

VxWorks® Benchmark Data Sheet

VxWorks 6.7 1 GHz MPC 7455

Introduction

This datasheet presents benchmark results for a Freescale MPC 7455 based MVME5500 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

- BSP: mv5500
- 1 GHz MPC 7455
- 64 KB L1 I-cache
- 64 KB L1 D-cache
- 256KB 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 (μ s), 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

Kernel Mode Timings

page 2

	Default Build µs	Maximal Performance µs
Key Measurements		
Interrupt response	2.22	2.189
Task response	2.85	2.744
Task Context Switches		
Task blocked on unavailable event	0.475	0.437
Task unblocked by incoming event	0.48	0.442
Task blocked by unavailable message	0.548	0.477
Task unblocked by incoming message	0.828	0.75
Task blocked by unavailable semaphore	0.475	0.43
Task unblocked by available semaphore	0.444	0.389
Semaphores		
semCreate		
Binary	1.023	0.969
Counting	1.03	0.948
Mutual Exclusion	1.022	0.947
semDelete		
Binary	0.919	0.942
Counting	0.91	0.783
Mutual Exclusion	0.916	0.783
semGive		
Tasks in queue		
Binary	0.197	0.159
Binary inline	0.15	0.126
Counting	0.195	0.159
Mutual Exclusion	0.24	0.209
Mutual Exclusion inline	0.198	0.186
No tasks in queue		
Binary	0.057	0.047
Binary inline	0.015	0.015
Counting	0.059	0.049
Mutual Exclusion	0.081	0.07
Mutual Exclusion inline	0.023	0.024
semTake		
Semaphore available		
Binary	0.061	0.045
Binary inline	0.014	0.013
Counting	0.06	0.045
Mutual Exclusion	0.076	0.05
Mutual Exclusion inline	0.02	0.015
Semaphore unavailable		
Binary	0.07	0.054
Binary inline	0.022	0.023
Counting	0.071	0.056
Mutual Exclusion	0.073	0.048
Mutual Exclusion inline	0.022	0.023

page 2

	Default Build µs	Maximal Performance µs
Reader-Writer semaphores		
semCreate	1.326	1.105
semDelete	0.987	0.663
semGive		
No tasks in queue		
acquired for writing, no tasks pending	0.088	0.076
acquired for reading, no tasks pending	0.095	0.084
Tasks in queue		
writer in queue; acquired for writing	0.25	0.207
writer in queue; acquired for reading	0.09	0.075
reader in queue; acquired for writing	0.088	0.075
semTake		
Semaphore available		
caller owns semaphore for writing	0.089	0.07
caller owns semaphore for reading	0.078	0.064
Semaphore unavailable		
caller owns semaphore for writing	0.085	0.067
caller owns semaphore for reading	0.067	0.053
Tasks		
taskInit	7.134	5.328
taskActivate	0.15	0.12
taskLock	0.016	0.006
taskUnlock	0.06	0.06
taskResume		
Ready task	0.092	0.09
Pended task	0.091	0.09
Suspended task	0.121	0.116
Delayed task	0.091	0.09
taskSuspend		
Ready task	0.12	0.109
Pended task	0.09	0.079
Suspended task	0.084	0.08
Delayed task	0.085	0.079
taskPrioritySet		
Ready task	0.199	0.189
Pended task	0.135	0.129
taskSpawn	8.03	6.971
taskDelete	5.706	5.326
Message Queues		
msgQReceive		
Message available	0.252	0.232
Message unavailable	0.192	0.168
msgQSend		
No tasks pending	0.288	0.259
Tasks pending	0.414	0.386
Queue full	0.202	0.174
msgQCreate	1.461	1.351
msgQDelete	0.98	0.872

	Default Build μs	Maximal Performance μs
Events		
Send self	0.046	0.039
Receive available	0.06	0.06
Receive unavailable	0.07	0.072
Task send wanted	0.189	0.192
Task send unwanted	0.049	0.045
Watchdogs		
wdCreate	0.736	0.576
wdDelete		
Timer started	0.858	0.702
Timer not started	0.814	0.662
wdStart		
No timer in queue	0.151	0.15
Timer in queue	0.18	0.204
wdCancel	0.113	0.097

User Mode Timings

page 5

Default Build

 μ s**Key Measurements**

System call overhead	1.057
----------------------	-------

Context Switches

Within an RTP

Task blocked on unavailable event	2.131
Task unblocked by incoming event	2.002
Task blocked by unavailable message	2.43
Task unblocked by incoming message	2.579
Task blocked by unavailable semaphore	2.187
Task unblocked by available semaphore	2.392

RTP-to-RTP

Task blocked on unavailable event	3.571
Task unblocked by incoming event	3.433
Task blocked by unavailable message	4.018
Task unblocked by incoming message	4.02
Task blocked by unavailable semaphore	3.481
Task unblocked by available semaphore	3.51

Semaphores

semCreate

Binary	7.268
Counting	5.561
Mutual Exclusion	4.903

semDelete

Binary	7.747
Counting	7.187
Mutual Exclusion	7.005

semGive

Tasks in queue	
Binary	1.59
Counting	1.59
Mutual Exclusion	3.36
No tasks in queue	
Binary	1.511
Counting	1.514
Mutual Exclusion	0.116

semTake

Semaphore available	
Binary	1.522
Counting	1.522
Mutual Exclusion	0.108
Semaphore unavailable	
Binary	1.582
Counting	1.575
Mutual Exclusion	0.108

page 5

μ s**Tasks**

taskSpawn	146.657
taskDelete	117.077
taskResume	
Ready task	1.563
Pended task	1.565
Suspended task	1.65
Delayed task	1.563
taskSuspend	
Ready task	1.65
Pended task	1.619
Suspended task	1.569
Delayed task	1.619

Events

Send self	1.365
Receive available	1.454
Receive unavailable	1.485
Task send wanted	1.605
Task send unwanted	1.469

Message Queues

msgQCreate	5.035
msgQDelete	3.062
msgQReceive	
Message available	1.755
Message unavailable	1.807
msgQSend	
No tasks pending	1.845
Task pending	1.963
Queue full	1.83

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.