WIND RIVER

VxWorks® Benchmark Data Sheet

VxWorks 6.7
532 MHz ARM1136JF-S

Introduction

This datasheet presents benchmark results for a Freescale ARM 1136JFS iMX31lite target board running VxWorks 6.7. For definitions of the individual benchmarks in this datasheet, please refer to the Glossary at the end of this report.

Target Board Configuration

- fsl_imx31lite
- 532 MHz ARM1136JF-S
- 8 KB L1 I-cache
- 8 KB L1 D-cache
- 128KB united L2 cache

Results

The benchmark results presented in this datasheet consist of interrupt and task response times, context switch times, and various kernel and user-mode routine timings.

VxWorks 6, like VxWorks 5 before it, fully meets the needs of devices requiring hard real-time performance. Of course, new features in VxWorks 6, such as MMU-based memory protection, have necessitated low-level changes to the RTOS. The performance of the individual OS routines listed in this datasheet may have been affected, but may not in general be relevant to application steady-state performance.

For example, although object creation and deletion routines have become slower, they are usually used in initialization and shutdown procedures and are not generally used where hard real-time functionality is required.

While the benchmark numbers reported in this datasheet are provided as a guideline to VxWorks 6 performance, how the performance of these routines affects overall application performance depends in large part on the behavior of the specific application.

General Timing Information

The measurements in this benchmark are obtained

using a VxWorks timestamp driver.
All times are given in microseconds (µs), unless otherwise noted.

Maximal Performance Profile

The Maximal Performance Profile is a build of the VxWorks runtime source, intended to approximate deployed kernel-mode-only VxWorks systems. This profile builds the kernel without support for the following:

- Real-Time Processes (RTP)
- System-Viewer Instrumentation
- Kernel object ID validation checks
- Kernel object ownership information
- Kernel object class lists

WIND RIVER

Kernel Mode Timings

	Default Build µs	Maximal Performance µs
Key Measurements		
Interrupt response	3.062	1.892
Task response	10.559	6.246
Task Context Switches		
Task blocked on unavailable event	4.174	3.828
Task unblocked by incoming event	4.239	2.831
Task blocked by unavailable message	4.127	3.794
Task unblocked by incoming message	6.821	5.395
Task blocked by unavailable semaphore	4.886	3.463
Task unblocked by available semaphore	4.681	3.488
Semaphores		
semCreate		
Binary	6.332	5.828
Counting	6.311	5.962
Mutual Exclusion	6.009	6
semDelete		
Binary	10.438	6.396
Counting	10.523	6.397
Mutual Exclusion	10.523	6.648
semGive		
Tasks in queue		
Binary	2.095	2.254
Binary inline	1.574	0.777
Counting	2.135	1.383
Mutual Exclusion	2.481	1.949
Mutual Exclusion inline	2.185	1.505
No tasks in queue		
Binary	0.402	0.28
Binary inline	0.12	0.127
Counting	0.409	0.281
Mutual Exclusion	0.567	0.473
Mutual Exclusion inline	0.192	0.195
semTake		
Semaphore available		
Binary	0.409	0.296
Binary inline	0.101	0.105
Counting	0.342	0.274
Mutual Exclusion	0.447	0.33
Mutual Exclusion inline	0.136	0.144
Semaphore unavailable		
Binary	0.494	0.378
Binary inline	0.191	0.214
Counting	0.493	0.378
Mutual Exclusion	0.443	0.327
Mutual Exclusion inline	0.188	0.214

	Default Build µs	Maximal Performance µs
Reader-Writer semaphores		
semCreate	6.977	7.619
semDelete	9.727	6.925
semGive		
No tasks in queue		
acquired for writing, no tasks pending	0.614	0.561
acquired for reading, no tasks pending	0.673	0.564
Tasks in queue		
writer in queue; acquired for writing	2.673	2.278
writer in queue; acquired for reading	0.812	0.691
reader in queue; acquired for writing	0.812	0.868
semTake		
Semaphore available		
caller owns semaphore for writing	0.51	0.379
caller owns semaphore for reading	0.468	0.372
Semaphore unavailable		
caller owns semaphore for writing	0.473	0.353
caller owns semaphore for reading	0.391	0.3
Tasks		
taskInit	37.618	39.316
taskActivate	1.33	1.782
taskLock	0.093	0.055
taskUnlock	0.375	0.3
taskResume		
Ready task	0.592	0.596
Pended task	0.589	0.526
Suspended task	0.962	1.042
Delayed task	0.586	0.523
taskSuspend		
Ready task	1.214	1.204
Pended task	0.909	0.766
Suspended task	0.611	0.52
Delayed task	0.925	0.751
taskPrioritySet		
Ready task	1.275	1.336
Pended task	0.939	0.842
taskSpawn	44.052	42.335
taskDelete	47.565	36.186
Message Queues		
msgQReceive		
Message available	1.614	1.413
Message unavailable	1.444	1.426
msgQSend		
No tasks pending	1.875	1.593
Tasks pending	2.945	2.691
Queue full	1.571	1.273
msgQCreate	7.253	9.088
msgQDelete	8.362	7.131

