s

MQX Flashx Test Suite Specification

# Objectives

Objectives of this test suite is to confirm functionality of flashx driver.

## 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 |

Table 2. Required software resources

|  |  |
| --- | --- |
| **Hardware Item Name** | **Version** |
| TWR – K60N512 INTRAM | N/A |
| TWR – K70N512 INTRAM | N/A |
| TWR – MCF52259 EXTRAM | N/A |
| TWR – MCF51CN INTFLASH | N/A |
| TWR – PXS20 INTRAM | N/A |

Table 3. Required hardware resources

# General API

|  |  |
| --- | --- |
| *API name* | *Test app containing API* |
| fopen | flashx: all tests |
| fclose | flashx: all tests |
| read | flashx: #3, #4, #5, #6, #7 |
| write | flashx: #4, #5, #7 |
| ftell | flashx: #4, #5, #7 |
| ioctl | flashx: all tests |
| \_mem\_alloc\_zeroc | flashx: #4, #5, #6 |
| \_mem\_free | flashx: #4, #5, #6 |

## IOCTL commands

|  |  |
| --- | --- |
| *API name* | *Test app containing API* |
| FLASH\_IOCTL\_GET\_BLOCK\_MAP | flashx: #1, #2, #6 |
| FLASH\_IOCTL\_GET\_SECTOR\_SIZE | flashx: #2, #4, #5, #7 |
| IO\_IOCTL\_DEVICE\_IDENTIFY | flashx: #1 |
| IO\_IOCTL\_GET\_NUM\_SECTORS | flashx: #1 |
| IO\_IOCTL\_GET\_BLOCK\_SIZE | flashx: #1 |
| FLASH\_IOCTL\_GET\_NUM\_SECTORS | flashx: #1 |
| FLASH\_IOCTL\_GET\_WIDTH | flashx: #1 |
| FLASH\_IOCTL\_ERASE\_CHIP | flashx: #3 |
| FLASH\_IOCTL\_ENABLE\_SECTOR\_CACHE | flashx: #4, #5 |
| FLASH\_IOCTL\_DISABLE\_SECTOR\_CACHE | flashx: #4 |
| FLASH\_IOCTL\_ENABLE\_BUFFERING | flashx: #4 |
| FLASH\_IOCTL\_DISABLE\_BUFFERING | flashx: #4, #5 |

# specifications

Flashx driver test

## Flashx test

Actual test is based on old flashx test, especially tests cases no. 1, 4. Some test cases use goto and [TC]\_deinit label for clean up purpose. [TC]\_deinit should free all test resources of test case.

Default bsp flashx settings are a bit inconsistent across platforms, so it was necessary to create our own one. Each supported platform has two specific source files: src/flashx\_settings.c, include/flashx\_settings.h .

flashx\_settings.h contains macros used as platform specific settings

|  |  |
| --- | --- |
| **Macro/settings** | **Description** |
| ENABLE\_CASE[1 - 7] | enable /disable specific test case |
| CASE3\_CFM\_ENABLE | enable writing cfm array after chip erase |
| CASE3\_CFM\_OFFSET | set cfm array offset |
| CASE3\_CFM\_SIZE | set cfm array size |
| CASE4\_HALF\_PARTIAL | enable specific sub test of test case 4 |
| CASE4\_ONE\_PARTIAL | enable specific sub test of test case 4 |
| CASE4\_THREE\_FULL | enable specific sub test of test case 4 |
| CASE4\_THREE\_PARTIAL | enable specific sub test of test case 4 |
| CASE5\_HALF\_PARTIAL | enable specific sub test of test case 5 |
| CASE5\_ONE\_PARTIAL | enable specific sub test of test case 5 |
| CASE5\_ONE\_FULL | enable specific sub test of test case 5 |
| CASE5\_TWO\_PARTIAL | enable specific sub test of test case 5 |
| CASE5\_THREE\_FULL | enable specific sub test of test case 5 |

flashx\_setting.c contains global variable definition used in test. “cfm”variable is cloned originaly from bsp/vectors.c, variables “\_test\_flashx\_file\_blocks”, “\_test\_flashx\_init” from bsp/init\_flashx.c

|  |  |
| --- | --- |
| **Variable** | **Description** |
| \_test\_flashx\_file\_blocks | list of defined flashx files |
| \_test\_flashx\_init | init structure for flashx driver |
| cfm | cfm array |

Test uses and defines (\_test\_flashx\_file\_blocks) these flashx files:

|  |  |
| --- | --- |
| **File** | **Description** |
| cfm | flash area to containing cfm array |
| work\_rw | workspace for read/write operation |
| mirror\_rw | mirror of read/write workspace |
| work\_ro | read only workspace |
| erased\_ro | flash area affected by FLASH\_IOCTL\_ERASE\_CHIP |
| whole\_ro | whole flash area available for testing hw blocks |

Flashx file has to be sector aligned. All necessary information about flash can be found in board bsp files.

