Specification

# Objectives

Objectives of this test suite is to confirm functionality of MFS API using both RAM and USB disk

## Reference documentation

| Document Name | Version |
| --- | --- |
| 1. MQX MFS User Guide | 2.2 |

Table 1. Reference documentation

## Test environment

|  |  |
| --- | --- |
| **Software Item Name** | **Version** |
| CodeWarrior | 10.1 |
|  |  |

Table 2. Required software resources

|  |  |
| --- | --- |
| **Hardware Item Name** | **Version** |
| TWR – K60N512 | N/A |
| USB stick | N/A |

Table 3. Required hardware resources

Note: USB stick is not required for testing on RAM disk.

# General API

|  |  |
| --- | --- |
| *API name* | *Test app containing API* |
| \_io\_mfs\_install | all tests |
| \_io\_mfs\_uninstall | install (RAM part) |
| \_io\_part\_mgr\_install | USB disk based tests (list) |
| \_io\_part\_mgr\_uninstall | install (USB part) |
| fclose | all tests |
| fopen | all tests |
| fwrite | fileswrite, format, full, infile |
| fread | fileswrite, format, infile, |
| ioctl | all tests |

## IOCTL commands

|  |  |
| --- | --- |
| *IOCTL command name* | *Test app containing command* |
| IO\_IOCTL\_BAD\_CLUSTERS |  |
| IO\_IOCTL\_CHANGE\_CURRENT\_DIR | cache, rename2, search |
| IO\_IOCTL\_CREATE\_SUBDIR | cache, subdirs1, subdirs2 |
| IO\_IOCTL\_DEFAULT\_FORMAT |  |
| IO\_IOCTL\_DELETE\_FILE | delete, full |
| IO\_IOCTL\_FAT\_CACHE\_OFF | cache, |
| IO\_IOCTL\_FAT\_CACHE\_ON | cache, |
| IO\_IOCTL\_FIND\_FIRST\_FILE | cache, date, delete, format, full, rename1, rename2, search, subdirs1, subdirs2 |
| IO\_IOCTL\_FIND\_NEXT\_FILE | cache, format, full, rename1, rename2, search, subdirs1 |
| IO\_IOCTL\_FLUSH\_FAT |  |
| IO\_IOCTL\_FORMAT | all tests |
| IO\_IOCTL\_FORMAT\_TEST |  |
| IO\_IOCTL\_FREE\_CLUSTERS |  |
| IO\_IOCTL\_FREE\_SPACE | full, infile |
| IO\_IOCTL\_GET\_CLUSTER\_SIZE |  |
| IO\_IOCTL\_GET\_CURRENT\_DIR | rename2, |
| IO\_IOCTL\_GET\_DATE\_TIME |  |
| IO\_IOCTL\_GET\_DEVICE\_HANDLE | cache, full |
| IO\_IOCTL\_GET\_FAT\_CACHE\_MODE |  |
| IO\_IOCTL\_SET\_FAT\_CACHE\_MODE |  |
| IO\_IOCTL\_GET\_FILE\_ATTR | rename2, subdirs2 |
| IO\_IOCTL\_SET\_FILE\_ATTR | subdirs2 |
| IO\_IOCTL\_GET\_LFN | cache, rename1, rename2, subdirs1 |
| IO\_IOCTL\_GET\_VOLUME |  |
| IO\_IOCTL\_SET\_VOLUME |  |
| IO\_IOCTL\_GET\_WRITE\_CACHE\_MODE |  |
| IO\_IOCTL\_SET\_WRITE\_CACHE\_MODE |  |
| IO\_IOCTL\_LAST\_CLUSTER |  |
| IO\_IOCTL\_REMOVE\_SUBDIR | cache, subdirs1, subdirs2 |
| IO\_IOCTL\_RENAME\_FILE | rename1, rename2 |
| IO\_IOCTL\_SET\_DATE\_TIME |  |
| IO\_IOCTL\_TEST\_UNUSED\_CLUSTERS |  |
| IO\_IOCTL\_CLEAR\_PARTITION |  |
| IO\_IOCTL\_GET\_PARTITION |  |
| IO\_IOCTL\_SET\_PARTITION |  |

## Open FLAGs list

|  |  |
| --- | --- |
| *Option* | *Test applications containing open option* |
| W | fileswrite, infile, |
| W+ | rename1, rename2 |
| R | fileswrite, format, infile, |
| R+ | date, format, |
| A |  |
| A+ | fileswrite, infile, |
| N |  |
| N+ | date, delete, format, full, search, subdirs1 |
| X | format |

# specifications

Tests are split into two parts. One is common for all tests, containing initialization of testing disk (RAM/USB). Second is specific for particular test. Test may use RAM or USB disk. Both RAM and USB disk tests share the common source, located in MSGSW/MQX/src/test/mfs directory.

All operations mentioned bellow are checked for return codes and these are used by EU\_ASSERT

Test applications do not cover open flag testing, e.g.: attempt to read from “write-only” file etc.