	Default Build µs	Maximal Performance µs
Events		
Send self	0.27	0.24
Receive available	0.315	0.315
Receive unavailable	0.436	0.466
Task send wanted	1.233	1.168
Task send unwanted	0.323	0.321
Watchdogs		
wdCreate	5.194	4.707
wdDelete		
Timer started	8.501	5.614
Timer not started	8.423	4.577
wdStart		
No timer in queue	1.195	2.056
Timer in queue	1.436	2.689
wdCancel	0.889	1.381

User Mode Timings

	Default Build µs
Key Measurements	
System call overhead	2.089
Context Switches	
Within an RTP	
Task blocked on unavailable event	21.252
Task unblocked by incoming event	18.274
Task blocked by unavailable message	21.268
Task unblocked by incoming message	27.022
Task blocked by unavailable semaphore	22.754
Task unblocked by available semaphore	23.499
RTP-to-RTP	
Task blocked on unavailable event	89.838
Task unblocked by incoming event	88.524
Task blocked by unavailable message	87.524
Task unblocked by incoming message	117.383
Task blocked by unavailable semaphore	84.679
Task unblocked by available semaphore	83.672
Semaphores	
semCreate	
Binary	43.657
Counting	41.905
Mutual Exclusion	42.444
semDelete	
Binary	56.092
Counting	55.543
Mutual Exclusion	55.844
semGive	
Tasks in queue	
Binary	9.396
Counting	9.658
Mutual Exclusion	20.432
No tasks in queue	
Binary	5.603
Counting	5.564
Mutual Exclusion	0.939
semTake	
Semaphore available	
Binary	5.749
Counting	5.666
Mutual Exclusion	0.882
Semaphore unavailable	
Binary	7.599
Counting	7.761
Mutual Exclusion	0.869

	Default Build µs
Tasks	
taskSpawn	449.275
taskDelete	460.44
taskResume	
Ready task	6.45
Pended task	5.959
Suspended task	7.041
Delayed task	6.405
taskSuspend	
Ready task	7.732
Pended task	7.086
Suspended task	6.514
Delayed task	5.789
Events	
Send self	3.459
Receive available	3.513
Receive unavailable	4.165
Task send wanted	7.522
Task send unwanted	4.653
Message Queues	
msgQCreate	27.718
msgQDelete	35.081
msgQReceive	
Message available	7.143
Message unavailable	10.66
msgQSend	
No tasks pending	7.823
Task pending	12.125
Queue full	10.585

Glossary of Benchmark Metrics

Context Switching

Task context switch time

Task blocked on unavailable event

Task context switch time caused by the outgoing task executing eventReceive() on an unavailable event.

Task unblocked by incoming event

Task context switch time caused by the outgoing task executing eventSend() to the incoming higher priority task.

Task blocked by unavailable message

Task context switch time caused by the outgoing task executing msgQReceive() on an unavailable message queue.

Task unblocked by incoming message

Task context switch time caused by the outgoing task executing msgQSend() to the incoming higher priority task.

Task blocked by unavailable semaphore

Task context switch time caused by the outgoing task executing semTake() on an unavailable semaphore.

Task unblocked by available semaphore

Task context switch time caused by the outgoing task executing semGive() to the incoming higher priority task.

Events

Send self

Task to send event to self

Receive available

Receive event that is already available (does not block)

Receive unavailable

Receive unavailable event with no wait (does not block)

Task send wanted

Send an event wanted by another lower-priority task

Task send unwanted

Send an event that no task is currently waiting for

Interrupt Response

The time from when an interrupt is raised until the user-installed handler starts execution. This time includes: saving the interrupted task context, setting up the interrupt context, and identifying the source of the interrupt by the interrupt controller driver, plus the overhead of the timer driver used as the auxiliary clock.

Task response

The task response is the time elapsed from when an interrupt is raised until a task resumes execution after completion of the interrupt handling. There are two types task response times provided. In the first case, there is no rescheduling, and the interrupted task resumes execution after the interrupt handling completed. In the second case, the interrupt handler causes rescheduling by giving a binary semaphore that unblocks a different task than the task that was interrupted.

