MMC and SD Media Driver for STM32 User Guide

Version 1.50

For use with MMC and SD Media Driver for STM32 versions 2.15 and above

Date: 18-Aug-2017 14:26

Table of Contents

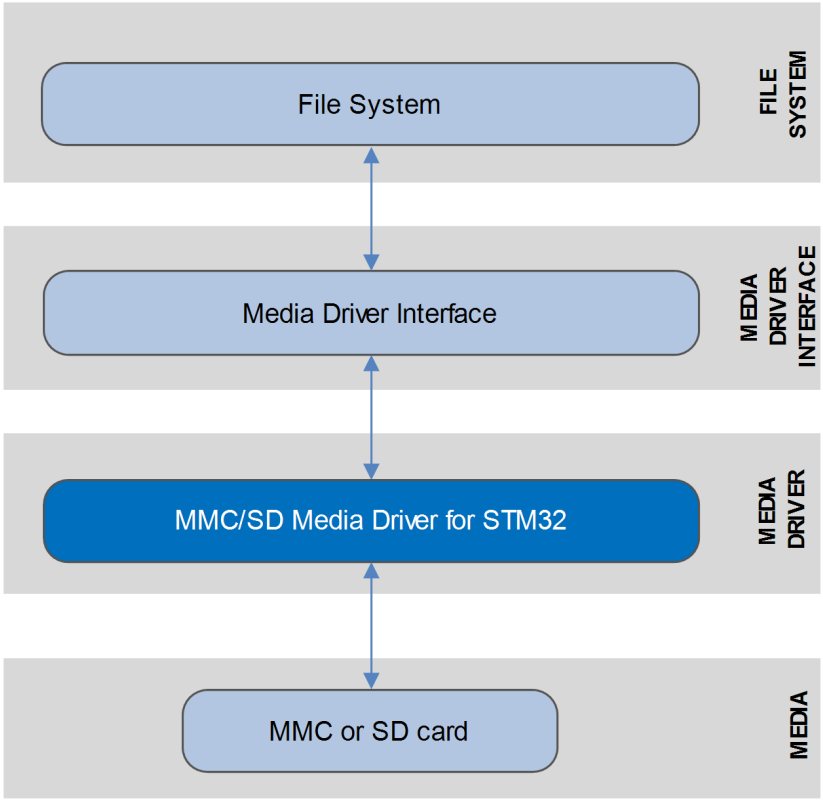
1 System Overview

1.1 Introduction

This guide is for those who want to use HCC Embedded's MMC/SD Media Driver for STM32 in their system. This module provides a media driver for STM32 microcontrollers from NXP Semiconductors. This guide covers all aspects of configuration and use.

This media driver conforms to the *HCC Media Driver Interface Specification*. It provides an interface for a file system to read from and write to Secure Digital (SD) or MultiMediaCard (MMC) storage devices. The file system handles all drives identically, regardless of their internal design features.

The diagram below shows a typical system architecture including a file system, media driver and media.



1.2 Feature Check

The main features of the media driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to the *HCC Media Driver Interface Specification*.
- Supports 1 bit, 4 bit, and 8 bit transfer width.
- Supports Direct Memory Access (DMA) transfers.
- The voltage range is configurable.
- Supports eMMC (embedded MMC).

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_doc	This contains the two guides that will help you get started.
media_drv_mmcsd_stm32	The media driver package described in this document.
media_drv_base	The base media driver package that includes the framework all media drivers use.
oal_base	The OS Abstraction Layer (OAL) base package.
util_hcc_mem	The HCC memory management utility.
media_drv_mmcsd_init	The generic **mmcsd_initcad()** routine.
media_drv_mmcsd_emmc_management	The EMMC Management extension, required if eMMC support is needed.

Documents

For an overview of HCC file systems and data storage, 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 Media Driver Interface Guide

This document describes the HCC Media Driver Interface Specification.

HCC MMC and SD Media Driver for STM32 User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To view or download earlier manuals, see Archive: MMC and SD Media Driver for STM32 User Guide .
- For the history of changes made to the package code itself, see History: media_drv_mmcsd_stm32.

The current version of this manual is 1.50. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.50	2017-08-18	2.15	Updated *Packages* list.
1.40	2017-06-23	2.14	New *Change History* format.
1.30	2016-02-02	2.11	Various small changes.
1.20	2015-11-23	2.11	Added functions to *PSP Porting*.
1.10	2015-05-11	2.11	Various small changes.
1.00	2015-03-24	2.11	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.

2.1 API Header File

The file **src/api/api_mdriver_mmcsd.h** is the only file that should be included by an application using this module. For details of the single API function, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_mdriver_mmcsd.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.

2.3 System Files

The following files in the directory **src/media-drv/mmcsd** are the source code for the media driver. **These files should only be modified by HCC**.

File	Description
mmcsd.c	MMC/SD source file.
mmcsd_init.c	MMC/SD initialization source file.
mmcsd_init.h	MMC/SD initialization header file.

2.4 Version Files

The following files in the directory **src/version** contain the version numbers of components of the module. The version numbers are checked by all modules that use a module to ensure system consistency over upgrades.