## Test app – CACHE

### Test case #1 – Testing chache

This test case tests if the outputs from numerous disk operations are same having FAT and WRITE cache memory turned on and off. These outputs are compared using CRC32 checksum of data on disk. Data are not fetched using MFS, but by direct memory access. Test flow: disk is formatted; crc table is generated; write and FAT cache are turned on, time is saved (for the next round of operations so the test conditions can be same); disk operations are executed; CRC from disk data is calculated and saved. Second pass: disk is formatted; write and FAT cache are turned off, time is adjusted from previously saved time; disk operations are executed; CRC is calculated and compared to previous CRC.

Expected result: both CRC’s must be equal.

Disk operations are located in \_test\_suite(), in test file (test.c).

Disk operations list: create directory “short.xxx”, recreate directory “short.xxx”, create directories “longfilename1.xxx” and “longfilename2.xxx”, recreate directory “longfilename1.xxx”, remove directory “short.xxx”, searching for created directories, using name conversion from DOS 8.3 format into “long file name” format, changing directory using all existing directories, flushing output to disk.

API used: IO\_IOCTL\_FAT\_CACHE\_ON, IO\_IOCTL\_FAT\_CACHE\_OFF, IO\_IOCTL\_WRITE\_CACHE\_ON, IO\_IOCTL\_WRITE\_CACHE\_OFF, IO\_IOCTL\_CREATE\_SUBDIR, IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_FIND\_NEXT\_FILE, IO\_IOCTL\_GET\_LFN, IO\_IOCTL\_REMOVE\_SUBDIR, IO\_IOCTL\_CHANGE\_CURRENT\_DIR, fflush

## Test app - DATE

### Test case # 1 - Testing date/time

Test file is created and closed. Time stamp is saved. Test file is re-opened and closed. New time stamp is compared to the saved one. File is re-opened once again and multiple bytes (count based on test.h file) are written. File is saved and third time stamp is captured.

Expected result: all timestamps must by equal.

API used: IO\_IOCTL\_FIND\_FIRST\_FILE

## Test app – DELETE

Test case missing: Attempt to delete a hidden file.

### Test case # 1 – Testing delete

Numerous test files are created. The exact number of files depends on macro NUM\_FILES defined in header file (test.h). All the created files are closed. Creation of test files is confirmed by searching each file. All test files are deleted. Deletion is again confirmed by searching each file.

Expected result: none of the test files should by found after deletion.

API used: IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_DELETE\_FILE

## Test app – FILESWRITE

### Test case #1 – Testing files write

In beginning of test, two write buffers are created: one containing number 1 on each position, second number 2. Files “data1.txt” and “data2.txt” are created and opened for writing. 5 bytes of data are written to these files from write buffers 1 and 2 respectively. Another set of 5 bytes are written. Numerous more bytes are written to files from write buffers. Exact number of bytes depends on macro FIELD\_SIZE defined in header file (test.h) This number must be bigger than sector size, so an cross-sector write can be performed. Test files are closed and opened for reading. Contents of files are summed separately and compared to expected counts.

Next two files “append1.txt” and “append2.txt” are created at EOF in read-write mode (a+). Numerous bytes are written to test files. Exact number of bytes depends on macro NUM\_CYCLES. The bytes value is defined in macros FP1\_INCREMENT and FP2\_INCREMENT, one for each test files respectively. After write, seek operation if executed on both files toward beginning of files. Content of files is summed together for each file separately and the results compared to expected numbers.

Expected result: content of file cannot be corrupted by cross-sector writes, seek operation must navigate to desired location within the file.

API used: write, \_io\_write, read, \_io\_read, fseek

## Test app – FORMAT

### Test case #1 – Testing format command

Test file is created. After closing test file whole MFS is closed. MFS is re-opened, file system is checked and test file is opened in read-write mode. 512 bytes of data are written and read. Attempt to format MFS with a file open. This attempt should fail. Test file is closed and MFS is formatted. Test file is opened for reading (this should fail), after that test file is created and opened in read-write mode. 512 bytes of data are written to test file. 10 temporary files are created (“x” as parameter for fopen) and search for test file and 10 temporary files is performed.

Expected results: after closing and re-opening MFS no format is required, attempt to format MFS with a file open must end with error (MFS\_SHARING\_VIOLATION). After format, opening a file in read-only mode (“r”) must fail, because the file no more exists, after all open files are closed formatting must be successful.

API used: ferror, write, read, fseek, IO\_IOCTL\_FORMAT

## Test app – FULL

### Test case #1 – Testing disk full

In this test is an extra sector right beyond the MFS created and filled with known pattern (“!”). Test file is created and opened in read-write mode. Disk free space is saved. Write operations are performed so only 1B of space is free. 1B is written to file. After that another 1B is written. This write should end with an error (MFS\_DISK\_FULL). After the test file is closed, content of extra sector is checked. Test file is deleted and disk free space is compared to previously saved value. Write operations are performed, so whole free space is used by one file.

Expected results: writing to file beyond free space should result in error. Extra sector created beyond MFS must stay intact. After deletion of file, that consumed all disk free space, newly gained free space must be same as before file creation.

