USB Device MTP Class Driver User Guide

Version 1.20

For use with USBD MTP Class Driver versions 2.14 and above

Exported on 02/01/2019

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.

Note: To download this manual as a PDF, see USB Device PDFs.

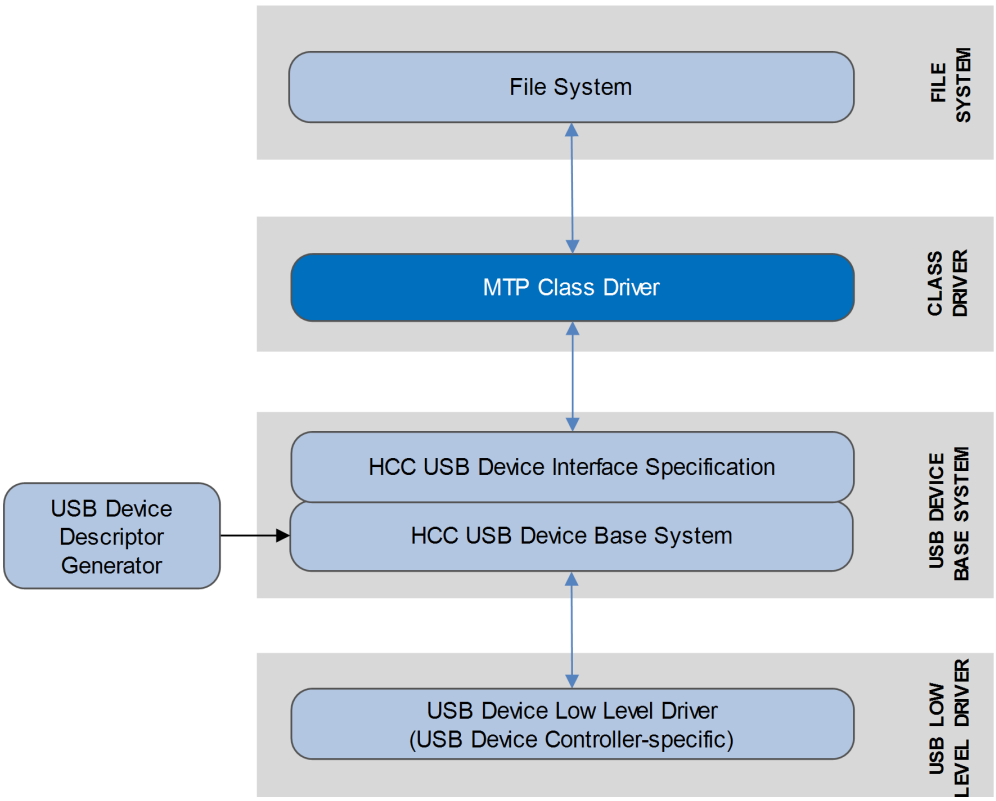
1.1 Introduction

This guide is for those who want to implement an embedded USB Media Transfer Protocol (MTP) class driver. This class driver can be integrated with HCC's FAT, THIN, TINY, and SafeFLASH file systems. The file system appears as though it is a drive on a PC.

This class driver supports two protocols:

- Media Transfer Protocol – used for atomic transfer of music and media files to/from portable devices. As a component of the Windows Media framework, MTP is associated with Windows Media Player.
- Picture Transfer Protocol (PTP) – used for atomic transfer of photograph files from digital cameras.

The system structure is shown in the diagram below:



