clock tick processing duration (CLK_P_DUR) 31
4.3 Thread tests (THR) 32
4.3.1 Thread creation behaviour (THR_B_NEW) 32
4.3.2 Round robin behaviour (THR_B_RR) 33

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.3.3 Thread switch latency between same priority threads (THR_P_SLS) ... 34
4.3.4 Thread creation and deletion time (THR_P_NEW) ... 36
4.4 Semaphore tests (SEM) ... 38
4.4.1 Semaphore locking test mechanism (SEM_B_LCK) ... 39
4.4.2 Semaphore releasing mechanism (SEM-B-REL) ... 40
4.4.3 Time needed to create and delete a semaphore (SEM_P_NEW) ... 41
4.4.4 Test acquire-release timings: no contention case (SEM_P_ARN) ... 42
4.4.5 Test acquire-release timings: contention case (SEM_P_ARC) ... 43
4.5 Mutex tests (MUT) ... 45
4.5.1 Priority inversion avoidance mechanism (MUT-B-ARC) ... 46
4.5.2 Mutex acquire-release timings: contention case (MUT_P_ARC) ... 51
4.6 Interrupt tests (IRQ) ... 52
4.6.1 Simultaneous interrupt priority handling (IRQ_B_SIM) ... 53
4.6.2 Interrupt latency (IRQ_P_LAT) ... 54
4.6.3 Interrupt dispatch latency (IRQ_P_DLT) ... 54
4.6.4 Interrupt to thread latency (IRQ_P_TLT) ... 55
4.6.5 Maximum sustained interrupt frequency (IRQ_S_SUS) ... 55
4.7 Memory tests (MEM) ... 56
4.7.1 Memory leak test (MEM_B_LEK) ... 56
5 Appendix A: Document revision history ... 57
5.1 Issue 1.0 (April 29, 2004) ... 57

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

DOCUMENT CHANGE LOG

Issue No.	Revised Issue Date	Para's / Pages Affected	Reason for Change
1	April 29, 2004	All	Initial Issue

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

1 Introduction

1.1 Purpose and scope

This paper explains the evaluation tests done and how the results are presented in the RTOS evaluation report. It explains how framework 2.9 describes the tested real-time OS and shows how detailed the results are. This document is an important companion document in reading and understanding the evaluation reports.

The evaluation report itself shows the test results obtained for a particular RTOS. The executed tests are explained here. Due to the tightly coupling between these documents, the issue of this document has to match the issue of the evaluation report. Information on how we handle versions of documents and tests can be found in "The evaluation framework.", see section 1.3 of this document.

This document defines and explains all tests executed in the framework 2.9. Generic pseudo "C" code for these tests is available in another document which can be obtained on request. The results of the test will indicate if a system may or may not be qualified as "real-time". For each test, objective criteria are set to determine if a result is real-time or not. (real-time is used here in the sense of "predictable response time and behavior"

1.2 Document issue: the 2.9 framework

Framework 2.9 is a serious revision of the previous framework to achieve:

- Better readability
- More objective qualification criteria

These are the main changes between the current (2.9) and the previous (2.5) framework:

- Test labeling is completely changed to improve the readability of the evaluation reports.
  Remark that in appendix, a lookup table is added to find the relation between the test labels used in the previous framework and the current framework.
- Separation of the RTOS memory model and the test labeling, again to improve readability.
- Generic pseudo code is made for each test.
  This code is based on the "C" programming language and uses macros for the RTOS dependent system calls. (Published in a separate document).
- Added a new test category to test the RTOS "Behavior".
  These tests check if the operating system acts as expected in a certain scenario we define. These scenarios correspond to classical scenarios one would use in an application.
- Added some objective criteria to differentiate between "real-time" and "non real-time".

Generic pseudo code is now available for download on request. Third parties can therefore more easily repeat the tests presented here. However, these third parties should also be aware of the fact that one

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

needs extensive and costly measuring equipment to do the timing measurements in order for these measurements not to impact on the behavior of the system!

1.3 Related documents

These are documents that are closely related to this document. They can all be downloaded using following link:
http://www.dedicated-systems.com/encyc/buyersguide/rtos/evaluations

Doc. 1 The evaluation framework.
This document presents the evaluation framework. It also indicates which documents are available, and how their name giving, numbering and versioning are related. This document is the base document of the evaluation framework.
EVA-2.9-GEN-01 Issue: 1 Date: April 29, 2004

Doc. 2 What is a good RTOS?
This document presents the criteria that Dedicated Systems Experts use to give an operating system the label “Real-Time”. The evaluation tests are based upon the criteria defined in this document.
EVA-2.9-GEN-02 Issue: 1 Date: TBD

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

2 The evaluation report

2.1 Introduction

2.1.1 The evaluation framework

The evaluation framework is introduced in [Doc. 1]. This document also describes the aim of the evaluation reports. Our concept of real-time is introduced there. More ideas and concepts used in this document are explained in the paper “What is a good RTOS?”, see [Doc. 2] in section 1.3 of this document.

The tests are designed in order to verify if:

- A system behaves like expected
- The system response is predictable in all circumstances (worst case time needed?)
- A system does not have buggy behavior or instable behavior.

Depending on the results of the tests a system is labeled “RT-VALIDATED”, “VALIDATED” or “DID NOT QUALIFY”. Logos are available for each of the qualifications.

2.1.2 The two evaluation tracks

Our evaluation is divided into two tracks:

- The first track is a qualitative study called the technical evaluation. In this approach, we focus on the system architecture of the operating system and/or the architecture of the platform.

- The second track is referred as the practical evaluation and is a quantitative approach. This second track is detailed in depth in this document.

2.1.2.1 Qualitative track: The technical evaluation

The two technical evaluations closely related with a test report are:

- The operating system evaluation
- The platform evaluation.

Both evaluations are merely theoretical. The content and layout of these reports are given in two other documents, but these are not mandatory to understand the evaluation reports in this case.

2.1.2.2 Quantitative track: The practical evaluation

The practical evaluation measures specific real-time features of the operating system. It does not certify that a set of application threads will meet their deadlines; rate monotonic scheduling and similar methods can be used for this purpose. The purpose is to assert that a RTOS is suitable or not for real-time applications. If the behavior of an application needs to be predictable, then the underlying software, i.e. the RTOS, needs to have all the features necessary to meet these requirements. In general, all the system calls and operations of an operating system should exhibit predictable behavior. This implies that the execution time has to be bounded, independent of the workload of the system.

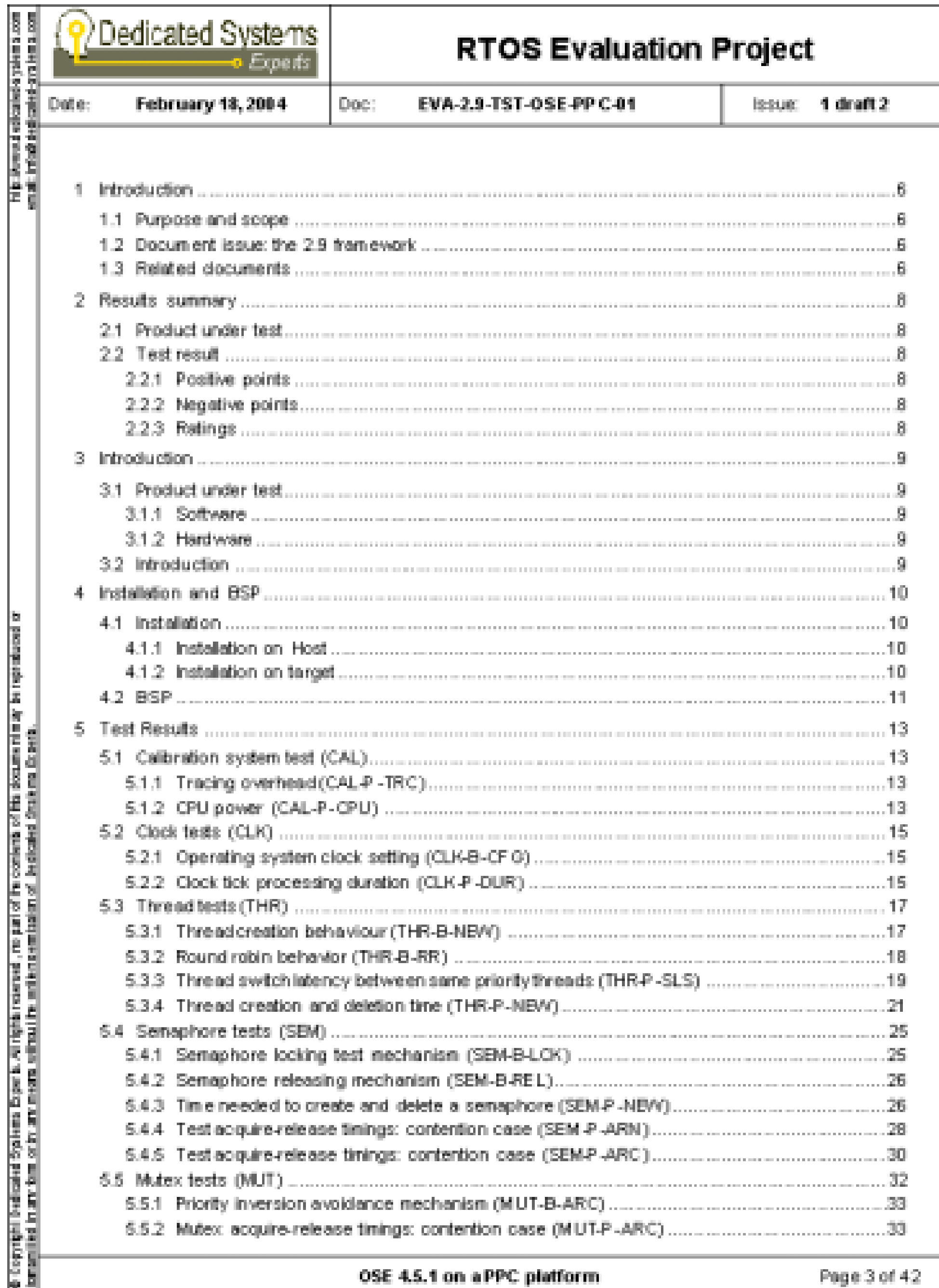
The purpose of our test suite is not only to measure the throughput and the responsiveness of the RTOS, but also to test its determinism and behavior, much more important than performance figures.