File	Description
ver_mdriver_mmcsd.h	Module version file.
ver_mmcsd_init.h	MMC/SD initialization version file.

2.5 Platform Support Package (PSP) Files

These files provide functions and other elements the core code needs to call, depending on the hardware. Modify these files as required for your hardware.

> **Note:** You must modify these PSP implementations for your specific microcontroller and development board; see PSP Porting for details.

The following files are in directories under **src/psp_stm32xxxx/hcc/src/psp/target**. There are directories for **psp_stm32f20x**, **psp_stm32f40x**, **psp_stm32f7xx** and **psp_stm32l1xx**.

File	Description
dma/dma.c	DMA source code.
dma/dma.h	DMA function definitions.
include/hcc_stm32fxxx_regs.h	Register definitions for the device.
mmcsd/mmcsd_hw.c	Hardware functions code.
mmcsd/mmcsd_hw.h	Hardware functions header file.

3 Configuration Options

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

MMCSD_NUM_UNITS

The number of MMC/SD channels. Do not change this value from the default of 1; it is always 1 for STM32Fxxx.

MMCSD1_VOLTAGE_RANGE_170_195, MMCSD1_VOLTAGE_RANGE_270_360

These two options set the voltage range in which the signals of unit 1 operate.

Set the first to 1 to set the valid voltage range to 1.7 - 1.95V. Its default is 0.

Set the second to 1 to set the valid voltage range to 2.7 - 3.6V. This is the default.

MMCSD1_ALLOW_4BIT, MMCSD1_ALLOW_8BIT

To set the mode to 4 bit, keep the first option at the default of 1, and the second at 0. To set 8 bit mode, reverse the settings.

MMCSDx_SPEED_LIMIT

Set this to the maximum desired frequency in kHz when testing prototype boards with wiring that supports only lower speeds.

Keeping this at 0 (the default) disables the speed limit.

MMCSD_DMA_SECTORS

The number of sectors the Direct Memory Access (DMA) buffer holds. The default is 4.

The DMA buffer is required for non-aligned transfers only.

MMCSD_ALLOW_EMMC_MANAGEMENT

Set this to 1 if the eMMC management module is used. The default is 0.

The eMMC module provides features including the following:

- Get/set card partitioning settings.
- Get/set Reliable Write settings for User Data Area and partitions.
- Get card health status.

4 Application Programming Interface

This section describes the single function, the structure it uses, and the error codes.

When the media driver is used:

1. The file system calls the media driver's **mmcsd_initfunc()** function.
2. **mmcsd_initfunc()** returns a pointer to an F_DRIVER structure containing a set of functions for accessing the media driver.

4.1 mmcsd_initfunc

Use this function to initialize the interface with the driver.

The caller passes a parameter to the initialization function of a conforming driver. The driver returns a pointer to an F_DRIVER structure defining the interface to that driver.

> **Note:** The call must allocate or use a static structure for the F_DRIVER structure. It must return a pointer to this structure, which must contain all the driver entry points, and also other data as required.

Format

```
F_DRIVER * ( mmcsd_initfunc )( unsigned long driver_param )
```

Arguments

Argument	Description	Type
driver_param	This identifies the drive to use. The first drive is 0. This cannot be greater than MDRIVER_MAX_VOLUME - 1.	unsigned long

Return values

Return value	Description
F_DRIVER *	A pointer to the driver structure, or NULL if the request failed.

4.2 F_DRIVER Structure

This is the format of the *F_DRIVER* structure. This structure is defined in the *HCC Media Driver Interface Specification*.

Element	Type	Description
separated	int	Non-zero if the driver is separated.
user_data	unsigned long	User-defined data.
user_ptr	void *	User-defined pointer.
writesector	F_WRITESECTOR	Write a sector to the drive. This is mandatory if format or any write access is required.
writemultiplesector	F_WRITEMULTIPLESECTOR	Write a series of sectors to the drive. If this is unavailable F_WRITESECTOR may be used.
readsector	F_READSECTOR	Read a sector from the drive.
readmultiplesector	F_READMULTIPLESECTOR	Read a series of sectors from the drive. If this is unavailable F_READSECTOR may be used.
getphy	F_GETPHY	Used to get the physical properties of the drive, such as the number of sectors.
getstatus	F_GETSTATUS	(Only for removable drives) Used to test whether a drive has been removed or changed.
release	F_RELEASE	Release any resources associated with a drive when it is freed by the host (file) system.
ioctl	F_IOCTL	Used to send user-defined messages to the driver and get a response.

4.3 Error Codes

If a function executes successfully, it returns with MMCSD_NO_ERROR, a value of zero. The following table shows the meaning of the error codes.

Return Value	Value	Description
MMCSD_ERR_NOTPLUGGED	-1	For high level.
MMCSD_NO_ERROR	0	Successful execution.
MMCSD_ERR_NOTINITIALIZED	101	Driver not initialized.
MMCSD_ERR_INIT	102	Initialization error.
MMCSD_ERR_CMD	103	Command error.
MMCSD_ERR_STARTBIT	104	Start bit error.
MMCSD_ERR_BUSY	105	Driver already active.
MMCSD_ERR_CRC	106	CRC error.
MMCSD_ERR_WRITE	107	Write error.
MMCSD_ERR_WRITEPROTECT	108	Media is write-protected.
MMCSD_ERR_NOTAVAILABLE	109	Media is not available.

