FAT File System Test Suite User Guide

Version 1.90

For use with FAT File System Test Suite version 2.30

Table of Contents

1. System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

1.1. Introduction

This guide is for those who want to test an HCC Embedded FAT file system.

This test suite allows you to test many file system functions, including the following:

- File creation and writing.
- Checking file content.
- Handling of "a", "a+", "w", and "w+" files.
- Concurrent access with "r" files.
- Formatting.
- Power failure.
- Finding and seeking.
- Renaming.
- Bad block detection.
- Partitions.
- Volume cleaning.
- Finding and deletion of Shift JIS Japanese character set files.
- File and directory operations with Shift JIS characters.
- File Multiple Write (FMW) functions (if FAT_FMW_ENABLE is set in the main FAT package).

1.2. Feature Check

The main features of the module are the following:

- Provides a full test capability for HCC Embedded FAT file systems.
- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to the HCC Coding Standard.

1.3. Packages and Documents

Packages

The table below lists the packages that you need in order to use this module.

Package	Description
hcc_base_docs	This contains the two guides that will help you get started.
fs_fat_test	The FAT file system test package.
fs_fat_fmw_test	The File Multiple Write (FMW) test package.
media_drv_base	The Media Driver base package that provides the base for all media drivers that attach to the file system.
psp_template_base	The Platform Support Package (PSP) base package.
oal_base	The OS Abstraction Layer (OAL) base package.

Documents

For an overview of HCC file systems and guidance on choosing a file system, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC FAT File System Test Suite User Guide

This is this document.

1.4. Change History

This section describes past changes to this manual.

- To download this manual or a PDF describing an earlier software version, see File System PDFs.
- For the history of changes made to the package code itself, see History: fs_fat_test and History: fs_fat_fmw_test.

The current version of this manual is 1.90. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.90	2020-07-21	2.30	Added TEST_WRITE_FULL configuration option.
1.80	2020-05-15	2.28	Added FMW tests. Added IS_SNPRINTF_AVAILABLE configuration option.
1.70	2020-02-28	2.26	Added TEST_VOLUME_CLEAN configuration option. Added more source code files. Added **psp_memcmp()** and **psp_printf()** to *PSP Porting*.
1.60	2020-01-24	2.24	TEST_PARTITIONS configuration option defaulted to 0.
1.50	2019-11-08	2.22	Added TEST_PARTITIONS configuration option.
1.40	2018-07-18	2.16	Added **psp_snprintf()** to *PSP Porting*.
1.30	2017-10-12	2.12	Updated *Packages* list.
1.20	2017-06-23	2.08	New *Change History* format.
1.10	2016-03-21	2.06	Added *Change History*.
1.00	2015-03-10	2.01	First online version.

2. Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

Note: Do not modify any files except the configuration file and PSP files.

API Header File

The file **src/api/api_fat_test.h** should be included by any application using the system. This is the only file that should be included by an application using this module. It defines the f_dotest() function.

Configuration File

The file **src/config/config_fat_test.h** contains all the configurable parameters of the system. Configure these as required. This is the only file in the module that you should modify. For details of these options, see Configuration Options.

Source Code Files

These files in the directory **src/fat/test** contain the source code. **These files should only be modified by HCC.**

File	Description
fat_test.c and **.h**	Source code and header file.
fat_test_volcl.c and **.h**	Volume cleaning test source code and header file.

The FAT FMW package has these files in the directory **src/fat/fmw_test**:

File	Description
fat_fmw_test.c and **.h**	Source code and header file.

PSP Files

These files in the directory **src/psp/target/fat_test** provide functions and elements that the core code needs to use, depending on the hardware. Modify these files as required for your hardware; see Setting up Your Drive.

File	Description
fat_testport.c	Source code.
fat_testport.h	Header file.

Version Files

The file **src/version/ver_fat_test.h** contains the version number of this module. This version number is checked by all modules that use this module to ensure system consistency over upgrades.

The FAT FMW package has a version file **ver_fat_fmw_test.h**.

3. Configuration Options

Set the system configuration options in the file **src/config/config_fat_test.h**. This section lists the available configuration options and their default values.

> **Note:** The first two options only apply if USE_MALLOC is set.

FN_MAXVOLUME

The number of volumes. This is only needed for library builds. The default value is 3.