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

2.2 The evaluation report layout

In the figure below a sample of the table of contents of an evaluation report is shown.



RTOS Evaluation Project

Date: February 18, 2004 | Doc: EVA-2.9-TST-OSE-PPC-01 | Issue: 1 draft 2

1 Introduction ... 6
1.1 Purpose and scope ... 6
1.2 Document issue: the 2.9 framework ... 6
1.3 Related documents ... 6
2 Results summary ... 8
2.1 Product under test ... 8
2.2 Test result ... 8
2.2.1 Positive points ... 8
2.2.2 Negative points ... 8
2.2.3 Ratings ... 8
3 Introduction ... 9
3.1 Product under test ... 9
3.1.1 Software ... 9
3.1.2 Hardware ... 9
3.2 Introduction ... 9
4 Installation and BSP ... 10
4.1 Installation ... 10
4.1.1 Installation on Host ... 10
4.1.2 Installation on target ... 10
4.2 BSP ... 11
5 Test Results ... 13
5.1 Calibration system test (CAL) ... 13
5.1.1 Tracing overhead (CAL-P-TRC) ... 13
5.1.2 CPU power (CAL-P-CPU) ... 13
5.2 Clock tests (CLK) ... 15
5.2.1 Operating system clock setting (CLK-B-CFG) ... 15
5.2.2 Clock tick processing duration (CLK-P-DUR) ... 15
5.3 Thread tests (THR) ... 17
5.3.1 Thread creation behaviour (THR-B-NEW) ... 17
5.3.2 Round robin behavior (THR-B-RR) ... 18
5.3.3 Thread switch latency between same priority threads (THR-P-SLS) ... 19
5.3.4 Thread creation and deletion time (THR-P-NEW) ... 21
5.4 Semaphore tests (SEM) ... 25
5.4.1 Semaphore locking test mechanism (SEM-B-LCK) ... 25
5.4.2 Semaphore releasing mechanism (SEM-B-REL) ... 26
5.4.3 Time needed to create and delete a semaphore (SEM-P-NEW) ... 26
5.4.4 Test acquire-release timings: contention case (SEM-P-ARN) ... 28
5.4.5 Test acquire-release timings: contention case (SEM-P-ARC) ... 30
5.5 Mutex tests (MUT) ... 32
5.5.1 Priority inversion avoidance mechanism (MUT-B-ARC) ... 33
5.5.2 Mutex acquire-release timings: contention case (MUT-P-ARC) ... 33

OSE 4.5.1 on a PPC platform — Page 3 of 42

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

Dedicated Systems Experts

RTOS Evaluation Project

Date: February 18, 2004 | Doc: EVA-2.9-TST-OSE-PPC-01 | Issue: 1 draft 2

1 Introduction 6
1.1 Purpose and scope 6
1.2 Document issue: the 2.9 framework 6
1.3 Related documents 6
2 Results summary 8
2.1 Product under test 8
2.2 Test result 8
2.2.1 Positive points 8
2.2.2 Negative points 8
2.2.3 Ratings 8
3 Introduction 9
3.1 Product under test 9
3.1.1 Software 9
3.1.2 Hardware 9
3.2 Introduction 9
4 Installation and BSP 10
4.1 Installation 10
4.1.1 Installation on Host 10
4.1.2 Installation on target 10
4.2 BSP 11
5 Test Results 13
5.1 Calibration system test (CAL) 13
5.1.1 Tracing overhead (CAL-P-TRC) 13
5.1.2 CPU power (CAL-P-CPU) 13
5.2 Clock tests (CLK) 15
5.2.1 Operating system clock setting (CLK-B-CFG) 15
5.2.2 Clock tick processing duration (CLK-P-DUR) 15
5.3 Thread tests (THR) 17
5.3.1 Thread creation behaviour (THR-B-NEW) 17
5.3.2 Round robin behavior (THR-B-RR) 18
5.3.3 Thread switch latency between same priority threads (THR-P-SLS) 19
5.3.4 Thread creation and deletion time (THR-P-NEW) 21
5.4 Semaphore tests (SEM) 25
5.4.1 Semaphore locking test mechanism (SEM-B-LCK) 25
5.4.2 Semaphore releasing mechanism (SEM-B-REL) 26
5.4.3 Time needed to create and delete a semaphore (SEM-P-NEW) 26
5.4.4 Test acquire-release timings: contention case (SEM-P-ARN) 28
5.4.5 Test acquire-release timings: contention case (SEM-P-ARC) 30
5.5 Mutex tests (MUT) 32
5.5.1 Priority inversion avoidance mechanism (MUT-B-ARC) 33
5.5.2 Mutex acquire-release timings: contention case (MUT-P-ARC) 33

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

OSE 4.5.1 on a PPC platform Page 3 of 42

Figure 1 Content table of example test report

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

2.3 Measurement method

2.3.1 Tracing PCI access cycles

An absolute time reference is required to measure time intervals. Most operating systems include software timers that could be used as a measurement tool. But timers in different operating systems do not necessarily have the same resolution or precision to perform accurate measurements. The use of software timers adds unpredictable overhead to the results because the measurements are performed by the operating system while it executes the test. The use of it may also change the behavior of the system. Furthermore, the measurements based on the software timers are synchronous with the system clock and this is not what we are looking for. Instead of a software timer, DS-Experts uses an external hardware device: a PCI bus analyzer.

During the execution of a test, tracing data is written at a valid PCI bus address before and after the evaluated system operation. Writing the trace to a double word aligned address on a 33MHz PCI bus takes on most platforms six to seven bus-cycles, i.e. 180ns to 210ns (this largely depends on the platform used). The trace is specific to its position in the code, making it possible to identify and follow the execution path of the test during the analysis of the results.

The PCI bus analyzer stores the data written on the PCI bus into its local memory and timestamps it. When the test is completed, the data is downloaded from the PCI bus analyzer to a PC where it can be further processed.

Tracing is done in the test code by issuing *TraceWriteXxx()* calls. There are four types of traces written on the PCI bus:

- A trace to indicate the start of a timing measurement: *TraceWriteBefore*.
- A trace to indicate the end of a timing measurement: *TraceWriteAfter*.
- A trace to check the behavior of a test (state analysis): *TraceWriteData*.
- A trace to indicate an error condition in the generic to RTOS dependent library: *TraceWriteError*

Figure 2 shows the generic performance measurement using the two timing traces: the time difference between the traces “before” and “after” written on the PCI bus gives the execution time for the system operation. As the collected data marks the start and the end of a system operation, it can be used to calculate the desired time intervals. Each test loops until the trace buffer of the PCI analyzer is full (currently this is 32K trace samples or about 16K time samples).

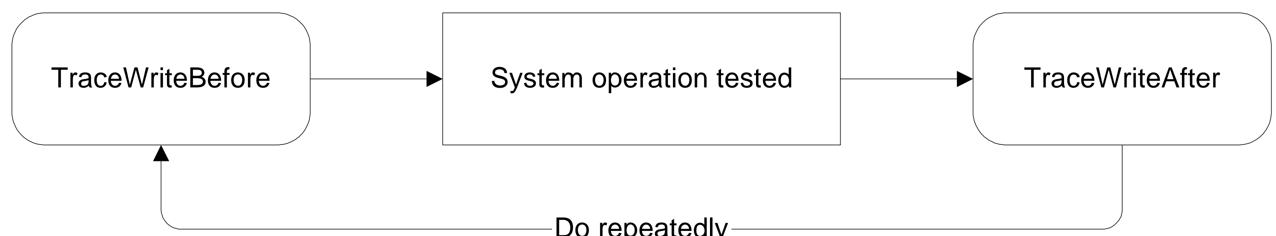


Figure 2 generic performance measurements

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

2.3.2 State analysis

The state traces written during any test performed have a specific identification indicating the state transition that will take place.

Therefore it is possible to do a state analysis on the captured trace: does the system behave as expected? Dedicated Systems Experts designed a tool to verify the captured trace files and to detect anomalies in the expected behavior. This tool improves the verification of a system: such errors would be difficult to notice without. Remark that the trace files can indeed be as large as ten thousands of samples making it almost impossible by hand to detect a fault that only occurs once in a while.

Sometimes multiple valid state changes are possible depending on the features of the system being tested. So the state checker does not only check for a valid trace, it can also determine if the trace is the one expected for a real-time system or not.

Furthermore, the tests are designed to detect invalid state changes and generate an error trace when such a condition is met.

2.3.3 Statistical analysis

The captured trace file, once validated by the state checker, is passed through a statistical program, which generates the diagrams and the “minimum, average and maximum” duration of a certain action.

These values and diagrams are shown in the test evaluation reports.

2.3.4 Generating interrupts

For the interrupt testing series, we use a PCI bus exerciser to generate interrupts at programmable intervals on the PCI bus. To test simultaneous interrupt behavior, two such exercisers are used.

It is important to note that the interrupts are generated by the firmware in the exerciser, this means completely independent of the system under test. This is an extremely important requirement otherwise interrupt generation will always be linked in a way or another to the operating system clock.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

2.4 System configuration parameters

Here we discuss the parameters that remain constant during all tests. They depend on the system configuration. The most important setting concerns the protection model used by the operating system.

2.4.1 Memory Protection model

Memory configurations of an operating system can be classified into the following protection models each being an enhancement of the precedent (see figure 3):

- **No memory protection model**: flat memory layout without memory protection between the threads in the system. This model does not require an MMU.
- **System/User memory protection model**: the system address space is protected from the user address space. User processes and system processes run in a common virtual address space. This model requires an MMU.
- **User/user protection model**: adds protection between user processes to the system/user model. This model requires an MMU.

Independent of the memory protection model, it is also possible to have virtual addressing by the MMU. In this case the physical addresses are mapped onto some virtual address space.