5 Integration

This section describes all aspects of the module that require integration with your target project. This includes porting and configuration of external resources.

5.1 OS Abstraction Layer

All HCC modules use the OS Abstraction Layer (OAL). This allows modules to run seamlessly with a wide variety of RTOSes, or without an RTOS.

This module uses the following OAL components:

OAL Resource	Number Required
Tasks	0
Mutexes	1
Events	0

5.2 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 media driver makes use of the following standard PSP functions:

Function	Package	Component	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.

The module makes use of the following PSP functions. These functions are provided by the PSP to perform various tasks. Their design makes it easy for you to port them to work with your hardware solution. The package includes samples in the PSP file **src/psp_xxx/target/mmcsd/mmcsd_hw.c** (where *xxx* is stm32f20x or stm32f40x):

Function	Description
mmcsd_hw_init()	Initializes the hardware (clocks, GPIO, and so on).
mmcsd_hw_delete()	Deletes the device, releasing the associated resources.
mmcsd_hw_power()	Powers on the device. This only returns when the power level is OK.
mmcsd_hw_get_speed()	Gets the SDIO clock frequency in kHz.
mmcsd_hw_set_speed()	Sets the SDIO clock frequency in kHz.
mmcsd_hw_dwidth()	Sets the MMC/SD data bus width.
mmcsd_hw_get_cd()	Gets the status of the CD pin. This is non-zero if no card is connected.
mmcsd_hw_get_wp()	Checks whether a card's write protect switch is on. This returns zero if it is off.
mmcsd_hw_drive_cmd()	Selects push-pull or open-drain mode on the SDIO_CMD pin.

These functions are described in the following sections.

mmcsd_hw_init

This function is provided by the PSP to initialize the device.

This enables the clocks, GPIO pin, and so on.

Format

```
int mmcsd_hw_init ( void )
```

Arguments

None.

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR_INIT	Operation failed.

mmcsd_hw_delete

This function is provided by the PSP to delete the device, releasing the associated resources.

Format

```
int mmcsd_hw_delete ( void )
```

Arguments

None.

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.

mmcsd_hw_power

This function is provided by the PSP to turn on the card's power.

This call blocks: it only returns when the power level is correct.

> **Note:** On the default development board the MMC/SD power cannot be turned off, so this call has no effect.

Format

```
void mmcsd_hw_power ( int on )
```

Arguments

Parameter	Description	Type
on	The power setting.	int

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.

mmcsd_hw_get_speed

This function is provided by the PSP to get the SDIO unit's clock frequency in kHz.

Format

```
uint32_t mmcsd_hw_get_speed ( uint32_t uid )
```

Arguments

Parameter	Description	Type
uid	The MMC/SD unit ID. This is ignored in this implementation.	uint32_t

Return Values

Return value	Description
The SDIO clock frequency	Successful execution.
MMCSD_ERROR	Operation failed.

mmcsd_hw_set_speed

This function is provided by the PSP to configure the SDIO unit to generate a specific clock frequency.

Format

```
void mmcsd_hw_set_speed (
    uint32_t   uid,
    uint32_t   khz )
```

Arguments

Parameter	Description	Type
uid	The unit ID. This is ignored in this implementation.	uint32_t
khz	The desired frequency in KHz.	uint32_t

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.

mmcsd_hw_dwidth

This function is provided by the PSP to set the MMC/SD data bus width.

Format

```
void mmcsd_hw_dwidth (
    uint32_t   uid,
    uint32_t   n_bits )
```

Arguments

Parameter	Description	Type
uid	The unit ID. This is ignored in this implementation.	uint32_t
n_bits	The width of the data bus.	uint32_t

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.

mmcsd_hw_get_cd

This function is provided by the PSP to check whether an MMC/SD card is present.

This returns the state of the CD pin.

Format

```
int mmcsd_hw_get_cd ( void )
```

Arguments

None.

Return Values

Return value	Description
0	No card is present.
1	A card is present.

mmcsd_hw_get_wp

This function is provided by the PSP to check a card's write-protect state.

> **Note:** On the default development board the write-protect switch is not connected to the MCU, so this call always returns non-protected status.

Format

```
int mmcsd_hw_get_wp ( void )
```

Arguments

None.

Return Values

Return value	Description
0	The card is not write-protected.
1	The card is write-protected.

mmcsd_hw_drive_cmd

This function is provided by the PSP to select push-pull or open-drain mode on the CMD pin.

Format

```
void mmcsd_hw_drive_cmd (
    uint32_t   uid,
    uint32_t   push_pull )
```

Arguments

Parameter	Description	Type
uid	The unit ID. This is ignored in this implementation.	uint32_t
push_pull	Set this to 1 to activate push-pull mode. Otherwise, the pin is in open-drain mode.	uint32_t

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.