MMC and SD Media Driver for LPC User Guide

Version 1.70

For use with MMC/SD Media Driver for LPC versions 2.2 and above

Date: 18-Aug-2017 12:58

Table of Contents

1 System Overview

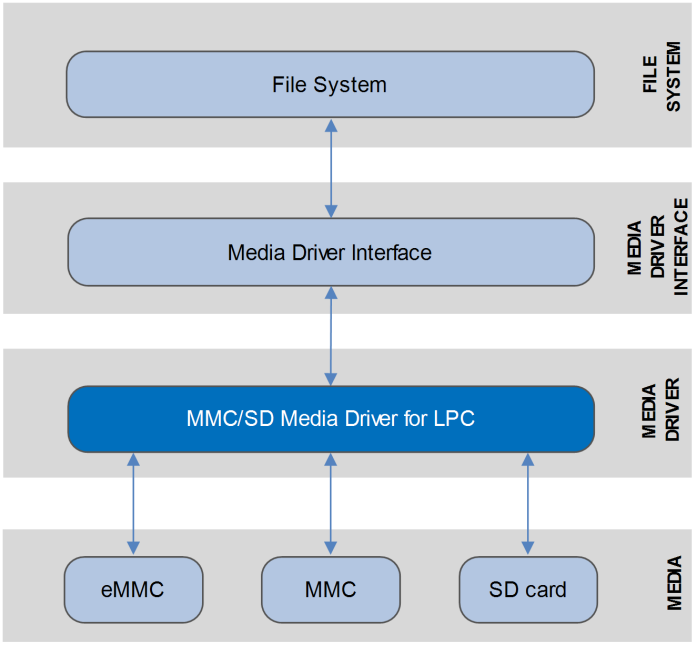
1.1 Introduction

This guide is for those who want to use HCC Embedded's MMC/SD Media Driver for LPC processors 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), MultiMediaCard (MMC), or eMMC (embedded MMC) storage devices. A single media driver can support one or more physical media, each of these being represented as a different drive at the media driver interface. The file system handles all drives identically, regardless of their internal design features.

If eMMC is used, this media driver can be used with HCC's eMMC Management Driver. This is an extension to HCC's MMC and SD media drivers and is independent of any particular micro-controller and its MMC/SD controller. For details, see the *HCC eMMC Management Extension for MMC and SD Drivers User Guide*.

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) and can be used with HCC's eMMC Management Driver.
- Supports 1 and 4 bit modes.
- Supports Direct Memory Access (DMA) transfers.
- The voltage range is configurable.

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_lpc	The media driver package described in this document.
media_drv_mmcsd_init	The generic **mmcsd_initcad()** routine.
oal_base	The base OS Abstraction Layer (OAL) package.
psp_template_base	The base Platform Support Package (PSP).
util_hcc_mem	The HCC memory management utility.

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 eMMC Management Extension for MMC and SD Drivers User Guide

This document describes HCC's embedded MMC extension.

HCC MMC and SD Media Driver for LPC 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 LPC User Guide.
- For the history of changes made to the package code itself, see History: media_drv_mmcsd_lpc.

The current version of this manual is 1.70. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.70	2017-08-18	2.2	Updated *Packages* list.
1.60	2017-06-23	2.2	New *Change History* format.
1.50	2017-04-24	2.2	Changed *Feature Check*.
1.40	2016-07-20	2.2	Removed some PSP functions.
1.30	2016-02-02	2.1	Various small changes.
1.20	2016-01-05	2.1	Added functions to *PSP Porting*.
1.10	2015-05-08	1.12	Various small changes.
1.00	2015-03-20	1.10	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

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

File	Description
gdma.c	Generic Direct Memory Access (DMA) source file.
gdma.h	Generic DMA header file.
gdmaregs.h	Generic DMA registers.
mmcsd.c	MMC/SD source file.
mmcsd.h	MMC/SD header file.
mmcsd_regs.h	MMC/SD registers.