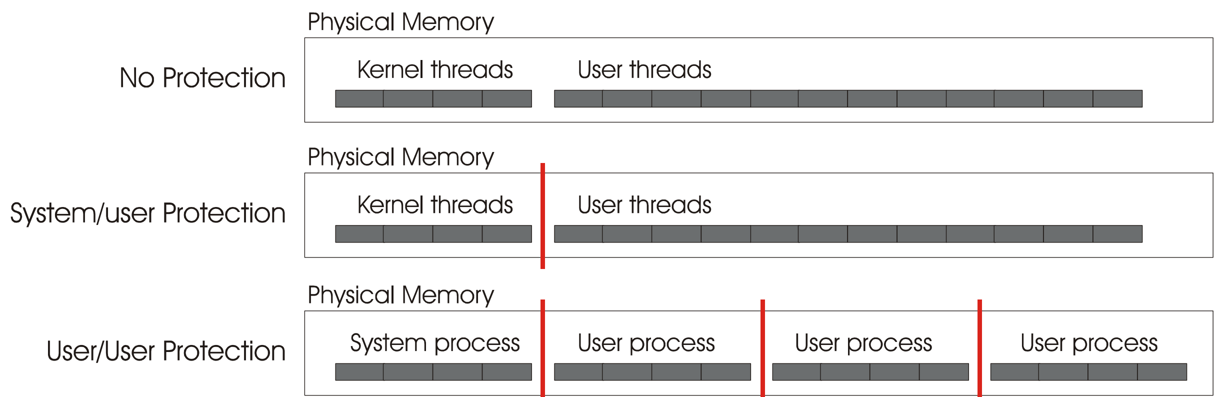


Figure 3 Memory models

Besides the memory protection model, most CPU's have also the notion of privileged CPU instructions that can only be called in some processor mode. This is used to restrict some functionality outside the kernel.

If an OS supports more than one model, some or all tests are redone for the different configurations. In such case, there will be multiple "test results" chapters in the evaluation report: one for each configuration. The same protection configuration is used for all tests published in one chapter.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3 Testing overview

This section of the document describes in detail how tests are identified, how they are compiled and the generic system calls used in the generic test code.

3.1 Naming of the test series

The naming of the different test series is based on multiple identifiers separated with an underscore. We can express this in the Backus-Naur Form (BNF), a formal meta-syntax used to express context-free grammars. A short introduction is given here below.

3.1.1 The Backus-Naur Form

A definition of the symbols used in the BNF notation is shown here:

- "HELLO" "" indicates a string exactly as it should appear
- "A" | "B" | indicates a choice (or)
- <name> <> indicates a definition (type), with the name: name
- <space>::= " " ::= indicates an equality (the definition <space> is equal " ")
- # comment # indicates start of comment (until end of line)

We give an example of a BNF syntax definition and possible valid syntax:

```
<bit>              ::=  "0" | "1"
<binary value>     ::=  <bit> | <bit><binary value>      # recursive example!
```

Valid syntax for <binary value> are then: "001101", "1", "110", ...

3.1.2 Test identifiers

Here we define the BNF used for the test identifiers:

```
<test identifier>        ::=  <main test id> | <main test id> "_" <optional parameters>
<main test id>           ::=  <tested object name> "_" <test class> "_" <test name>
<tested object name>     ::=  # one of the entries as shown in the table below
<test class>             ::=  # one of the entries as shown in the table below
<test name>              ::=  # defined in a table for each tested object name
<optional parameters>    ::=  <parameter> | <parameter> "_" <optional parameters>
<parameter>              ::=  # for each test, the description of the optional parameters
```

An example would be "THR_P_NEW": this would be a performance (P) test on threads (THR), more specific it would test the creation (NEW) time of a thread.

All tests are grouped by object type. We believe that this is the most logical way of ordering the tests.

In the tables below, we define the valid <tested object name> and <test class> used in this document.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

<test object name>	description: test related to
CAL	Calibration test: test used in order to calibrate the platform
CLK	Operating system's internal clock
THR	Threads (in same user space)
SEM	Semaphores
MUT	Mutex = mutual exclusion semaphore. In the scope of the evaluations we differentiate the mutex from a semaphore when a priority inheritance mechanism is involved.
IRQ	Interrupts

<test class>	description
B	Test Behavioral issues
P	Test the Performance of something
S	Stress testing: test the behavior under heavy load conditions

3.1.3 Diagram identifiers

The test result diagram will have the same identifier, but may have an extra parameter if multiple measurements are performed for the same test.

Here we define the name giving for each diagram:

<diagram identifier>	::=	<test identifier> | < test identifier > "_" <measurement type>
<measurement type>	::=	# for some tests a measurement type may be defined

An example would be "THR_P_NEW_DEL": this would be a performance (P) test on threads (THR), more specific it would test the creation/deletion (NEW) time of a thread: results shown in the diagram are the ones of the deletion (DEL) time.

3.1.4 Source code identifier

The source code related with a certain test has the same file name as the test identifier. If the test has optional parameters, the same source code will be compiled (by use of a make tool) in different test executables depending on the optional parameters.

We can illustrate this with an example:

- Say we perform a test with name "X_Y_Z"
- Say that this test has one parameter, a loop quantity "PAR_LOOP"
- Say that we perform this test with the "PAR_LOOP" parameter set to the values 1, 10 and 128 respectively

http://www.dedicated-systems.com
email: info@dedicated-systems.com

The generic source code file would have the name “X_Y_Z.c”. An example of such code file is shown below:

```
void TestEntry(void)
{
   int iLoopCounter = PAR_LOOP;

   while (iLoopCounter > 0)
   {
      DoTheTest();
      iLoopCounter--;
   }
}
```

It is then possible to change the parameter by using compiler command line parameters. All “C” compilers support a command parameter to add a preprocessor macro name, like shown here: “-Dname=value”. This is then used in the make file to differentiate the compilation for different parameter settings.

An example of the make description file for the test above could be:

```
X_Y_Z_1:     X_Y_Z.c
             $(CC) -DPAR_LOOP=1   -o X_Y_Z_1   X_Y_Z.c

X_Y_Z_10:    X_Y_Z.c
             $(CC) -DPAR_LOOP=10  -o X_Y_Z_10  X_Y_Z.c

X_Y_Z_128:   X_Y_Z.c
             $(CC) -DPAR_LOOP=128 -o X_Y_Z_128 X_Y_Z.c
```

The test executables generated in the directory would then be

```
> ls
X_Y_Z_1
X_Y_Z_10
X_Y_Z_128
```

The results of these tests would also have these labels in the test report.

This approach makes it easy to add extra tests with other parameter settings without writing any code. It guaranties that the same code is used again. If these scenarios would be stored in different source files, it would be almost impossible to guarantee consistent code.

The same system may be used for issuing a test in different scenarios. Then the parameter can generate different executables by using the #if, #else, #elif and #endif preprocessor constructions.

We can illustrate this again with an example:

– Say we perform a test with name “X_Y_Z”
– Say that this test has different scenarios set by the “SCENARIO” parameter
– Say that we perform this test with two scenarios: “SC1” and “SC2”

The generic source code file would have the name “X_Y_Z.c”. An example of such code file is shown below:

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

```
/* test scenarios: if invalid used, compiler errors will occur */
#if   SCENARIO==1
# define DoTheTest                Test1
#elif SCENARIO==2
# define DoTheTest                Test2
#endif

void TestEntry(void)
{
   /* init depends on test scenario */
   DoTheTest();
}

void Test1(void)
{
   …
}

void Test2(void)
{
   …
}
```

It is then possible to change the scenario by using again some compiler command line parameters. This is used in the make file to differentiate the compilation for the different scenarios.

An example of a make description file for the test above could be:

```
X_Y_Z_SC1:  X_Y_Z.c
            $(CC) -DSCENARIO=1   -o X_Y_Z_SC1   X_Y_Z.c

X_Y_Z_SC2:  X_Y_Z.c
            $(CC) -DSCENARIO=2   -o X_Y_Z_SC2   X_Y_Z.c
```

The test executables generated in the directory would be

```
> ls
X_Y_Z_SC1
X_Y_Z_SC2
```

The results of these tests would also have these labels in the test report.

Using scenarios makes only sense when the code differences between each scenario are minimal. Otherwise another test name will be used instead.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3.2 Coding style

A coding style is used to clearly identify functions, type definitions and variables and to describe the layout of bracketing and indenting. This should considerably enhance the readability of the code.

Remark that we use strict ANSI C for coding. As a consequence, comments are always written using the "/**/" construct and not the "//" construct.

3.2.1 Identifiers

3.2.1.1 Functions

Functions are written using concatenated words starting with an uppercase for each word, the rest of the characters are written in lower case.

Remark that some of the functions used may also be macro definitions. There is no syntax difference between the two types.

Example:

```
/* write a state trace */
TraceWriteData(5);
```

3.2.1.2 Type definitions

These are written just like the functions as defined before, but a character "t" is added in front of definition.

Example:

```
/* data needed for os dependent part of mutex */
typedef struct tOsMutexData
{
   int *ipMutex;     /* the mutex */
} tOsMutexData;
```

3.2.1.3 Variable declarations

Written just like the functions defined before, but prefix are added in front to indicate the type of the variable. The examples below show how this is done.

Examples:

```
int      iCounter;                            /* an integer */
int     *ipCounter;                           /* a pointer to an integer */
int      iaCounters[10];                      /* an array of integers */
int     *ipaCounters[10];                     /* an array of pointers to integers */

tData   *tpData;                              /* pointer to a type defined structure */

char     cChar;                               /* a character */
unsigned int uiCounter;                       /* an unsigned int */
void    *vpDummy;                             /* a void pointer */
```

So the prefix does not only explains the type, but also how it is used (as pointer, array etc …)

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3.2.1.4 Pre-processor constants

These are written in all uppercase, where each word is separated by an underscore. Prefixes are used to show the origin of the constant.

Examples:

```
TRACE_BUFFER_SIZE                           /* size of the trace buffer in samples */
PAR_QTY_LOOP                                /* number of loops to perform */
```

3.2.2 Bracing styles

In the code the bracing style as shown in the example will always be used.

Examples:

```
/* bracing in a if/else scenario */
if (iX != 0)
{
   Function1();
}
else
{
   Function2();
}

/* bracing in a switch/case scenario */
switch(iX)
{
case 1:
   Function1();

case 2:
   Function2();

default:
   FunctionDefault();
}
```

3.2.3 Indenting

Braces are used to separate code blocks. Each deeper level, the code will be indent with three white spaces.

So no tabs will be used (as this depends on the editor) and a fixed font has to be used (so that each letter has the same width).

