AN1475
APPLICATION NOTE

DEVELOPING AN ST7265x MASS STORAGE APPLICATION

By Microcontroller Division Applications

INTRODUCTION

This application note describes how to develop a USB Mass Storage application using an ST7265x microcontroller.

The proposed solution includes a generic USB Mass Storage Layer (MSL) compliant with the USB Mass Storage Class that can be used for all types of storage media (FLASH memory cards, hard disk drive, etc.) and a dedicated Media Access Layer (MAL) add-in for each type of storage media.

The MSL and the MAL modules are available to ST customers on request. Contact your ST sales office to obtain the ST7265 Mass Storage software.

The MSL and the MAL must be considered as function libraries and therefore cannot be modified by the user. The software interfaces of these function libraries provide the user with the required flexibility for designing his application:

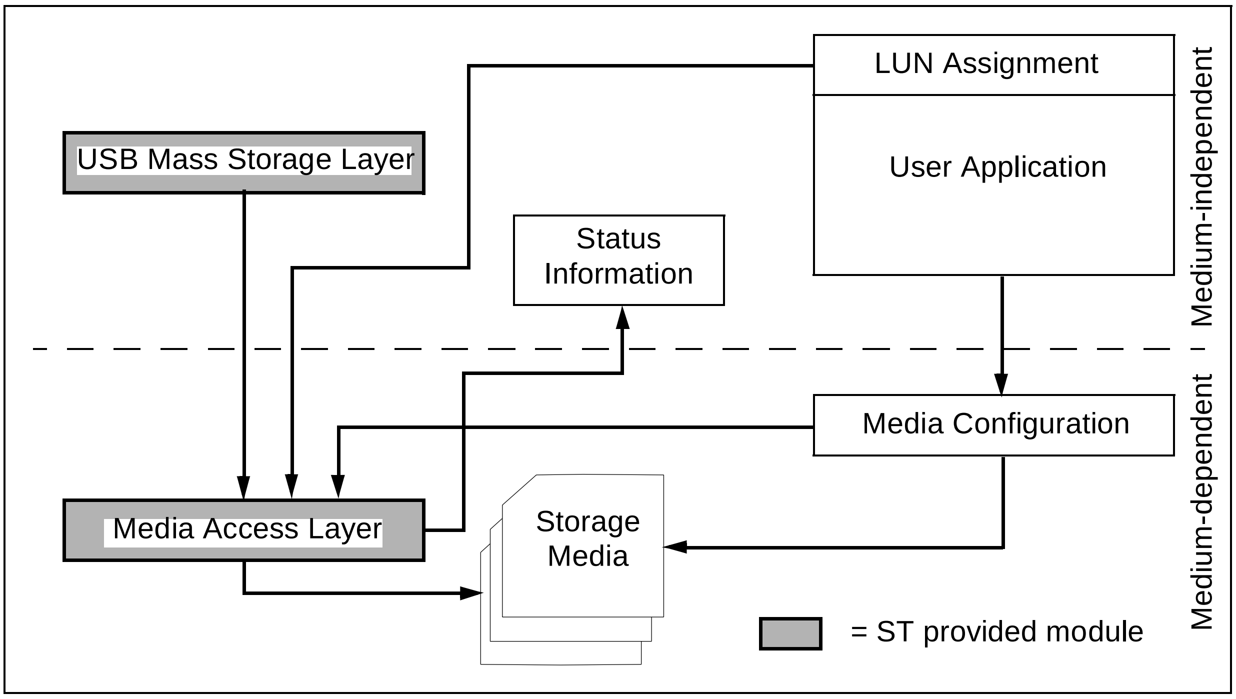
– type and number of media,
– insertion/withdrawal management,
– protection management,
– user interface.

Note that the MAL requires the use of certain fixed MCU pins to interface with the various mediums. As a result, a specific application hardware configuration is required when using the MAL.

1 SOFTWARE ARCHITECTURE

The ST7265 software is designed in modules. Each module has a unique and clearly defined interface. This architecture enables the user to easily replace any module and to reuse the individual module in other projects. This architecture provides maximum flexibility so users are able to customize their application as they want. The module software design is showed in Figure 1.