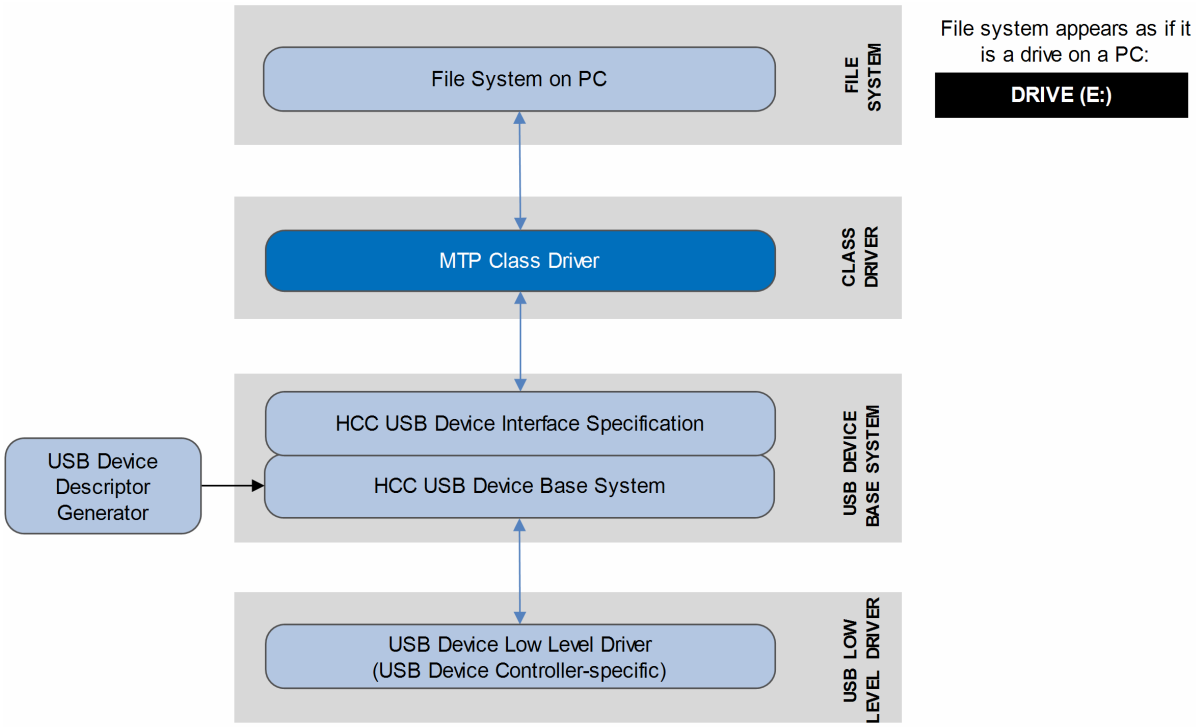
The package provides a set of API functions that are used to initialize and start the module and also set up directory protection. When a directory is protected its files can only be listed, not accessed, created, deleted, renamed, or copied.

Note:

- This module is part of HCC's Embedded USB Device (EUSBD) system, as described in the *HCC Embedded USB Device Base System User Guide*.
- This module communicates with the EUSBD base system through the EUSBD device interface, as described in the above manual.

To specify the type of HCC file system to use, simply set the MTP_FS_TYPE configuration option to one of the four MTP_FS_* values available. For example, to use SafeFLASH as the file system, set the type option to MTP_FS_SAFEFLASH. (Note that you can only use type of file system at a time.)

The system structure when the module is used with an HCC file system is shown in the diagram below:



1.2 Feature Check

The main features of the class driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Supports both Media Transfer Protocol (MTP) and Picture Transfer Protocol (PTP).
- Can be integrated with HCC's FAT, THIN, TINY, and SafeFLASH file systems.
- Supports use of the PictBridge standard for printing from digital cameras.
- Supports a test mode which creates test **.jpeg** files.
- Compatible with sample device files produced by using the HCC USB Device Descriptor Generator.

1.3 Packages and Documents

Packages

This table 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.
usbd_base	The USB device base package. This is the framework used by USB class drivers to communicate over USB using a specific USB device controller package.
usbd_cd_mtp	The MTP class driver described in this document.

Documents

For an overview of HCC's embedded USB stacks, 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 Embedded USB Device Base System User Guide

This document describes the Embedded USB Device base system.

HCC USB Device MTP Class Driver User Guide

This is this document.

HCC USB Device Descriptor Generator User Guide

This document describes the tool that creates USB descriptor files for inclusion in a project that uses the EUSBD stack.

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 USB Device PDFs.
- For the history of changes made to the package code itself, see History: usbd_cd_mtp.

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	2019-02-01	2.15	Corrected omission of refs. to HCC file systems in *Introduction* and *Feature Check*. Added *Configuration Options* from **MTP_SEND_FS_EVENT** to end of list.
1.10	2018-05-08	2.14	Added SafeFLASH files to *Source Files*. Changed default setting of MTP_PICTBRIDGE option.
1.00	2018-03-20	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.

2.1 API Header File

The file **src/api/api_usbd_mtp.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. For details of the API functions, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_usbd_mtp.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 following files in the directory **src/usb-device/class-drivers/mtp** provide interfaces to file systems. **These files should only be modified by HCC**.

File	Description
fat/mtp_fat.c	FAT functions source code.
fat/mtp_fat.h	FAT functions header file.
fat_thin/mtp_fat_thin.c	FAT THIN functions source code.
fat_thin/mtp_fat_thin.h	FAT THIN functions header file
safeflash/mtp_safeflash.c	SafeFLASH functions source code.
safeflash/mtp_safeflash.h	SafeFLASH functions header file.
tiny/mtp_tiny.c	TINY functions source code.
tiny/mtp_tiny.h	TINY functions header file.

The following files are in the directory **src/usb-device/class-drivers/mtp**. **These files should only be modified by HCC**.