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3.2.4 Code blocks and comments

Comments will be placed before the code block or code lines it comments. There will be a blank line before the comment and no blank line after the comment.

A code block (separated by braces) can be used to comment a larger part of the code.

Example:

```
/* handle this */
HandleThis();

/* this block of code does something */
{
   /* do the initialisation */
   InitIt();

   /* now do the real thing */
   DoIt();
}
```

A comment may be put on the same line where an identifier is defined.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3.3 Libraries

As we want to re-use the same generic test code again, independent of the platform under test, libraries are made.

These libraries abstract the tracing system and the operating system. As such the API used in the generic code remains the same, but the implementation may differ.

The different generic API calls used are explained in this section.

3.3.1 Tracing API

All tracing API calls use the "Trace" prefix.

Following constants are defined:

- *TRACE_BUFFER_SIZE*
  Number of samples that can be stored in the trace buffer.

Following API calls are defined:

- *void TraceInitialise(void)*
  This call will initialise the tracing system, it has to be called before any other Trace API call.
- *void TraceCleanUp(void)*
  This call will clean up the tracing system first initialised with TraceInitialise. It has to be called at the end of the test.
- *void TraceWriteData(int iState)*
  This call has to be used for tracing state data. This data can then be analysed by the state analyser to detect invalid behaviour.
  - *int iState*
    Number used to identify the current state.
- *void TraceWriteBefore(int iTimer)*
  This call has to be used for starting a time measurement. Multiple such calls may be used for multiple time measurements at the same time. Therefore the iTimer parameter is used.
  - *int iTimer*
    The identifier of the timer to start.
- *void TraceWriteAfter(int iTimer)*
  This call has to be used to stop a time measurement which was started with the TraceWriteBefore call. Multiple such calls may be used for multiple time measurements at the same time. Therefore the iTimer parameter is used.
  - *int iTimer*
    The identifier of the timer to stop.
- *void TraceWriteError(int iErrno)*
  This call is used to signal an erroneous condition whenever detected. It is also used within the libraries. The trace file analyzer will always detect such traces and declare the test result as invalid.
  - *int iErrno*

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

An identifier signalling the error type.

3.3.2 Interrupt generating API

Interrupts are generated by PCI boards (P-Drive) containing their own processor. On the P-Drive some software is running that may generate PCI interrupts. The test application starts the interrupt generation process by sending a command to the P-Drive (using a PCI mailbox register).

In the current test environment, it is possible to activate two P-Drives for testing interrupt prioritisation.

All the interrupt generating API calls use “IrqGenX” as prefix, where X can be 1 or 2 for the two P-drives used:

– *void IrqGen1Initialise(void)*
  This will initialise the PCI card for generating interrupts.
– *void IrqGen1Start(void)*
  Will start the interrupt generating process.
– *void IrqGen1Disable(void)*
  Will stop the interrupt generating process.
– *void IrqGen1AckInterrupt(void)*
  Will acknowledge the interrupt and clear the interrupt source.
– *int IrqGen1GetVector(void)*
  This will return the interrupt vector in the operating system used for this device.

Above calls exist also with the “IrqGen2” prefix.

3.3.3 Generic operating system API

These API calls abstract the operating system used, otherwise it would not be possible to write portable generic testing code.

All operating system API calls start with the “Os” prefix followed by the object related to the call. For instance, all thread related calls shall have the prefix “OsThread”.

As not only the API calls may differ between the different operating systems, but also the object data, the operating system object data will be hidden in a type definition with the name “tOsObjData”.

To avoid complex handling in the library that would delay the effective system-call and generate measurement overhead, most of these API calls will be macros. In general, the complex system calls (like creating and starting a thread) will be split in two parts:

– An “OsObjCreate” call, that will be used to set-up an operating system dependent structure. This call will be handled by a real function and may take some time.
– An “OsObjXxx” call, that will be a macro using the operating system dependent structure already set-up by the previous call so the extra overhead will be minimum.

Both the “OsObjCreate” and “OsObjDelete” calls are used only for this purpose and are never measured in the scope of this framework.

In the next sections the API calls for each object type are explained in detail.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

3.3.3.1 Threads

All API calls concerned with this library use the "OsThread" prefix

Following data structure is defined:

- *tOsThreadData*
  A data structure containing all operating system dependent data for accessing threads and the related data. This is used to hide the operating system internals.

Following constants are defined:

- *OS_PRIORITY_HIGHEST*
  Highest priority an application thread can use in the system.
- *OS_PRIORITY_HIGH*
- *OS_PRIORITY_MIDDLE*
- *OS_PRIORITY_LOW*
- *OS_PRIORITY_LOWEST*
  Lowest priority an application thread can use in the system.
  Remark that the exact priority is of no importance, only the RELATIVE priority is relevant.

Following API calls are defined:

- *int OsPriorityIncrement(int iPriority)*
  This call is used to increment the priority: increment means in this context to higher the priority level, which is not the same as setting the priority variable to a higher number!
  - *return: int*
    The incremented priority.
  - *int iPriority*
    The priority to increment.
- *int OsPriorityDecrement(int iPriority)*
  This call is used to decrement the priority.
  - *return: int*
    The decremented priority.
  - *int iPriority*
    The priority to decrement.
- *void OsThreadSetMyPriority(int iPriority)*
  This call is used to change the priority of the current active thread.
  - *int iPriority*
    The new priority of the calling thread after executing this system call.
- *tOsThreadData *OsThreadCreate(int iPriority, void (*Function)(void*), void *vpArgument)*
  This call is used to create the operating system dependent data for starting, stopping the thread. Remark that in scope of this framework priority based scheduling will always be used.
  - *return: tOsThreadData**
    The operating system dependent data.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

- *int iPriority*

  The default priority that the thread of execution will have when started.

- *void (*Function)(void*)*

  The entry function called when the thread starts.

- *void *vpArgument*

  The argument passed to the entry function of the thread when the thread starts. Remark that some operating systems do not have an ability to pass arguments to a thread. In such case a more complex library will be needed to pass arguments to a starting thread. Without arguments our generic tests cannot run.

- *void OsThreadStart(tOsThreadData *tpThread)*

  This call is used to start the execution thread already initialised by the OsThreadCreate call.

  - *tOsThreadData* tpThread*

    The operating system dependent thread data for the thread to start.

- *void OsThreadStop(tOsThreadData *tpThread)*

  This call is used to stop the execution thread started by the OsThreadStart call.

  - *tOsThreadData* tpThread*

    The operating system dependent thread data for the thread to stop.

- *void OsThreadDelete(tOsThreadData *tpThread)*

  This call is used to clean up the structure allocated and initialised by the OsThreadCreate call.

  - *tOsThreadData* tpThread*

    The operating system dependent thread data to clean up. After this call, the thread data will be invalid.

- *void OsThreadYield (void)*

  This call is used to yield the CPU to another ready thread at the same priority level (if any).

- *void OsThreadSleepSeconds(int iSeconds)*

  This call is used to delay the current active thread for some seconds (non-busy blocking wait).

  - *int iSeconds*

    The number of seconds the executing thread should wait.

- *void OsThreadSleepOneTick(void)*

  This call is used to delay the current active thread until next clock tick (block until next OS tick).

3.3.3.2 Semaphores

All API calls concerned with this library use the “OsSem” prefix

Following data structure is defined:

- *tOsSemData*

  Data structure containing operating system dependent data for accessing semaphores and related data. This structure is used to hide operating system internals.

Following API calls are defined:

- *tOsSemData *OsSemCreate(unsigned int uiInitialCount)*

  This call is used to create the operating system dependent data.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

  - *return: tOsSemData**
    Operating system dependent data.
  - *unsigned int uiInitialCount*
    Initial count of the semaphore.
- *void OsSemInit(tOsSemData *tpSemaphore)*
  This call is used to initialise the semaphore with its initial value.
  - *tOsSemData *tpSemaphore*
    Operating system dependent data.
- *void OsSemP(tOsSemData *tpSemaphore)*
  This call is used to try and take the semaphore.
  - *tOsSemData *tpSemaphore*
    Operating system dependent data.
- *void OsSemV(tOsSemData *tpSemaphore)*
  This call is used to release the semaphore.
  - *tOsSemData *tpSemaphore*
    Operating system dependent data.
- *void OsSemDestroy (tOsSemData *tpSemaphore)*
  This call is used to destroy the operating system semaphore initialised by the OsSemInit call.
  - *tOsSemData *tpSemaphore*
    The semaphore to destroy.
- *void OsSemDelete (tOsSemData *tpSemaphore)*
  This call is used to clean up the structure allocated and initialised by the OsSemCreate call.
  - *tOsSemData *tpSemaphore*
    The operating system dependent semaphore data to clean up. After this call, the semaphore data will be invalid.

3.3.3.3 Mutex

All API calls concerned with this library use the “OsMutex” prefix

Following data structure is defined:

- *tOsMutexData*
  Data structure containing all operating system dependent data for accessing mutexes and related data. This is used to hide operating system internals.

Following API calls are defined:

- *tOsMutexData *OsMutexCreate(void)*
  This call is used to create the operating system dependent data to use a mutex.
  - *return: tOsMutexData**
    Operating system dependent data.
- *void OsMutexInit(tOsMutexData *tpMutex)*
  This call is used to initialise the operating system mutex object.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

  - *tOsMutexData *tpMutex*
    Operating system dependent data.
- *void OsMutexP(tOsMutexData *tpMutex)*
  This call is used to try and take the mutex.
  - *tOsMutexData *tpMutex*
    The mutex to take.
- *void OsMutexV(tOsMutexData *tpMutex)*
  This call is used to release the mutex.
  - *tOsMutexData *tpMutex*
    The mutex to release.
- *void OsMutexDestroy (tOsMutexData *tpMutex)*
  This call is used to destroy the operating system mutex initialised by the OsMutexInit call.
  - *tOsMutexData *tpMutex*
    The mutex to destroy.
- *void OsMutexDelete (tOsMutexData *tpMutex)*
  This call is used to clean up the structure allocated and initialised by the OsMutexCreate call.
  - *tOsMutexData *tpMutex*
    The operating system dependent mutex data to clean up. After this call, the mutex data will be invalid.

3.3.3.4 Interrupts

Remark that for some operating systems it is not possible to use the generic interrupt handling test code. In such case specific custom interrupt test software is written. These tests still follow the structure of the generic tests.