Note:

1. FLASH\_IOCTL\_ERASE\_CHIP has a different behavior across platform. Pxs20 respect borders of file, k60 erase entire chip (purpose of ” erased” file)
2. If erase chip should not be performed, ENABLE\_CASE3 should be set to 0
3. Generated patterns are “kinetis safe” so significant changes should be firstly verified on this platform
4. After chip erase (CodeWarrior, OpenBDM) openbdm doesn’t work anymore.
5. How to unlock kinetis: http://supp.iar.com/Support/?note=77989

### Test case # 1 – Test device and flash information

Flashx driver provides basic information about device driver itself and flash memory. Test verifies these values against expected results.

Expected result: Specific values for general device/driver information, positive number for any other specific information.

API used: fopen, fclose, ioctl

### Test case # 2 - Test blocks size and seek

Test obtain array of blocks. Each item (one block) contains information about number of sectors, sector size, … Test iterate over an array and set file cursor to each sector of blocks. Test compares sector size field of block map against value gained from ioctl FLASH\_IOCTL\_GET\_SECTOR\_SIZE. The values should be the same.

Expected result: Sector size of blocks must have the same value as sector size from ioctl command.

API used: fseek, fopen fclose, ioctl

### Test case # 3 – Test test chip erase

Test try to erase whole flash area by calling the ioctl command FLASH\_IOCTL\_ERASE\_CHIP. Once erased, the function place\_is\_erased is called to ensure that “ erased\_ro“ file is empty.

Expected result: “erased\_ro” file should be empty.

API used: fopen, fclose, fseek, read, ioctl

### Test case #4 – Test single write operation

Test try to writes data with different settings (size and offset) to single file.

|  |  |  |  |
| --- | --- | --- | --- |
| Constant/Settings | Data offset [sec. boundary] | Data size | Consume |
| CASE4\_HALF\_PARTIAL | ¼ size | ½ sector size | 1 sectors |
| CASE4\_ONE\_PARTIAL | ¾ sector size | 1 sector size | 2 sectors |
| CASE4\_THREE\_FULL | 0 – at the boundaries | 3 sector size | 3 sectors |
| CASE4\_THREE\_PARTIAL | ¼ sector size | 3 sector size | 4 sectors |

Each constant/settings can consists of these sub tests.

|  |  |
| --- | --- |
| Constant | Description |
| CASE4\_USE\_CACHED\_BUFFERED | Perform buffered/cached data write |
| CASE4\_USE\_CACHED\_UNBUFFERED | Perform cached/unbuffered data write |
| CASE4\_USE\_ERASE\_SECTOR | Perform sectors (placed under data) erase |
| CASE4\_USE\_UNCACHED\_PASS | Perform uncached write, write has to pass |
| CASE4\_USE\_UNCACHED | Perform uncached write, result depends on sectors content placed under written data |

For example CASE4\_ONE\_PARTIAL can be defined in flashx\_settings.h as:

#define CASE4\_ONE\_PARTIAL

( \

(CASE4\_USE\_CACHED\_BUFFERED << (CASE4\_SHIFT\_SIZE \* 0)) | \

(CASE4\_USE\_UNCACHED << (CASE4\_SHIFT\_SIZE \* 1)) | \

(CASE4\_USE\_CACHED\_UNBUFFERED << (CASE4\_SHIFT\_SIZE \* 2)) | \

(CASE4\_USE\_ERASE\_SECTOR << (CASE4\_SHIFT\_SIZE \* 3)) | \

(CASE4\_USE\_UNCACHED << (CASE4\_SHIFT\_SIZE \* 4)) \

)

will perform sub tests (buffered, uncached, unbuffered, erase, …) in defined order with settings corresponding to CASE4\_HALF\_PARTIAL. If CASE4\_HALF\_PARTIAL is defined as 0, this setting will not perform.

Expected result: expected results of write operations depend on flash state and specific driver options.

API used: fopen, fclose, read, ftell, write, fseek, ioctl, \_mem\_free, \_mem\_alloc\_zero

### Test case #5 – Test multiple write operation

Test try to write data to the same flash space by using three files. “read\_file” and “write\_file” use flashx “write\_rw” file, “mirror\_file” use “mirror\_rw” flashx file. Test try to write data with different settings (offset and size).

|  |  |  |  |
| --- | --- | --- | --- |
| Loop | Data offset [sec. boundary] | Data size | Consume |
| CASE5\_HALF\_PARTIAL | ¼ size | ½ sector size | 1 sectors |
| CASE5\_ONE\_PARTIAL | ¾ sector size | 1 sector size | 2 sectors |
| CASE5\_ONE\_FULL | 0 – at the boundaries | 1 sector size | 1 sectors |
| CASE5\_TWO\_PARTIAL | 1/3sector size | 2 sector size | 3 sectors |
| CASE5\_THREE\_FULL | 0 – at the boundaries | 3 sector size | 3 sectors |

Test creates a buffer, multiple filled with different pattern. Buffer is written to specific position of first file and then red back from second one. After reading, the buffer should be the same as written data. Firstly is performed write to write\_file/read\_file and then mirror\_file/read\_file.

