MQX Int\_psp Test Suite Specification

# Objectives

Objectives of this test suite are to confirm functionality of task priorities, task synchronization by messages, semaphores, events. Test also confirms interrupt enable/ disable functions and ISR table operations.

## Reference documentation

| Document Name | Version |
| --- | --- |
| 1. MQX User Guide | rev. 3 |
| 1. MQX Reference Manual | rev 6 |

Table 1. Reference documentation

## Test environment

|  |  |
| --- | --- |
| **Software Item Name** | **Version** |
| CodeWarrior | 10.1 |
| IAR Embedded Workbench | 6.21 |
| Code Warrior [MPC] | 9.2 |

Table 2. Required software resources

|  |  |
| --- | --- |
| **Hardware Item Name** | **Version** |
| TWR – K60N512 | N/A |
| TWR – MCF51CN | N/A |
| TWR – MCF52259 | N/A |
| TWR – MCF54418 | N/A |
| TWR – MPC5125 | N/A |

Table 3. Required hardware resources

# General API

|  |  |
| --- | --- |
| *API name* | *Test app containing API* |
| \_int\_disable | Int\_psp: #1 |
| \_int\_enable | Int\_psp: #2 |
| \_int\_set\_vector\_table | Int\_psp: #3 |
| \_int\_get\_vector\_table | Int\_psp: #3, #13 |
| \_int\_get\_kernel\_isr | Int\_psp: #4, #5 |
| \_int\_install\_kernel\_isr | Int\_psp: #4, #5, #6, #7, #8, #9, #10, #11, #14 |
| \_msgpool\_create | Int\_psp: #6 |
| \_msg\_alloc | Int\_psp: #6 |
| \_msgq\_send | Int\_psp: #6 |
| \_msgq\_open | Int\_psp: #6 |
| \_event\_create | Int\_psp: #7 |
| \_event\_set | Int\_psp: #7 |
| \_event\_open | Int\_psp: #7 |
| \_event\_wait\_all | Int\_psp: #7 |
| \_sem\_create | Int\_psp: #8 |
| \_sem\_post | Int\_psp: #8 |
| \_sem\_open | Int\_psp: #8 |
| \_sem\_wait | Int\_psp: #8 |
| \_lwsem\_create | Int\_psp: #9 |
| \_lwsem\_post | Int\_psp: #9 |
| \_lwsem\_wait | Int\_psp: #9 |
| \_taskq\_create | Int\_psp: #10 |
| \_taskq\_resume | Int\_psp: #10 |
| \_taskq\_suspend | Int\_psp: #10 |
| \_task\_get\_td | Int\_psp: #11 |
| \_task\_ready | Int\_psp: #11 |
| \_time\_dequeue\_td | Int\_psp: #11 |
| \_int\_get\_previous\_vector\_table | Int\_psp:#12 |

# specifications

Some of kernel tests.

## Test app – int\_psp

Testing tasks priorities, ISR priority, custom ISR, vector table location, sending messages, semaphores, lightweight semaphores, events, task queue.

### Test case #1 – Testing \_int\_disable function

Test multiple call \_int\_disable function. After function call, DISABLED\_LEVEL field of task descriptor should be incremented by one and all interrupts should be disabled. Verification of disabled interrupts is performed by comparing value of kernel\_data->Time.TICKS in the loop. After finish loop, the value of kernel\_data->Time.TICKS should be the same as at the start.

Expected result: Valid value of DISABLED\_LEVEL. Disabled kernel ticks.

API used: \_int\_disable

### Test case # 2 - Testing \_int\_enable function

Test multiple call \_int\_enable function. After function call, DISABLED\_LEVEL field of task descriptor should be decremented by one and if the value is zero, original interrupts should be enabled. Verification of enabled interrupts is performed by comparing value of kernel\_data->Time.TICKS in the loop. After finish loop, the value of kernel\_data->Time.TICKS should be the different as at the start.

Expected result: Valid value of DISABLED\_LEVEL. Enabled kernel ticks.

API used: \_int\_enable

### Test case # 3 – Testing location of vector table

Testing whether the vector table location can be changed.

Note: PowerPC (MPC5125) doesn't support this feature.

Expected result: \_int\_get\_vector\_table should return the same value as passed to \_int\_set\_vector

API used: \_int\_set\_vector\_table, \_int\_get\_vector\_table

### Test case #4 – Testing \_int\_get\_kernel\_isr(index)

Testing task errno for valid and invalid ISR table index.

Expected result: errno MQX\_INVALID\_VECTORED\_INTERRUPT for invalid index, MQX\_OK for valid index

API used: \_int\_get\_kernel\_isr

### Test case #5– Testing \_int\_install\_kernel\_isr(index)

Checked correct ISR replacement for user routine and checked valid values of indexes for ISR table. Test depends on MQX\_ROM\_VECTORS settings. If MQX\_ROM\_VECTORS is 1, vector table is placed in flash and location of vector table cannot be changed.