All API calls concerned with this library use the “OsIrq” prefix.

Following data structure is defined:

- *tOsIrqData*
  Data structure containing OS specific data.

Following API calls are defined:

- *tOsIrqData* OsIrqCreate(int vector, void(*isr)(void*), void *arg)*
  This API call will create and set-up the tOsIrqData structure.
  - *Int vector*
    Interrupt vector to connect interrupt handler on (use the IrqGenXGetVector() call to get this from the interrupt generating library).
  - *void(*isr)(void*)*
    The interrupt handler to be called from the interrupt.
  - *void *arg*
    Argument to be passed to the interrupt handler.
- *void OsIrqDelete(tOsIrqData* Irq)*

This API call will clean-up and free the tOsIrqData structure.

- *tOsIrqData* Irq*
  Data structure containing OS specific data.

- *void OsIrqEnable(tOsIrqData* Irq)*
  API call used to enable the interrupt.
  - *tOsIrqData* Irq*
    Data structure containing OS specific data.
- *void OsIrqDisable(tOsIrqData* Irq)*
  API call to disable the interrupt.
  - *tOsIrqData* Irq*
    Data structure containing OS specific data.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

4 The tests described

4.1 Calibration system test (CAL)

These tests are used to calibrate the tracing overhead in comparison with the processing power of the platform. This is important to understand the accuracy of the measurements done in scope of this report.

Here the minimum time between two traces is measured, which shows the time duration of taking a sample. The results in the evaluation report are the trace timestamps decremented with this value.

To detect how much the impact is of the tracing system in comparison with CPU performance a second test is done. In this test, the loop time is measured and compared with the tracing overhead.

In short:

<test name>	description
P_TRC	Measure the tracing overhead
P_CPU	Measure the CPU performance in comparison with the tracing overhead

4.1.1 Tracing overhead (CAL_P_TRC)

This test will calibrate the tracing system overhead. The result found will not only be used to have an idea of the overhead but also to depict the accuracy of the measurements. If the trace delay is stable and known, then the measurement accuracy will be better than with an unstable delay. Therefore, the test loop will disable interrupts during the two traces to avoid any influence from the platform on the time measurements. For this the critical section operating system calls are used.

In the rest of the report, the tracing overhead will be subtracted from the results obtained.

4.1.1.1 Test Parameters

None

4.1.1.2 Measurements done

Following times are measured during the test:

- Tracing overhead.

4.1.1.3 Test results

Results will be shown in a table as shown below:

Test	result
Average tracing overhead	In nsec
Minimum tracing overhead	In nsec
Maximum tracing overhead	In nsec

http://www.dedicated-systems.com
email: info@dedicated-systems.com

Test	result
Tracing accuracy	In nsec
Critical section primitive present?	YES or NO

4.1.2 CPU power (CAL_P_CPU)

This test will calibrate the CPU performance and the memory bandwidth of the platform being used. This test does measurements with the same code in 2 cases: cached (loop) or not cached (un-looped) code and data. As such the effects of the cache can be calculated and performance of platforms can be compared with our standard platform (Pentium MMX 200 MHz platform).

4.1.2.1 Test Parameters

None

4.1.2.2 Measurements done

- Duration of CPU test loop (cached and not cached)
- Duration of memory test loop (cached and not cached)

4.1.2.3 Test results

Caching effect:

Test	no cache	cached	cache effect
CPU test duration			
MEM test duration			
Average caching effect (CPU and MEM)			

The same test on our standard platform (Pentium MMX 200 MHz):

Test	no cache	cached	cache effect
CPU test duration	401.9 us	270.8 us	1.48
MEM test duration	5.442 ms	1.512 ms	3.60
Average caching effect (CPU and MEM)			2.54

Performance compared with our standard platform, larger values mean faster:

Test	no cache	cached
CPU performance factor		
MEM performance factor		

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.2 Clock tests (CLK)

The clock test measures the time an operating system needs to handle its clock interrupt. Some platforms (like the x86 motherboard) have the clock interrupt on the highest system interrupt. This is certainly not the best choice for real-time systems. In such a case, a clock interrupt showing up during a testcycle will cause a delay in the measurement for that cycle. If any test takes multiple operating system clock cycles to finish, than these spikes will be seen in the test results.

This is the reason why this test is the first run in our test bench. The table below shows the different tests done:

<test name>	description
B_CFG	Test the internal clock period setting
P_DUR	Test the clock interrupt processing duration

4.2.1 Operating system clock setting (CLK_B_CFG)

This will test the setting of the clock tick in the operating system. In our tests we use the default setting from the OS vendor. This test verifies the setting, or detects the clock tick timing if it is not settable.

4.2.1.1 Test Parameters

None

4.2.1.2 Measurements done

None

4.2.1.3 Test results

These test results only show the clock tick time. They are not used to validate a system as being real-time. If the test would not behave as expected (e.g. sleep behavior problem) than the clock time is measured by other means.

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Tested clock period	Time measured in msec
Clock period adaptable	YES or NO

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.2.2 Clock tick processing duration (CLK_P_DUR)

This will test the clock tick processing duration in the kernel. The test can be parameterized to detect any impact of the system load on the clock processing time.

This test will have only one application thread running. This application performs busy loops. The time needed for each busy loop is measured and expected to always be the same. However, if during the loop a clock interrupt occurs, the busy loop will be interrupted for some time. The differences between the longer busy loop duration and the normal busy loop duration can only be caused by the clock interrupt, as al other interrupts are turned off during this test.

The test results are extremely important, as the clock interrupt will disturb all other measurements done in this framework.

It may be possible to disable the clock interrupt for some operating systems and platforms. Of course, then some other OS features may cease functioning such like “perform any delay” or “waiting with timeout”. Tests of these are then impossibleest.

4.2.2.1 Test Parameters

None

4.2.2.2 Measurements done

– Clock duration time

4.2.2.3 Test results

These test results show the clock tick processing duration. The test results are shown in a diagram together with a table containing the normal busy loop time and the loop time when a clock interrupt occurred.

The loop counter used in the test loop has to be chosen so that:

– Enough loop samples including a clock interrupt are gathered (some hundreds)

– Time can be measured accurately.

With long loops, more samples are generated, but the timing accuracy of the samples becomes less (timing is done with a 16-bit exponential timing system: limited bit size of mantissa!).

Test	result
CLOCK_LOOP_COUNTER	The value set for the test loop.
Normal busy loop time	Time in µs of the busy loop when parameter above is used for this test.
Busy loop time with clock interrupt	Time in µs of the busy loop when an clock irq occurred.
Clock interrupt duration	Difference between the two values above.

Diagrams:

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

- Clock duration time

4.3 Thread tests (THR)

Thread tests shows the behavior of the scheduler. What is the thread switch latency? Does it depend on system load? What's the time to create/delete a thread? Are the priorities handled well? These are some of the questions we solve with the Thread tests.

The table below shows the different tests done:

<test name>	description
B_NEW	Thread creation behavior: when creating a thread with lower, same or higher priorities how are these scheduled?
B_RR	Test checks the Round Robin scheduling of threads at the same priority level.
P_SLS	Thread switch latency performance between same priority threads.
P_NEW	Thread creation and deletion time.

4.3.1 Thread creation behaviour (THR_B_NEW)

This will test the thread creation behavior. Does the operating system behave as a real-time operating system should behave?

This test checks following issues when generating a thread:

- When creating a thread with a lower priority than the creating thread, following rule applies: "The new thread shall not run while the creating thread is active".
- When creating a thread with the same priority than the creating thread, following rule applies: "The new thread should not be activated immediately; it should be put at the tail of the ready thread queue".
- When yielding the processor, another thread in the same priority FIFO queue (if any) shall run.
- When creating a thread with a higher priority than the creating thread, following rule applies: "The new thread shall preempt the creating thread and become active immediately.
- When a thread lowers its priority below the priority of another thread that is in the ready-to-run state, then the thread shall be preempted and the highest priority ready-to-run thread shall be activated.

If these tests fail, most other thread tests will not be able to run.

4.3.1.1 Test Parameters

None

4.3.1.2 Measurements done

None

4.3.1.3 Test results

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

The data trace of this test should be:

```
/* normal behavior */
STATE_START
STATE_YIELDING
STATE_MIDDLE_ACTIVE
STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE
STATE_END
```

Also no error trace may be generated, otherwise the test invalidates the OS as being RT!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Lower priority not activated?	OK or FAILED
Same priority at tail?	OK or FAILED
Yielding works?	YES or NO
Higher priority activated?	OK or FAILED

Diagrams: none

4.3.2 Round robin behaviour (THR_B_RR)

This test checks if the scheduler uses a fair round robin mechanism when threads are having the same priority and all are in the ready-to-run state!

All threads are in the ready-to-run state so on each operating system clock tick (end of time slice) the round robin mechanism should schedule the next thread in the ready queue and store the current thread at the end of the FIFO queue.

Remark that not all operating systems use round robin scheduling between threads running at the same priority. As this feature is not needed for guarantying real-time behavior, an OS without this feature can still receive the “RT-VALIDATED” logo.

4.3.2.1 Test Parameters

– PAR_THREADS_QTY: number of threads that will be used in this test, the number of threads that have to be scheduled in the FIFO
  In normal circumstances, this test is run only with PAR_THREADS_QTY set to 10. If anomalies are detected, other values than 10 may be used trying to understand the bad behavior.

4.3.2.2 Measurements done

None

4.3.2.3 Test results

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

If the operating system uses a fair round robin system for scheduling ready-to-run threads of the same priority, then each clock tick a trace will show up. Also the order of the trace is important: the thread with ID PAR_THREADS_QTY will be the first followed by decreasing thread IDs until zero is reached. Then it should restart the same scenario all over again.

When there is no fair scheduling it can occur that a thread will loop endlessly. After some time the main thread will higher its priority to stop the test in all cases.

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Time slice following this test	Time slice in ms

Diagrams: none because it deals with behavior

4.3.3 Thread switch latency between same priority threads (THR_P_SLS)

This test will measure the time to switch between threads of the same priority. Therefore the "voluntary_yield_processor_to_other_thread" system call is used. If the THR_B_NEW test fails, then this test cannot be run.