TEST_MALLOC_MAXFILES

Set this to 1 if you want to check the maximum number of files. The default value is 0.

TEST_PARTITIONS

Set this to 1 to enable the partition handling test. The default value is 0.

TEST_FAT_MEDIA

The default value is F_FAT32_MEDIA.

MAX_BUFFERSIZE

The size of the buffer used for test functions. A smaller buffer size results in fewer tests. Valid values are: 256, 512, 1024, 2048, 8192, 16384 and 32768. The default value is 32768.

- For the *f_writing* test set at least 2560.
- For the *f_seekpos* test set at least 8192 bytes. Lower values will result in fewer *f_seeking* tests.

RIT_NUM_OF_RECORDS

The number of RIT tests. The default value is 100.

TEST_OPEN_NONSAFE

Set this if non-safe open needs to be tested. (This is only valid if safe mode is allowed). The default value is 0.

> **Note:** The following two tests can be very slow on larger volumes as they write to all clusters.

TEST_VOLUME_CLEAN

Set this to 1 to enable testing of the **f_volume_clean()** function.

TEST_WRITE_FULL

Set this to 1 to enable testing of write media full.

IS_SNPRINTF_AVAILABLE

Keep the default value of 0 if the **snprinf()** functionality is not implemented in the standard library used. Otherwise, set it to 1.

4. Running Tests

This section shows how to run a test.

4.1. f_dotest Function

Use this function to run tests on a volume.

If FAT_FMW_ENABLE is set in the main FAT package, this function calls the FMW tests as well.

Format

```
void f_dotest ( uint16_t vol_id )
```

Arguments

Argument	Description	Type
vol_id	The ID of the volume to test.	uint16_t

Return Values

None.

4.2. Setting up Your Drive

Editing fat_testport.c

To set up your drive for testing, modify the two lines in the PSP template file **fat_testport.c** described below.

- Line 41 initially appears as shown below. Edit this line to include the API file for the drive you want to test.

```
#include "../../../api/api_mdriver_ram.h"
```

- Line 67 initially appears as shown below. Edit this line to change *ram_initfunc* to the **init()** function of the drive that you want to test. This should be exposed in the API file included at line 41.

```
ret = f_initvolume( (int)vol_id, ram_initfunc, vol_id );
```

This table shows examples of the **init()** functions used for the main drive types:

Drive type	Function
RAM	**ram_initfunc()**
Compact Flash (CFC)	**cfc_initfunc()**
MMC	**mmc_initfunc()**
Mass Storage	**mst_initfunc()**

These can be seen in the **f_initvolume()** function.

Selecting Functions to Print Test Results

The **f_dump()** and **f_result()** functions in **fat_testport.c** call **printf()** to output the test resuts. You can replace these **printf()** calls with calls to appropriate functions that output to your required location.

4.3. Starting the Test

Use the function **f_dotest()** to run the tests on the specified volume. The following example code shows how to do this:

```
{
    fs_init();      /* Initialize the file system */
    fs_start();     /* Start the file system */
    f_enterFS();    /* Register the task with the file system */
    f_dotest( 0 );  /* Execute the test suite on the first volume */
.....
}
```

5. Integration

The test suite is designed to be as open and as portable as possible. No assumptions are made about the functionality, the behavior, or even the existence, of the underlying operating system. For the system to work at its best, perform the porting outlined below. This is a straightforward task for an experienced engineer.

PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The test suite makes use of the following standard PSP functions:

Function	Package	Element	Description
psp_getrand()	psp_base	psp_rand	Gets a random number.
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.
psp_printf()	psp_base	psp_stdio	Prints a string.
psp_snprintf()	psp_base	psp_stdio	Sends formatted output to compose a string holding the same text that would be printed if *format* was used on **psp_printf()**. Instead of being printed, the content is stored as a *C string* in a buffer.
psp_strncmp()	psp_base	psp_string	Compares two strings of defined length.
psp_strncpy()	psp_base	psp_string	Copies one string to another.
psp_strnlen()	psp_base	psp_string	Gets the length of a string.

The module makes use of the following standard PSP macro:

Macro	Package	Element	Description
PSP_WR_BE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as big-endian to a memory location.

6. Version

Version 1.90

For use with FAT File System Test Suite version 2.30