Expected results: If MQX\_ROM\_VECTORS is 1, errno should be MQX\_INVALID\_VECTORED\_INTERRUPT, return value NULL.

Otherwise errno should be MQX\_OK and result points to old ISR routine.

API used: \_int\_install\_kernel\_isr, \_int\_get\_kernel\_isr

### Test case #6 – Testing \_msg\_send from an ISR

Test task creates a message pool, install msg\_timer\_isr routine and

create msg\_higher\_task, msg\_lower\_task. From msg\_timer\_isr it sends messages to task.

Expected results: ISR installation should be successful, all \_msq\_\* functions should return a valid values. Inside tasks, kernel\_data->IN\_ISR

should be 0. Higher task should end first.

API used: \_msgpool\_create, \_msg\_alloc, \_msgq\_send, \_msgq\_open, \_int\_install\_kernel\_isr

### Test case #7 – Testing \_event\_set from an ISR

Test task creates event, low and high priority tasks and after that install event\_timer\_isr. Tasks open event and waiting for event set. Event\_timer\_isr set events.

Expected results: ISR installation should be successful, all \_event\_\* functions should return valid values. Inside tasks, kernel\_data->IN\_ISR

should be 0. Higher task should end first.

API used:

\_event\_create, \_event\_set, \_event\_open, \_event\_wait\_all, \_int\_install\_kernel\_isr

### Test case #8 – Testing \_sem\_post from an ISR.

Test tasks create binary semaphore, higher and lower task, and set sem\_timer\_isr and install sem\_timer\_isr. Tasks perform \_sem\_wait operation. \_sem\_post is called from sem\_timer\_isr to release semaphore.

Expected results: ISR installation should be successful, all \_sem\_\* functions should return valid values. Inside tasks, kernel\_data->IN\_ISR

should be 0. Higher task should end first.

API used:

\_sem\_create, \_sem\_post, \_sem\_open, \_sem\_wait, \_int\_install\_kernel\_isr

### Test case #9 – Testing \_lwsem\_post from an ISR.

The same as test case #8, semaphores are replaced with lightweight semaphores.

Expected result: ISR installation should be successful, all \_lwsem\_\* functions should return valid values. Inside tasks, kernel\_data->IN\_ISR

should be 0. Higher task should end first.

API used:

\_lwsem\_create, \_lwsem\_post, \_lwsem\_wait, \_int\_install\_kernel\_isr

### Test case #10 – Testing \_tasq\_resume from an ISR

Firstly it creates task queue with fifo policy, and then create higher and lower tasks. Both tasks will suspend themselves. After, the taskq\_timer\_isr is installed and during execution it will call \_taskq\_resume.

Expected result: ISR installation should be successful, all \_task\* functions should return valid values. Inside tasks, kernel\_data->IN\_ISR

should be 0. Higher task should end first.

API used:

\_taskq\_create, \_taskq\_resume, \_taskq\_suspend, \_int\_install\_kernel\_isr

### Test case #11 – Testing \_time\_dequeue from an ISR

It creates timer\_higher\_task, timer\_lower\_task and then install BSP\_TIMER\_INTERRUPT\_VECTOR timer\_timer\_isr. Both task will delay themself, and timer\_timer\_isr will set them to the ready queue.

Expected result: ISR installation should be successful, all \_task\* functions should return valid values. Inside tasks, kernel\_data->IN\_ISR should be 0. Higher task should end first.

API used:

\_task\_get\_td, \_task\_ready, \_time\_dequeue\_td, \_int\_install\_kernel\_isr

### Test case #12 – Testing \_int\_get\_previous\_vector\_table

Just tests whether \_int\_get\_previous\_vector\_table is avaible.

Note: PowerPC version of PSP doesn’t implement this function.

Expected results: compiler will not thrown an error

API used: \_int\_get\_previous\_vector\_table

### Test case #13 – Testing \_int\_get\_vector\_table

Just tests whether \_int\_get\_vector\_table is avaible.

Expected results: compiler will not thrown an error

API used: \_int\_get\_vector\_table

### Test case #14 – Testing tasks and ISR priority

Test tests task and ISR priority. It install count\_kernel\_isr (normal priority) where it increment counter. In the end of high task (highest priority) the value of counter should be the same as at start. In the end of the low task (lowest priority) the value should be different than at the start.

Expected results: Valid value of counter.

API used: \_int\_install\_kernel\_isr, using PIT via register

# KNOWN ISSUES

# POSSIBLE IMPROVEMENTS

**Revision SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Date** | **Author** | **Description of Revision & Writer** | **Spec Coord.** |
| 20.9.2011 | Marián Cingel | Initial version |  |
|  |  |  |  |