A number of threads at the same priority will be generated. Each thread will yield another thread. These threads are handled in round-robin mode (if supported by the OS).

As for a voluntary yield, only the ready-to-run queue of the same priority level has to be checked, this test may have better results than the switch latency test for different priorities.

The aim here is to check the FIFO queuing mechanism: this should not depend on the number of threads in the FIFO: otherwise the scheduler is has not a predictable behavior.

4.3.3.1 Test Parameters

– PAR_THREADS_QTY: number of threads that will be used in this test, the number of threads that have to be scheduled in the FIFO
  This test is run with PAR_THREADS_QTY set to:
  – 2
  – 10
  – 128

4.3.3.2 Measurements done

Test	Sample qty	Avg	Max	Min
Thread switch latency, 2 threads				
Thread switch latency, 10 threads				
Thread switch latency, 128 threads				

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.3.3.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO

Diagrams:

- Thread switch latency between two threads.
- Thread switch latency between ten threads.
- Thread switch latency between 128 threads.

Remark that for the OS to be predictable, the number of threads in the ready queue may not have an impact on the switch latency measured.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.3.4 Thread creation and deletion time (THR_P_NEW)

This will test the time to create a thread and the time to delete a thread. Different scenarios are possible, from which the following are tested here:

- Scenario “NER” (NEver Run): The created thread has a lower priority than the creating thread and is deleted before it had any change to run: in this test no thread switch occurs.
- Scenario “RTE” (Run and TErminate): The created thread has a higher priority than the creating thread and activates. The created thread immediately terminates itself (thread does nothing).
- Scenario “RNT” (Run, but does Not Terminate): The same scenario as above, but the created thread does not terminate (it lowers it’s priority when it is activated).

In the scenarios “RTE” and “RTN”, the creation time is the duration from the system call creating the thread to the time when the created thread activates. For the “NER” scenario the creation time is the duration of the system call.

Remark that this test cannot run if the THR_B_NEW test failed!

4.3.4.1 Test Parameters

- SCENARIO: test scenario selected
  - SC1: NER, Never run
  - SC2: RTE, Run and terminate
  - SC3: RNT, Run, but do not terminate

4.3.4.2 Measurements done

Test	Sample qty	Avg	Max	Min
Thread creation, never run				
Thread deletion, never run				
Thread creation, run and terminate				
Thread deletion, run and terminate				
Thread creation, run and block				
Thread deletion, run and block				

4.3.4.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

Test	result
Test succeeded	YES or NO

Diagrams:

- Thread creation time in the “NER” scenario (duration of system call).
- Thread deletion time in the “NER” scenario.
- Thread creation time in the “RTE” scenario (duration of system call start to activated thread).
- Thread deletion time in the “RTE” scenario.
- Thread creation time in the “RNT” scenario (duration of system call start to activated thread).
- Thread deletion time in the “RNT” scenario.

http://www.dedicated-systems.com
email: info@dedicated-systems.com
© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.4 Semaphore tests (SEM)

This is testing the performance and the behavior of a counting semaphore. The counting semaphore is a system object that protects for simultaneous accesses to some device or resource. This is well known as the Dijkstra paradigm. A semaphore object has:

- A semaphore counter
- A P() function which tries to acquire the semaphore (from the Dutch language “Probeer” = Try). If the counter is zero, this system call will block the calling thread until the semaphore is released by another thread. If the semaphore counter is not zero, the counter will decrement and the thread continues.
- A V() function which releases the semaphore (from the Dutch language “Vrij” = Release). This will increment the semaphore counter.

Remark that in scope of this test, only protection semaphores between threads belonging to the same process are tested.

In most operating systems, there exists system calls for a simpler version of the counting semaphore object: where the counter can only be zero or one (acquired or free). In the next section we discuss such an object that we will call the “MUTEX” (MUTual EXclusive semaphore).

The table below shows the different tests done for this object:

<test name>	description
B-LCK	Semaphore protection behavior.
B-REL	Verifies that blocked thread with highest priority activates on a release operation.
P-NEW	Semaphore creation and deletion time.
P-ARC	Acquire and release time in contention case
P-ARN	Acquire and release time in no contention case

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.4.1 Semaphore locking test mechanism (SEM_B_LCK)

This will test if the counting semaphore locking mechanism works as expected. The P() call should block only when the count is zero. The V() call should increment the semaphore counter. In the case the semaphore counter is zero, the V() call should cause a rescheduling in the kernel: indeed blocked threads may be activated.

The aim of this test is to detect the good behavior of the the counter functions.

4.4.1.1 Test Parameters

None

4.4.1.2 Measurements done

Only state behavior is measured.

4.4.1.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo! In case where the test fails in a way that the semaphore cannot guaranty some protection over shared objects, then the operating system will receive the DID_NOT_QUALIFY logo.

The data trace of this test should be:

```
/* normal behavior */
STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE
STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE
STATE_HIGH_ACTIVE
```

Also no error trace may be generated, otherwise the test invalidates the OS as being RT!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Maximum semaphore value?	Not tested: from documentation
Rescheduling on free?	OK or FAILED

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.4.2 Semaphore releasing mechanism (SEM-B-REL)

This test verifies that the highest priority thread being blocked on a semaphore will be released by the release operation. This should be independent of the order of the acquisitions taking place.

Therefore the test is run in two scenarios:

- Where the highest priority thread acquires first the semaphore (called the scenario “HI”)
- Where the lowest priority thread acquires first the semaphore (called the scenario “LOW”)

The release result of both scenarios should be the same: the highest priority thread should be activated upon the release.

4.4.2.1 Test Parameters

- SCENARIO: test scenario selected
  - 1: “HI” Highest priority thread acquires first.
  - 2: “LOW” Lowest priority thread acquires first.

4.4.2.2 Measurements done

Only state behavior is measured

4.4.2.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo! However, the semaphore can still be used to protect shared data and devices.

The data trace of this test should be:

```
/* behavior in "HI" */                /* behavior in "LOW" */
STATE_LOW_ACTIVE                      STATE_LOW_ACTIVE
STATE_HIGH_ACTIVE                     STATE_MIDDLE_ACTIVE
STATE_MIDDLE_ACTIVE                   STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE                      STATE_LOW_ACTIVE
STATE_HIGH_ACTIVE                     STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE                      STATE_LOW_ACTIVE
STATE_MIDDLE_ACTIVE                   STATE_MIDDLE_ACTIVE
STATE_LOW_ACTIVE                      STATE_LOW_ACTIVE
```

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

4.4.3 Time needed to create and delete a semaphore (SEM_P_NEW)

This will test the time needed to create a semaphore and the time to delete it. The deletion time is checked in two cases:

- Where the semaphore is used between the creation and deletion (the “USE” scenario).
- Where the semaphore is not used between the creation and deletion (the “DUM” or dummy scenario).

For a good real-time operating system it is expected that there is no difference between the two scenarios. If a difference is detected, then this probably means that the operating system handles some initializations on the semaphore on its first use (making the first use slower, which is not desirable in a RT system).

4.4.3.1 Test Parameters

- SCENARIO: test scenario selected
  - 1: “USE” semaphore used.
  - 2: “DUM” semaphore not used: thus a dummy semaphore.

4.4.3.2 Measurements done

Following times are measured during the test:

Test	Sample qty	Avg	Max	Min
Semaphore creation time, used				
Semaphore deletion time, used				
Semaphore creation time, never used				
Semaphore deletion time, never used				

4.4.3.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO

Diagrams:

- Semaphore creation time, used.
- Semaphore deletion time, used.
- Semaphore creation time, never used.
- Semaphore deletion time, never used

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.4.4 Test acquire-release timings: no contention case (SEM_P_ARN)

This tests the acquisition and release time in the no contention case. As in this test case the semaphore does not block nor causes any rescheduling (thread switch), the duration of the system call should be very short.

In fact, the OS will only need to increase or decrease the semaphore counter in an atomic way.

4.4.4.1 Test Parameters

None

4.4.4.2 Measurements done

Following times are measured:

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, no contention				
Semaphore release time, no contention				

4.4.4.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO

Diagrams:

- Semaphore acquisition time, no contention.
- Semaphore release time, no contention.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

4.4.5 Test acquire-release timings: contention case (SEM_P_ARC)

This is used to test the time needed to acquire and release a semaphore depending on the number of threads blocked waiting for the semaphore. It measures only the time in the contention case: this means when the acquisition and release system call causes a rescheduling to occur.

The aim of this test is to verify if the number of blocked threads waiting for the semaphore has an impact on these timings. So this will answer the question: “how much time the operating system needs to find out the next thread to schedule?”.

This test is very important to detect how predictable the operating system is depending on the number of blocked threads in the wait for semaphore list. When a good search algorithm is used the impact should be small. This test also figures out where most time is spend in ordering the threads:

- During acquisition: when the thread gets in the blocked state.
- During release: when the thread gets in the read-to-run state.

This test will be done with a number of threads equal to the number of priorities available in the operating system with a maximum of.

Remark that we are testing the contention case: so each measurement will always include the thread switch latency!

4.4.5.1 Test Parameters

- None

4.4.5.2 Measurements done

Following times are measured:

Test	Sample qty	Avg	Max	Min
Semaphore acquisition time, contented				
Semaphore release time, contented				

4.4.5.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Max number of threads pending	128 or number of priority levels if less than 128

Diagrams:

- Semaphore acquisition time, contended.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

– Semaphore release time, contended.

Most of the time, a zoom-in diagram will be shown to see the influence on the number of blocked threads.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.5 Mutex tests (MUT)

Here the performance and the behavior of the mutual exclusive semaphore are tested.

Although the mutual exclusive semaphore (further called mutex) could be the same as the counting semaphore where the count is one, this is not the aim of this test to copy the precious described tests. In the scope of the framework, this test will look into the details of a mutex system object that avoids priority inversion.

Details about the priority inversion situation can be found in [Doc. 2].

Different mechanisms exists to avoid a priority inversion scenario, most RTOS use one of these:

- Priority inheritance: the blocking thread will inherit the priority of the blocked thread.
- Priority ceiling: the priority of the blocking thread will be set to a high fixed (ceiling) priority.

In scope of this framework, it does not matter how the operating system avoids priority inversion. It only detects if such a system actually does prevent the priority inversion.

If the operating system does not has such a mechanism, then this section will be skipped (tests will not be done).