File	Description
mtp.h	Main code for the MTP class driver.
mtp_abstract_storage.c and **.h**	Abstract storage code and header file.
mtp_copy_handler.c	Copy handler code.
mtp_dbg.c	Debugger code.
mtp_delete_object_handler.c	Delete object handler code.
mtp_device.c and **.h**	Device code and header file.
mtp_fs.h	File system header file.
mtp_get_device_info_handler.c	Get device information code.
mtp_get_device_prop_desc_handler.c	Get device property descriptor code.
mtp_get_device_prop_value.c	Get device property value code.
mtp_get_number_objects_handler.c	Object handler code (number of objects).
mtp_get_object_handler.c	Object handler code.
mtp_get_object_handlers_handler.c	Object handler code.
mtp_get_object_info_handler.c	Object information code.
mtp_get_object_prop_desc.c	Object property descriptor code.
mtp_get_object_prop_value.c	Object property value code.
mtp_get_object_props_supported_handler.c	Object properties supported code.
mtp_get_partial_object_handler.c	Partial object handler code.
mtp_get_storage_id_handler.c	Storage ID code.
mtp_get_storage_info_handler.c	Storage information code.
mtp_handlers.c and **.h**	Handlers code.
mtp_main.c	Main source code.
mtp_properties.c	Properties code.
mtp_reset_device_prop_value_handler.c	Reset device property code.

File	Description
mtp_set_object_prop_value.c	Set object property code.
mtp_set_object_protection_handler.c	Set object protection code.
mtp_transport.c and **.h**	Transport code and header file.
mtp_types.h	MTP types header file.
mtp_utils.h	MTP utilities header file.

2.4 Setup Information

The file **driver/usb_device/class-drivers/mtp/HccMtp.inf** contains the setup information. **This file should only be modified by HCC**.

2.5 Version File

The file **src/version/ver_usbd_mtp.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.

3 Configuration Options

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

USBD_MTP_TASK_STACK_SIZE

The MTP task stack size. The default is 4096.

MTP_TASK_DELAY

The time delay used. This specifies how often to check whether the MTP device is configured. The default is 10.

MTP_DATA_BUFFER_SIZE

The size of the MTP data buffer in bytes. This must be at least 1024 bytes if PictBridge is active. The default is 1024.

STANDARD_DEVICE_FUNCTIONAL_MODE

The functional mode. The default is 0x0000.

DEVICE_FRIENDLY_NAME

The device name.The default is "HCCMTPdevice".

DEVICE_SERIAL_NUMBER

The device serial number. The default is "00000000000000000000000012345678".

DEVICE_VERSION

The device version. The default is "V1.00.01".

DEVICE_MANUFACTURER

The device manufacturer. The default is "HCC Embedded".

DEVICE_MODEL

The device model. The default is "Storage".

MTP_F_MAXNAME

The maximum length of the filename including the path. The default is 128.

MTP_F_MAX_FILE_NAME_LENGTH

The maximum length of the filename without the path. The default is 128.

MTP_DEVICE_FRIENDLY_NAME_LENGTH

The length of the device friendly name (see above). The default is 32.

MTP_DATA_COMMUNICATION_ENABLED

Set this option to 1 to allow use of a user data communication (DCOM) channel. The default is 0.

DCOM_TODEVICE_FILENAME

The name of the file used for data communication (DCOM) to a device. The default is "TODEVICE.TXT".

DCOM_FROMDEVICE_FILENAME

The name of the file used for data communication (DCOM) from a device. The default is "FROMDEVICE.TXT"

MTP_PICTBRIDGE

Set this to 1 if the PictBridge function is used. The default is 0.

DPSR_DDISCVRY_DIR

The name of the directory where standard PictBridge files are stored. This directory will only be present if PictBridge is enabled. The default is "_DPSDIR".

DPSR_DREQUEST_OBJECT_HANDLE

The request object handle of a standard PictBridge file. The default is 0xFFFFFFF8.

DPSR_DRESPONSE_OBJECT_HANDLE

The response object handle of a standard PictBridge file. The default is 0xFFFFFFF9.

START_WITH_TESTJPEG

Set this to 1 if you want the drive to be formatted after startup, in which case a test .jpeg file is created. Use this for test purposes. The default is 0.

MTP_FORMAT_MEDIA_AT_STARTUP

Set this to 1 if you want the media to be formatted after startup. Use this for test purposes. The default is 0.

MTP_VENDORDEFINED_SCRIPT_FORMAT_CODE

This ObjectFormatCode is mainly used for PictBridge objects with value 0x3002. This is its default value.

This can be changed to any other value for vendor-specific implementations. Define vendor-specific format codes in the range 0xB000 - 0xB7FF.