2.4 Version File

The file **src/version/ver_mdriver_mmcsd.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.

2.5 Platform Support Package (PSP) Files

These files in the directory **src/psp/target/mmcsd** provide functions the core code needs to call, 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
psp_mmcsd.c	Source code.
psp_mmcsd.h	Function definitions.

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.

MMCSD_TARGET_CPU

Select an LPC type from the list: LPC32XX, LPC23XX, LPC24XX and LPC17XX. The default is LPC32XX.

MMCSD_NUM_UNITS

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

PLL_CLK_IN_KHZ

The ARM_CLK in KHz. The default is 208000.

MMCSD1_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.

MMCSD1_VOLTAGE_RANGE_270_360

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

MMCSD1_ALLOW_4BIT

Keep this at the default of 1 to enable 4 bit mode.

MMCSD1_ALLOW_8BIT

Keep this at the default of 0. 8 bit mode is not supported by LPCs.

MMCSD1_SPEED_LIMIT

Set this to the maximum desired frequency in kHz when testing prototype boards whose wiring supports only lower speeds, for example HCC's daughterboard. The default of 0 disables the speed limit.

MMCSD_ALLOW_RELIABLE_WRITE

Set this to 1 if Reliable Write is used for all eMMC (embedded MMC) writes instead of normal block write. The default is 0.

Note:

- Reliable Write is slower than normal block write.
- Currently only enhanced reliable write is supported, not the legacy type of Reliable Write.

DMA_CHANNEL

The DMA channel to use. The default is 0.

Pin sets

Use the following three options to set the right setting for your board. Set the macro value to 1 to use the left-hand set of pins in the table below. Set the macro value to 2 to use the right-hand set of pins.

MCI_PINSET1

The default is 1.

MCI_PINSET2

The default is 2.

MCI_PINSET

The default is MCI_PINSET1.

The LCP23(64/66/68/78) chips can use only the first row of GPIO pins for the MCI. The LPC2468/78 can use the second row too.

	MCI_PINSET1	MCI_PINSET2
MCICLK	P0.19	P1.2
MCICMD	P0.20	P1.3
MCIPWR	P0.21	P1.5
MCIDAT0	P0.22	P1.6
MCIDAT1	P2.11	P1.7
MCIDAT2	P2.12	P1.11
MCIDAT3	P2.13	P1.12

USE_CD

Set this to 0 (the default) if the Card-Detect (CD) of the MMC/SD card is not connected to the chip.

PORT_CD

Specify which port CD is connected to. The default is 0.

PIN_CD

Specify the pin on the above port.

PIN_VAL_CD

Specify the value of the pin if CD is active.

USE_WP

Set this to 0 (the default) if the write-protect (WP) of the MMC/SD card is not connected to the chip.

PORT_WP

Specify the port WP is connected to.

PIN_WP

Specify the pin on the above port.

PIN_VAL_WP

Specify the value of the pin if WP is active.

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 **mmcsd_initfunc()** 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 incorrect.
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_DATAOK	109	Data valid.
MMCSD_ERR_RX	110	Receive error.
MMCSD_ERR_NOTAVAILABLE	111	Not available.

5 Integration

This section specifies the single element of this package that needs porting, depending on the target environment.

5.1 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 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/target/mmcsd/psp_mmcsd.c**:

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.
mmcsd_hw_get_cd()	Gets the card status, non-zero if the card is powered on.
mmcsd_hw_get_wp()	Checks whether a card's write protect switch is on. This returns 0 if it is off.
mmcsd_hw_drive_d0()	Enables or disables pull_up on the D0 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_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_d0

This function is provided by the PSP to enable/disable pull-up on the D0 pin.

Format

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

Arguments

Parameter	Description	Type
uid	The unit ID. This is ignored in this implementation.	uint32_t
pull_up	Set this to 1 to enable pull-up on D0. Set it to 0 to disable it.	uint32_t

Return Values

Return value	Description
MMCSD_NO_ERROR	Successful execution.
MMCSD_ERROR	Operation failed.