The tests in this section will also detect how much time it takes to deal with the priority inversion avoidance mechanism.

The table below shows the different tests done for this object:

<test name>	description
B_ARC	Acquisition en release behavior in the contention case with priority inversion. Does the system call really avoid the priority inversion case?
P_ARC	Acquire and release time in contention case with priority inversion.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.5.1 Priority inversion avoidance mechanism (MUT-B-ARC)

This test will determine if the system call under test prevents the priority inversion case. Therefore the test will artificially create a priority inversion.

The flow chart, with the expected execution path is shown in the figure below (execution path in light green). The important steps in the execution flow are:

- 1: The main test execution thread, which runs at low priority will create two other threads.
  - A high priority thread, that will start execute immediately. This thread will block on the semaphore "high".
  - A middle priority thread, with the priority between the creating and the high level thread. Also this thread will block, now on the semaphore "middle".
- 2: The main thread will acquire the mutex (so the critical section lock starts there).
- 3: The main thread will release the semaphore "high" so the high priority thread activates. This simulates an external event in a real system.
- 4: The high level thread also acquires the mutex: as the mutex is taken, the high priority thread blocks, and the low level priority activates again.
- 5: This is the crucial point: the low level thread activates the middle level thread (by releasing the middle semaphore). If priority inversion protection is enabled, then the middle level thread will NOT activate! Instead, the low-level priority thread has inherited the high level priority of the blocked high level thread (or received the mutex ceiling priority). As this priority is now higher than the middle priority, the low level thread continues!
- 6: The low level thread comes at the end of the critical section and releases the mutex. At that moment the priority of the thread is restored and the operating system will schedule the high level thread when the lock is released.

The rest of the flow is straightforward.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

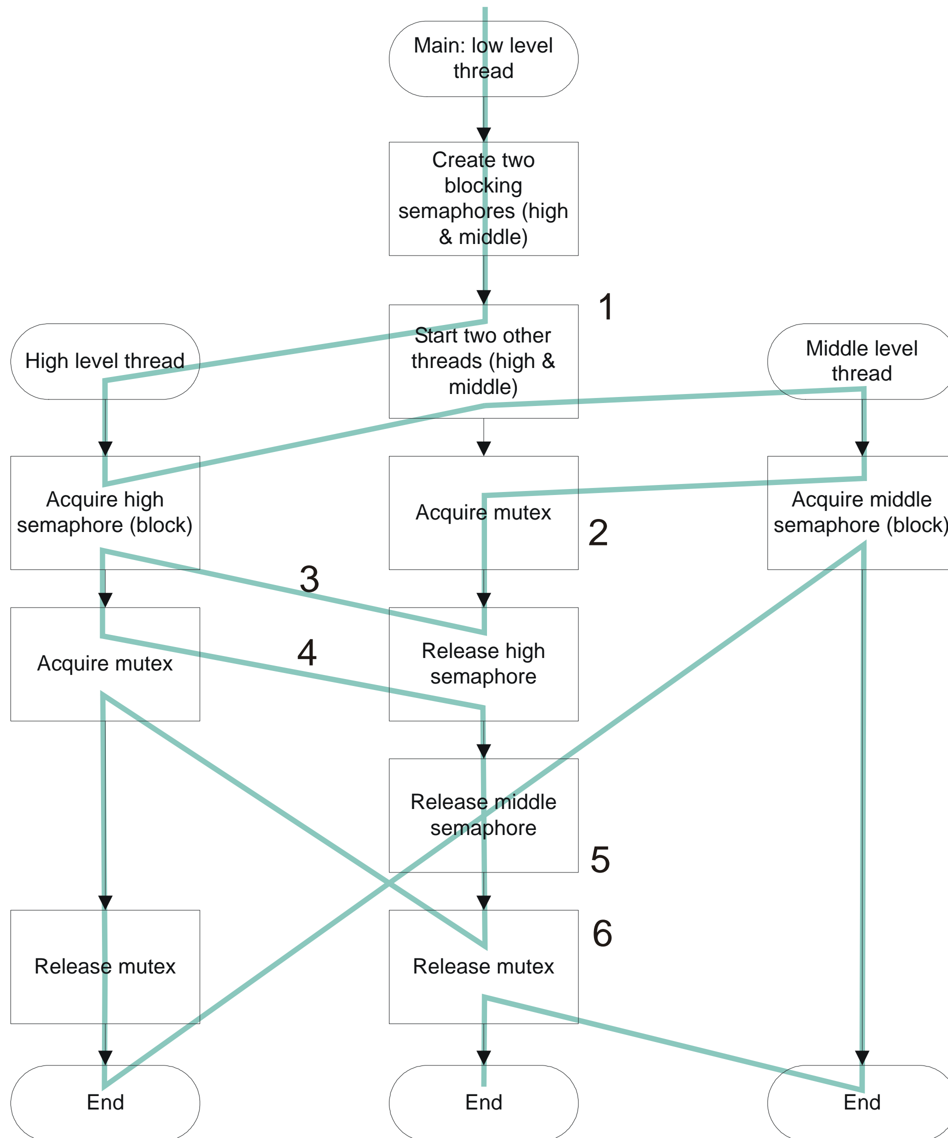


Figure: Priority inversion avoidance

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

When there is no priority inversion avoidance mechanism, the flow of the test will be as shown in the following diagram. In that case the middle level thread will be activated first!

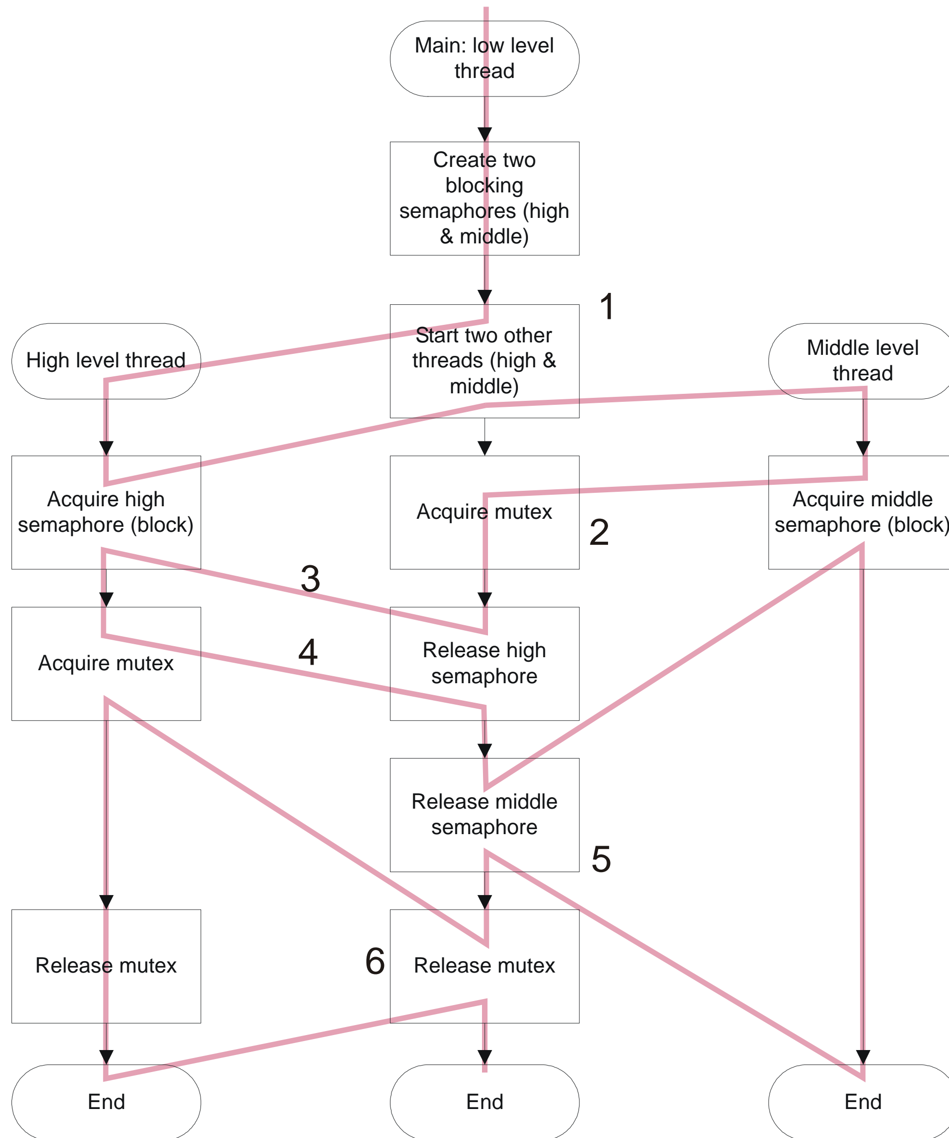
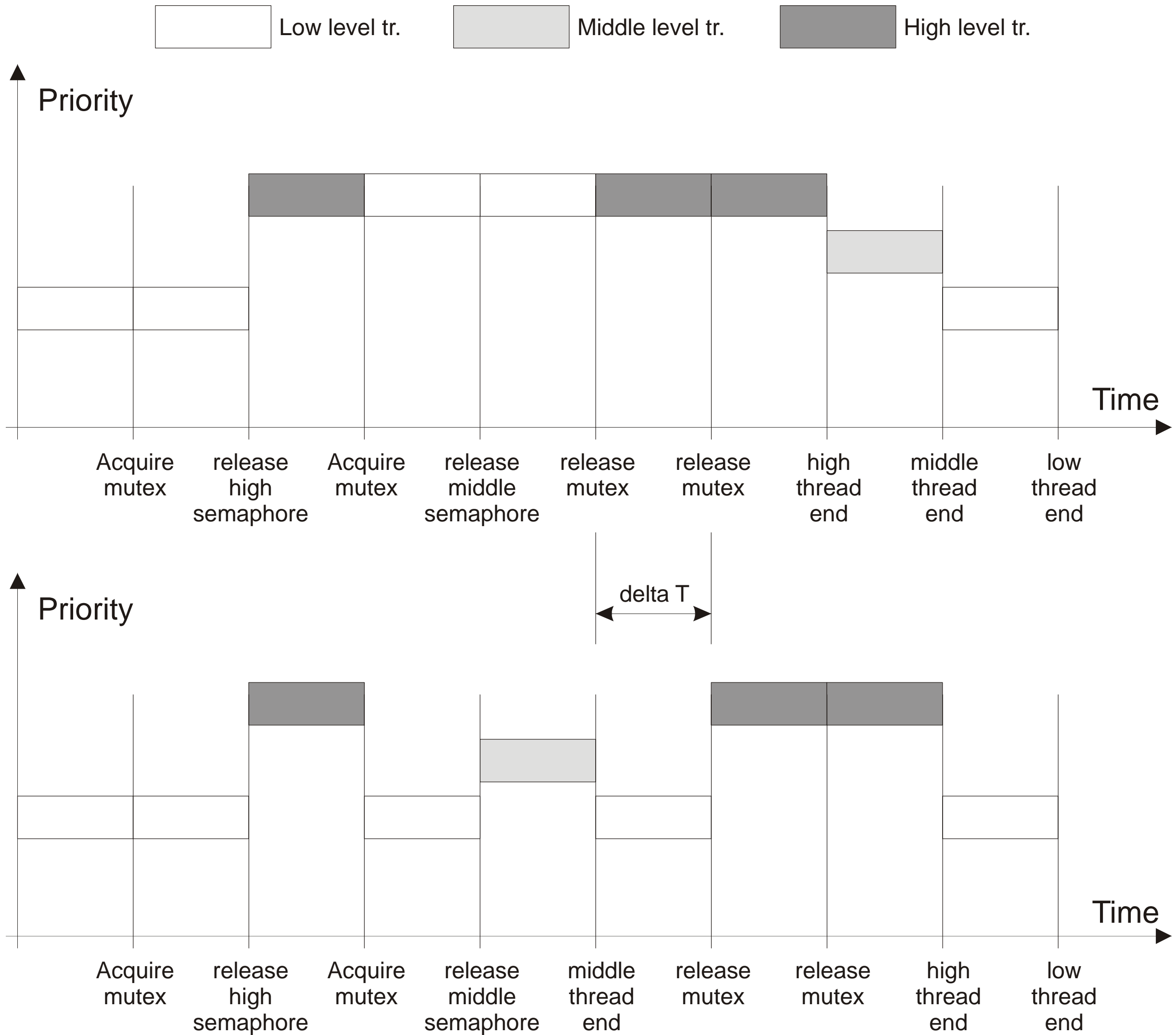