Figure 1. ST7265 Mass Storage Application Architecture



All the components described above are grouped into a project. The “Status Information” interface enables the user to monitor the execution of the software. The “Media Configuration” interface enables the user to customize the application configuration.

2 PROJECT CONTENT

Files required for the various modules are grouped into different directories. Most files are user application independent; i.e. they are not designed to be modified by the user application. Other files are application dependent and can be modified by the user. Table 1 explains the usage of the files included in the project.

Table 1. Application Files

Directory	Path/File		Usage
MCL (Application specific)	Appli.c, Appli.h		Application-related routines (called from main.c) to be developed by the user.
	Descript.c, Descript.h		These files contain all the descriptors for the USB enumeration. These files are designed to be modified by the user depending on the user application.
	MAL_Func.c		This file contains the global Media Access Library settings. These files are designed to be modified by the user depending on the user application.
	Mconfig.h		Definitions of the software interface described in the document. These files are designed to be modified by the user depending on the user application.
	Int_7265.c, Int_7265.h		Groups all interrupt routines. These files areto be modified by the user depending on the user application.
MAS Sto	Micro	Cond-comp.h	Conditional compilation definitions. This file is used to configure the USB behaviour. This file is not designed to be modified by the user.
		Main.c	Main software entry point. This file is not designed to be modified by the user (see Appli.c)
		User_lib.c, User_lib.h	USB routines related to this application. These files are not designed to be modified by the user.
	MAL		Media Access Library. These files are not designed to be modified by the user.
	Class	All files	Files related to the standard Mass Storage operation. These files are not designed to be modified by the user.
USBlib	All files.		Files related to the standard USB operations. These files are not designed to be modified by the user.

3 USING THE MASS STORAGE LIBRARIES

3.1 MAXIMUM NUMBER OF LOGICAL UNITS

A constant representing the number of medium slots implemented in the application project must be defined by the user application.

```
        #define MAX_LUN
```

The valid range of this variable is between 0 and 15. The MAX_LUN value is one less than the number of slots; for example, if the MAX_LUN value is 3, there are 4 slots.

3.2 MEDIA SELECTION

A Media Access Layer (MAL) add-in for each type of storage media is provided by STMicroelectronics. Several functions are provided with this interface and are called using a single entry function. To assign logical units to each type of media, an array pointing to the entry function must be provided by the application.

```
        const void (*Mal_Func[MAX_LUN+1])();
```

The medium is identified by the index of this array. For example, the function array is assigned as:

```
        extern void MAL_SMC_Funcs(void);      include SMC support in MAL
        extern void MAL_MMC_Funcs(void);      include MMC support in MAL

;  Assign LUNs (Logical Unit Numbers) in ascending order starting with LUN0
        const void (*Mal_Func[MAX_LUN+1])() = {
                MAL_SMC_Funcs,
                MAL_MMC_Funcs
                };
```

In the above example, the MAL_SMC_Funcs() function is provided with the MAL for SMC and the MAL_MMC_Funcs() function is provided with the MAL for MMC. Thus, the SMC device will be identified as LUN0 and the MMC device will be identified as LUN1.

3.3 INTERFACE DESCRIPTION

3.3.1 Hardware Interfaces

3.3.1.1 USB Interface

The pins used for the USB connection are defined in the ST7265x product datasheet.

3.3.1.2 Compact Flash Card 8-bit Memory Mode

Some hardware interface pins are fixed by the MAL firmware as shown in the table below.

CompactFlash-Bus	D0	D1	D2	D3	D4	D5	D6	D7	A0	A1	A2	RE	WE
ST7265x Pin	PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7	PA0	PA1	PA2	PA3	PA5

Other pins are user definable in the Media Configuration Module (Mconfig.c)

In this configuration, pins PA4, PA6 and PA7 can only be used as inputs (no interrupt capability).

3.3.1.3 SmartMediaCard Interface

Some hardware interface pins are fixed by the MAL firmware as shown in the table below. Other pins are user definable in the Media Configuration Module (Mconfig.c)

SmartMediaBus	D0	D1	D2	D3	D4	D5	D6	D7	ALE	WE	RE	RBUSY	CLE
ST7265x Pin	PB0	PB1	PB2	PB3	PB4	PB5	PB6	PB7	PA2	PA1	PA3	PA4	PA0

In this configuration, pins PA5, PA6 and PA7 can only be used as inputs (no interrupt capability).

