MMC and SD Media Driver for TI AM37x User Guide

Version 1.20

For use with MMC and SD Media Driver for TI Am37x versions 2.05 and above

Date: 23-Jun-2017 11:56

Table of Contents

System Overview 3
Introduction 3
Feature Check 4
Packages and Documents 5
Packages 5
Documents 5
Change History 6
Source File List 7
API Header File 7
Configuration File 7
Source Code 7
Platform Support Package (PSP) Files 7
Version Files 8
Configuration Options 9
Application Programming Interface 10
mmcsd_initfunc 10
F_DRIVER Structure 11
Error Codes 12
Integration 13
OS Abstraction Layer 13
PSP Porting 14
psp_tick_init 15
psp_get_tick_count 16

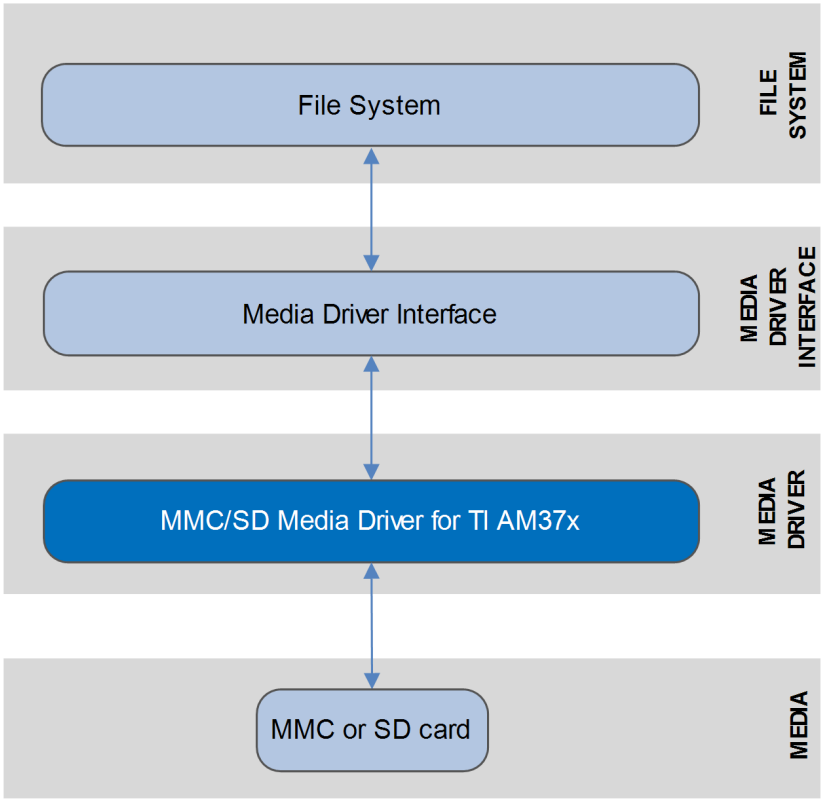
1 System Overview

1.1 Introduction

This guide is for those who want to use HCC Embedded's MMC and SD Media Driver for AM37x processors from Texas Instruments Incorporated. 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 multiple card types: MMC and SD and also SDHC (SD High Capacity) and SDXC (SD eXtended Capacity).
- 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_base	The base media driver package that includes the framework all media drivers use.
media_drv_mmcsd_am37x	The media driver package described in this document.

Documents

For an overview of HCC file systems and data storage, refer to the Product Information section of 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 TI Am37x 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 TI AM37x User Guide.
- For the history of changes made to the package code itself, see History: media_drv_mmcsd_am37x.

The current version of this manual is 1.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.20	2017-06-23	2.05	New *Change History* format.
1.10	2015-05-11	2.05	Various small changes.
1.00	2015-03-20	2.05	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. For details of these options, see Configuration Options.

2.3 Source Code

The file **src/media-drv/mmcsd/am37x/mmcsd.c** holds the source code for the media driver. **This file should only be modified by HCC**.

2.4 Platform Support Package (PSP) Files

These files in the directory **src/psp** provide functions and elements the core code needs to use, depending on the hardware.

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

File	Description
include/psp_tick.h	Timer setup.
target/tick/psp_tick.c	Timer source code.
target/include/hcc_am37x_reg.h	Defines the register settings.

2.5 Version Files

These files in **src/version** contain the version numbers for this module. The version number is checked by all modules that use a module to ensure system consistency over upgrades.

File	Description
ver_mdriver_mmcsd.h	Main version number.
ver_psp_proc_reg.h	Version number for the register file.

3 Configuration Options

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

MMCSD1_VOLTAGE_RANGE_170_195

Set this to 1 to set the voltage range in which the MMCHS1 signals operate to 1.7-1.95V. The default is 0.

MMCSD1_VOLTAGE_RANGE_270_360

Keep this at the default of 1 to set the voltage range in which the MMCHS1 signals operate to 2.7-3.6V. If you enable the above option, set this to 0.

MMCSD2_VOLTAGE_RANGE_170_195

Set this to 1 to set the voltage range in which the MMCHS2 signals operate to 1.7-1.95V. The default is 0.

MMCSD2_VOLTAGE_RANGE_270_360

Keep this at the default of 1 to set the voltage range in which the MMCHS2 signals operate to 2.7-3.6V. If you enable the above option, set this to 0.

MMCSD_SPEED_LIMIT

Set this to 1 when testing MMCHS2 with slow hardware, for example with HCC's daughterboard. The default of zero disables the speed limit.

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	0U	Successful execution.
MMCSD_ERR_NOTINITIALIZED	101U	Driver not initialized.
MMCSD_ERR_INIT	102U	Initialization error.
MMCSD_ERR_CMD	103U	Command error.
MMCSD_ERR_WRITEPROTECT	104U	Media is write-protected.
MMCSD_ERR_NOTAVAILABLE	105U	Media 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 module makes use of the following PSP functions. These functions are provided by the Platform Support Package (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/target/tick/ psp_tick.c**.

Function	Description
psp_tick_init()	Initializes the timer.
psp_tick_count()	Gets the current tick count.

These functions are described in the following sections.

psp_tick_init

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

The timer is ms-based.

Format

```
void psp_tick_init ( void )
```

Arguments

None.

Return Values

None.

psp_get_tick_count

This function is provided by the PSP to get the current tick count.

The tick resolution must be 1 ms.

Format

```
uint32_t psp_get_tick_count ( void )
```

Arguments

None.

Return Values

Return value	Description
Count	The tick count; successful execution.
MMCSD_ERR_CMD	Operation failed.