Figure: No priority inversion avoidance

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

In the figure below, the timing difference between the two scenarios is shown. It is clear that priority inversion causes an extra delay (delta T) for the high level thread. As a result, the system becomes less predictable. Do not forget that in a real live situations the middle priority thread can be active for a long time!



Timing diagrams

Therefore, an operating system that does not have any system call to avoid a priority inversion case will not receive the RT-VALIDATED logo!

4.5.1.1 Test Parameters

None

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.5.1.2 Measurements done

None

4.5.1.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo! However, it still can receive the VALIDATED logo if the locking of the mutex is usable for disabling mutual access to some part of the code.

The data trace of this test should be:

```
/* RT-VALIDATED */              /* VALIDATED */
STATE_LOW_ACTIVE                STATE_LOW_ACTIVE
STATE_HIGH_ACTIVE               STATE_MIDDLE_ACTIVE
STATE_LOW_ACTIVE                STATE_LOW_ACTIVE
STATE_LOW_ACTIVE                STATE_MIDDLE_ACTIVE
STATE_HIGH_ACTIVE               STATE_LOW_ACTIVE
STATE_MIDDLE_ACTIVE             STATE_HIGH_ACTIVE
STATE_LOW_ACTIVE                STATE_LOW_ACTIVE
```

Results will be shown in a table as shown below:

Test	result
Priority inversion avoidance system call present	YES or NO
System call used	
Test succeeded	YES or NO
Priority inversion avoided	YES or NO
Mechanism used if any?	INHERITENCE or CEILING

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.5.2 Mutex acquire-release timings: contention case (MUT_P_ARC)

This is the same test as above, but performed in a loop. In this case, the time is measured to acquire and release the mutex in the priority inversion case.

4.5.2.1 Test Parameters

None

4.5.2.2 Measurements done

Following times are measured:

Test	Sample qty	Avg	Max	Min
Mutex acquisition time, contended				
Mutex release time, contended				

4.5.2.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO

Diagrams:

- Mutex acquisition time, contended.
- Mutex release time, contended.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.6 Interrupt tests (IRQ)

Here the performance of the interrupt handling in the operating system and hardware is tested.

In a real-time system, interrupt handling is a major part of the system: indeed such systems are typically event driven. The stress tests, which check how stable the interrupt latency is shows how much the kernel uses a critical section with disabling interrupts. Real-Time operating systems do this as less and as short as possible.

The table below shows the different tests done for this object:

<test name>	description
B_SIM	Behavior of nested interrupts: do they prioritize, or are they handled in a FIFO way.
P_LAT	Interrupt latency (from interrupt to interrupt handler), hardware and operating system delay combined.
P_DLT	Interrupt dispatch latency (from interrupt handler to interrupted thread) when no rescheduling occurs.
S_SUS	Maximum sustained interrupt frequency the system can handle without loosing interrupts.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.6.1 Simultaneous interrupt priority handling (IRQ_B_SIM)

This test verifies if simultaneous interrupts are handled prioritized. It answers the question if a lower priority interrupt can be pre-empted by a higher-level interrupt.

Just like thread priorities, prioritization of interrupts makes higher level interrupts more predictable. Remark that it is not always possible to change the priority of a certain interrupt: this depends largely on the platform used.

Prioritization behaviour can easy tested by starting the interrupt generation of one device in the interrupt handler of the other device. This is done in two scenarios, in one of the two scenarios the interrupt handler will be interrupted by the other. In the other scenario the interrupt handler won't be interrupted. If this is the case, then prioritization occurs.

4.6.1.1 Test Parameters

- SCENARIO: test scenario selected
  - 1: Interrupt handler A will initiate interrupt B.
  - 2: Interrupt handler B will initiate interrupt A.

4.6.1.2 Measurements done

Only state change is measured

4.6.1.3 Test results

If the test fails, then the operating system will NOT receive the RT-VALIDATED logo!

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Interrupt pre-emption existing following the documentation?	-
Lower level interrupt pre-empted by higher level interrupt?	YES or NO
Higher-level interrupt not pre-empted by lower level interrupt?	YES or NO
If not priority based: which mechanism is used?	LIFO or FIFO

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.6.2 Interrupt latency (IRQ_P_LAT)

This measures the time it takes to switch from a running thread to an interrupt handler. So it only measures the software latency.

The latency caused by the hardware: from interrupt line going high, until the processor starts the exception vector, is not measured here.

4.6.2.1 Test Parameters

None

4.6.2.2 Measurements done

Following times are measured:

Test	Sample qty	Avg	Max	Min
Dispatch latency from interrupt handler				

4.6.2.3 Test results

Diagrams:

- Dispatch latency from interrupt handler.

4.6.3 Interrupt dispatch latency (IRQ_P_DLT)

This measures the time it takes to switch from the interrupt handler back to the interrupted thread.

The total overhead on an interrupted thread is both the interrupt latency and the dispatch latency. Of course also the duration of the interrupt handling itself has to be added.

4.6.3.1 Test Parameters

None

4.6.3.2 Measurements done

Following times are measured:

Test	Sample qty	Avg	Max	Min
Dispatch latency				

4.6.3.3 Test results

Diagrams:

- Dispatch latency.

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.6.4 Interrupt to thread latency (IRQ_P_TLT)

4.6.4.1 Test results

This measures the time it takes to switch from the interrupt handler to the thread that is activated (by using a semaphore if this can be provided by the OS) from the interrupt handler.

This can largely depend on the operating system under test. In the generic code the semaphore is used to do this. However, in Linux for instance, there is no simple way to do this. Most drivers in Linux will block the calling process and activate it again when data is available (interrupt occurred).

Most RTOS do provide such a mechanism.

4.6.4.2 Test results

Test	Sample qty	Avg	Max	Min
Latency from interrupt to activated thread				

4.6.4.3 Diagrams

Diagrams:

– Latency from interrupt to activated thread.

4.6.5 Maximum sustained interrupt frequency (IRQ_S_SUS)

This test measures the probability an interrupt is missed: is the interrupt handling duration stable and predictable?

The test is done on different levels, depending on the RTOS and the results of each test:

– 100 000 interrupts, initial phase: each test takes only some seconds.

– 1 000 000 interrupts, second phase based on the results from the first phase. This test still takes less than a minute and gives already accurate results.

– 1 000 000 000 interrupts, takes some hours: to verify stability.

On some operating system the worst-case interrupt latency is so large, that it is impossible to do the test with a billion interrupts. In such case, a smaller number of interrupts will be used.

4.6.5.1 Test results

Shown in a table as below. Remark that this table is just an example.

Interrupt period	#interrupts generated	#interrupts serviced	#interrupts lost
20 µs	100 000	100 000	0
20 µs	1 000 000	999 982	18
25 µs	1 000 000	1 000 000	0
25 µs	1 000 000 000	1 000 000 000	0

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

http://www.dedicated-systems.com
email: info@dedicated-systems.com

4.7 Memory tests (MEM)

This test will check if there are memory leaks in the operating system. It will create and delete in a loop different type of operating system objects (threads, semaphores, mutex, …).

The table below shows the different tests on this subject:

<test name>	description
B_LEK	Test if there are memory leaks in the OS.

4.7.1 Memory leak test (MEM_B_LEK)

This test continuously create/remove objects in the operating system (threads, semaphores, mutexes, …).

4.7.1.1 Test Parameters

– None

4.7.1.2 Measurements done

Nothing: memory consumption is checked after a large number of test loops.

4.7.1.3 Test results

Results will be shown in a table as shown below:

Test	result
Test succeeded	YES or NO
Test duration (how long we let the endless loop run)	
Number of main test loops done	

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.

5 Appendix A: Document revision history

5.1 Issue 1.0 (April 29, 2004)

Initial version

http://www.dedicated-systems.com
email: info@dedicated-systems.com

© Copyright Dedicated Systems Experts. All rights reserved, no part of the contents of this document may be reproduced or transmitted in any form or by any means without the written permission of Dedicated Systems Experts.