API used: ferror, write, IO\_IOCTL\_GET\_DEVICE\_HANDLE, IO\_IOCTL\_FREE\_SPACE, IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_FIND\_NEXT\_FILE, IO\_IOCTL\_DELETE\_FILE

## Test app – INFILE

### Test case #1 – Testing infile navigation

First a write buffer is prepared with length defined by macro FILESIZE from header file (test.h). Free disk space is checked so whole write buffer can be written. After data been written, file is opened in read mode and its content is compared to write buffer. File is opened in “append read-write” mode (a+). Fseek() function is used with various parameter values. Read and write operations are used to confirm exact navigation.

Expected results: values read from file must be same as in write buffer

API used: fwrite, fread, fseek

## Test app – INSTALL

Test case missing: Partition manager testing

### Test case #1 – Testing install/uninstall

This test application is divided into two separated parts, one for each type of disk (RAM/USB). Both parts contain functions to install and uninstall partitions on disk. Function \_ftest\_count\_parts() return partitions count.

Expected results: partitions count should be equal to expected count.

API used: \_io\_mfs\_install, \_io\_mfs\_uninstall, \_partition\_test, \_io\_part\_mgr\_install, \_io\_part\_mgr\_uninstall

## Test app – RENAME1

### Test case #1 – Testing rename

Numerous files are created and closed. Exact number of files depends on macro NUM\_FILES defined in header file (test.h). Search operations are performed to find all created files. All test files are renamed by adding known suffix (“T”). After that, search is performed using new names.

Expected result: Files after creation and renaming should by found using new assigned names.

API used: IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_RENAME\_FILE, IO\_IOCTL\_GET\_LFN

## Test app – RENAME2

### Test case #1 – Testing rename 2

Test file named “TEST\_LONG\_FILE\_NAME.TXT” is created and closed. Creation is confirmed by search for this file. “long file name” conversion is performed. Directory is created (“SHORT.XXX”). Test file is renamed using created directory as part of new name (moving file). Search for test file with old name is performed. This action should fail. Current directory is changed to “SHORT.XXX” and search is performed again.

Expected result: After rename (move) of file, search using old name should fail. After changing current directory, search using old name should pass.

API used: IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_GET\_LFN, IO\_IOCTL\_FIND\_NEXT\_FILE, IO\_IOCTL\_GET\_FILE\_ATTR, IO\_IOCTL\_CHANGE\_CURRENT\_DIR, IO\_IOCTL\_GET\_CURRENT\_DIR

## Test app – SEARCH

### Test case #1 – Testing searching

Directory is created and current directory is changed to this directory. Numerous test files are created and closed. Count and names are defined in header file (test.h) together with search results in 8.3 format. Various search operations use patterns defined also in header file. In test source file (test.c) is “Search\_results” table defined. First item in each line represent searching pattern and rest represents expected search results for that specific pattern.

Expected result: all search result should match expected results defined in “Search\_results”

API used: IO\_IOCTL\_CREATE\_SUBDIR, IO\_IOCTL\_CHANGE\_CURRENT\_DIR, IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_FIND\_NEXT\_FILE

## Test app – SUBDIRS1

Test cases missing: Attempt to change directory using name in 8.3 format.

### Test case #1 – Testing subdirectories

Directory “short.xxx” is created. Directory with same name is created again. Directory with long name “longfilename1.xx” is created. Duplicate directory is created. Another directory with long name is created “longfilename2.xxx”. Search operations are performed in order to find all mentioned directories together with “long file name” conversion. Short-named directory is deleted and third long-named “longfilename3.txt” directory is created. Current directory is changed to all 3 directories, one by one, using full names. In root a file named “kl\_dct1152\_2a.xml” and directory “kl\_dct1152\_2a” are created. Current directory is changed to “kl\_dct1152\_2a” and then to “kl\_dct1152\_2a.xml”.

Expected results: two directories with same name cannot be created. Change directory into a file is not allowed.

API used: IO\_IOCTL\_CREATE\_SUBDIR, IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_GET\_LFN, IO\_IOCTL\_FIND\_NEXT\_FILE, IO\_IOCTL\_REMOVE\_SUBDIR, IO\_IOCTL\_CHANGE\_CURRENT\_DIR

## Test app – SUBDIRS2

### Test case #1 – Testing subdirectories 2

Numerous directories are created. Exact count depends on macro TEST\_NUM\_SUBDIRS defined in header file. First of directories is made hidden. Search is performed to find the hidden directory. All created directory are deleted. Search for deleted directory is performed.

Expected results: hidden directory shall not be found. Hidden directory can be deleted.

API used: IO\_IOCTL\_CREATE\_SUBDIR, IO\_IOCTL\_GET\_FILE\_ATTR, IO\_IOCTL\_SET\_FILE\_ATTR, IO\_IOCTL\_FIND\_FIRST\_FILE, IO\_IOCTL\_REMOVE\_SUBDIR

**Revision SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Date** | **Author** | **Description of Revision & Writer** | **Spec Coord.** |
| 15.6.2011 | Michal Starecek | Initial version |  |
| 21.6.2011 | Vladimir Vanek | Added comments and request for updates |  |