Expected results: Successful write operations and the same content of write and read buffer.

API used: fopen, fclose, read, ftell, write. fseek, ioctl, \_mem\_free, \_mem\_alloc\_zero

### Test case #6 – Test read and seek operation

Test try to read data from five different place of “work\_ro” file area. Position of these places is set from start position of file, end position of file, and previous relative position. For each one is performed read operation and readed data should be the same for all of these three types.

Note: Test results are more accurate if flash file contain different pattern.

Expected results: Sucessfull read operation and data comparsion.

API used: fopen, fclose, read, ftell, fseek, ioctl, \_mem\_free, \_mem\_alloc\_zero

### Test case #7 – Test read, write after valid location

Test case 7 contains two subparts, tc\_7\_main\_task\_valid\_write and tc\_7\_main\_task\_invalid\_write.

tc\_7\_main\_task\_valid\_write try to write buffer over the end of the file. The number of bytes placed inside flash file area should be written. Back reading from valid area should return the same data as were written.

tc\_7\_main\_task\_invalid\_write try to write data after end of the file. Write should fail.

Expected results: Write/read overlap flash size should write/read a valid number of data. Write/read after flash size should fail.

Api used: fopen, ftell, write, read, fseek, fclose, ioctl

### Supported functions:

#### buffer\_create\_pattern

|  |  |
| --- | --- |
| **Params** | **Description** |
| uchar\_ptr buffer | Pointer to buffer |
| \_mqx\_uint length | Length of buffer |
| \_mqx\_uint type | Different type of pattern |
| return | void |

Function fill up buffer with specific pattern – depending on type number.

#### buffer\_compare\_pattern

|  |  |
| --- | --- |
| **Params** | **Description** |
| uchar\_ptr buffer | Pointer to buffer |
| \_mqx\_uint length | Length of buffer |
| \_mqx\_uint type | Different type of pattern |
| return | TRUE if pass, FALSE if fail |

Function compare buffer with specific pattern – depending on type number.

#### buffer\_optimal\_size

|  |  |
| --- | --- |
| **Params** | **Description** |
| \_mqx\_int required\_buffer\_size | Required size of buffer |
| \_mqx\_int number\_of\_buffers | Number of buffers |
| return | Optimal buffer size |

Function check if there is enough memory to allocate (number\_of\_buffers \* required\_buffer\_size). If condition passes, the function returns required\_buffer\_size, otherwise the maximum available memory for one buffer.

#### buffer\_compare\_buffer

|  |  |
| --- | --- |
| **Params** | **Description** |
| uchar\_ptr first | First buffer pointer |
| uchar\_ptr second | Second buffer pointer |
| \_mqx\_uint length | Length of buffer |
| return | TRUE if pass, FALSE if fail |

Function compares content of two buffers of length “length”.

#### buffer\_compare\_sign

|  |  |
| --- | --- |
| **Params** | **Description** |
| uchar\_ptr buffer | Buffer pointer |
| \_mqx\_uint length | Length of buffer |
| uchar sign | Sign |
| return | TRUE if pass, FALSE if fail |

Function checks if buffer is filled with specific value (sign).

#### find\_sector\_start

|  |  |
| --- | --- |
| **Params** | **Description** |
| MQX\_FILE\_PTR flash\_file | File pointer |
| \_mqx\_int location | Required location |
| return | -1 if fail, otherwise location of sector |

Function calculates base address of sector where location should be placed.

#### check\_valid\_place

|  |  |
| --- | --- |
| **Params** | **Description** |
| MQX\_FILE\_PTR flash\_file | File pointer |
| \_mqx\_int location | Required location |
| \_mqx\_int size | Size of data |
| return | TRUE if pass, FALSE if fail |

Function calculates whether data of “size” placed at “location” will be situated inside flash file.

#### place\_is\_erased

|  |  |
| --- | --- |
| **Params** | **Description** |
| MQX\_FILE\_PTR flash\_file | File pointer |
| \_mqx\_int location | Required location |
| \_mqx\_int remain\_size | Size of data |
| return | 1 if area is erased  0 if area is not erased  -1 for unknown error  -2 if out of memory |

Function calculates if sectors placed under data of size “remain\_size” from location “location” are erased.

#### place\_is\_clean

|  |  |
| --- | --- |
| **Params** | **Description** |
| MQX\_FILE\_PTR flash\_file | File pointer |
| \_mqx\_int location | Required location |
| \_mqx\_int size | Size of data |
| return | 1 if area is clean  0 if area is not clean  -1 for unknown error  -2 if out of memory |

Function check if overhead area (unused space of sectors placed under data) are erased.

**Revision SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Date** | **Author** | **Description of Revision & Writer** | **Spec Coord.** |
| 12.10.2011 | Marián Cingel | Initial version |  |
| 23.11.2011 | Marián Cingel | Rewritten to cross platform support |  |