MTP_COPYOBJECT_WITH_1_FILEOPENED

If this is set to the default of 1, only one file is opened at a time during CopyObject to save RAM. This can slow performance down if big files are transferred.

MTP_SENDOBJECT_ENABLED

Keep the default of 1 to enable transfer of PictBridge objects.

Do not forget to update MTP_OBJECT_PROPERTIES_NUMBER if you change this.

MTP_PROTECTED_DIRECTORY_ENABLED

Set this to 1 if you want to use a protected directory. The default is 0.

MTP_PROTECTED_DIRECTORY

The name of the directory that becomes protected when **mtp_set_dir_protection()** is called with a value of 1. When a directory is protected, its files can only be listed, not accessed, created, deleted, renamed, or copied. The default is "/protect".

MTP_FS_EVENT_COUNT

The maximum number of file system events. The default is 3.

MTP_SEND_FS_EVENT

Set this to 1 if you want file system events to be reported. The default is 0.

CAPTURE_FORMAT_NUMBER

Set this to the number of supported capture formats. The default is 0.

MTP_HIDDEN_PROP_SUPPORTED

Set this to 1 if you want to support the Hidden property. If not hidden, an object is displayed to a user browsing a device's contents. Hidden objects are hidden from browsing users but not hidden from other applications. The default is 0.

MTP_FS_FAT

Do not change this from the default, 0.

MTP_FS_FAT_THIN

Do not change this from the default, 1.

MTP_FS_TINY

Do not change this from the default, 2.

MTP_FS_SAFEFLASH

Do not change this from the default, 3.

MTP_FS_TYPE

Set this to one of the four MTP_FS_* values listed above. For example, to use SafeFLASH as the file system, set it to MTP_FS_SAFEFLASH.

4 Application Programming Interface

This section documents the Application Programming Interface (API). It includes all the functions that are available to an application program.

4.1 Functions

The functions are the following:

Function	Description
mtp_init()	Initializes the module and allocates the required resources.
mtp_start()	Starts the module.
mtp_set_dir_protection()	Enables directory protection for a specified directory.

mtp_init

Use this function to initialize the class driver and allocate the required resources.

> **Note:** You must call this before any other function.

Format

```
int mtp_init ( uint32_t mode )
```

Arguments

Parameter	Description	Type
mode	MTP or PTP mode.	uint32_t

Return Values

Return value	Description
MTP_OK	Successful execution.
Else	See Error Codes.

mtp_start

Use this function to start the class driver.

> **Note:** You must call **mtp_init()** before this to initialize the module.

Format

```
int mtp_start ( void )
```

Arguments

Parameter
None.

Return Values

Return value	Description
MTP_OK	Successful execution.
Else	See Error Codes.

mtp_set_dir_protection

Use this function to enable directory protection for the directory specified by the configuration option MTP_PROTECTED_DIRECTORY.

When a directory is protected, its files can only be listed, not accessed, created, deleted, renamed, or copied.

> **Note:** The configuration option MTP_PROTECTED_DIRECTORY_ENABLED must be set to 1 for this call to work.

Format

```
void mtp_set_dir_protection ( uint8_t set )
```

Arguments

Parameter	Description	Type
set	Set this to 1 to enable directory protection for the directory.	uint8_t

Return Values

Return value	Description
MTP_OK	Successful execution.
Else	See Error Codes.

4.2 Error Codes

If a function executes successfully, it returns with MTP_OK, a value of 0. The following table shows the meaning of the error codes.

Return Value	Value	Description
MTP_OK	0	Successful execution.
MTP_ERR_INIT	1	Initialization error.
MTP_ERR_FSINIT	2	File system initialization error.
MTP_ERR_IO	3	I/O error.
MTP_ERR_INV_REQ	4	Invalid request.

4.3 Types and Definitions

This section describes the MTP modes that are defined in the API header file.

MTP Modes

The following two modes are supported:

Type	Value	Description
MTP_MODE_MTP	1	Media Transfer Protocol (MTP) mode.
MTP_MODE_PTP	2	Picture Transfer Protocol (PTP) mode.

5 Integration

This section specifies the elements of this package that need porting, depending on the target environment.

5.1 OS Abstraction Layer

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

The module uses the following OAL components:

OAL Resource	Number Required
Tasks	1
Mutexes	0
Events	3

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. For full details of its functions, see the *HCC Base Platform Support Package User Guide*.

The module makes use of the following standard PSP functions:

Function	Package	Element	Description
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.
psp_strncat()	psp_base	psp_string	Appends a string.
psp_strnicmp()	psp_base	psp_string	Compares two strings of defined length.
psp_strncpy()	psp_base	psp_string	Copies one string of defined length to another.
psp_strnlen()	psp_base	psp_string	Gets the length of a string.