3.3.1.4 MultiMediaCard Interface

This interface is used for MultimediaCard (MMC), Secure Digital (SD) Card in MMC mode and Hitachi cards.

Some hardware interface pins are fixed by the MAL firmware as shown in the table below. Other pins are user definable in the Media Configuration Module (Mconfig.c).

MMC Bus	DATA	COMMAND	CLOCK
ST7265x Pin	PE6	PE5	PE7

3.3.1.5 Sony Stick

Some hardware interface pins are fixed by the MAL firmware as shown in the table below. Other pins are user definable in the Media Configuration Module (Mconfig.c).

Sony Stick Bus	BS	DATA	CLOCK
ST7265x Pin	PC1	PC2	PC3

3.3.2 Software Interfaces

The software interfaces described in this document are not media specific.

3.3.2.1 Status Information

Medium Capacity

The capacity of each medium can be read from the Medium_Capacity[ ] variable

```
DWORD Medium_Capacity[MAX_LUN+1];
```

The medium is identified by the index of this array.

Current Operation

The MAL_State variable gives the internal state of the current operation. Read, write, format and verify operations are indicated with this variable. The following values are available for the MAL_State function:

```
#define MAL_IDLE          0     // no operation
#define MAL_READING       1     // upload data to USB
#define MAL_WRITING       2     // download data from USB
#define MAL_VERIFYING     3     // verifying
#define MAL_FORMATTING    4     // formatting
#define MAL_MEM_READING   5     // reading data to RAM
#define MAL_MEM_WRITING   6     // writing data from RAM
```

Active Medium

The MAL_Mediano variable identifies the medium that is currently accessed.

```
unsigned charMAL_Mediano;
```

3.3.2.2 Media Configuration

This interface is related with the physical and electrical configuration of the media device itself. Through this interface, the user must provide certain information to the MAL or carry out certain functions required by the MAL. Indeed, some functions depend on the user hardware or software implementation, and can therefore not be directly carried out by the MAL.

None of the functions defined in this section can be omitted. Even if no action is taken, the function must be defined.

Existence of the Medium

There are two operations in this category, medium existence testing and the medium removal interrupt.

Medium existence testing should be done by the user application. The application calls the MAL_Medium_Plug() interrupt to inform the MAL library that a medium is present:

```
void MAL_Medium_Plug(unsigned char slot)
```

The slot parameter indicates which medium is present.

The user application should use an interrupt routine to monitor medium removal. When the medium is removed, an interrupt is triggered and an application-defined interrupt routine is called. In this interrupt routine, the user application calls the MAL_Medium_Removing() function to notify the MAL that the medium is removed. The parameter indicates the slot where the medium removal has occurred. The MAL guarantees that the MAL_Medium_Removing() function returns to the medium removal interrupt routine as soon as possible. The template of this function is:

```
void MAL_Medium_Removing(char Index);
```

The index parameter indicates the medium that has been removed.

Selection of the Medium

Medium select and deselect operations are defined as follows:

```
#define  SELECT()
#define  DESELECT()
```

The index parameter indicates the medium to be selected or deselected.

The evaluation values of these two operations are not important. The calling function is used to detect the existence of the medium.

The SELECT() function enables the chip select bit of the selected device, and the DESELECT() function disables the chip select bit of the selected device. The application can (optional) indicate the media being accessed by switching ON an LED in the SELECT() function and switching OFF the LED in the DESELECT() function.

Once the SELECT() function is executed, the media is ready to be accessed.

Detection of the Write Protection

The testing of the write protection is defined as:

```
#define  isWRITE_PROTECT()
```

The index parameter indicates the medium to be tested.

The value of this operation is non-zero if the medium is write-protected, otherwise a zero is emitted.

Medium Initialization

The media ready function is initialized by the user application. For example, the user application must verify that the medium access is authorized. Either the MAL or the MSL assumes that the medium is powered on and ready to be operated after the SELECT() function is executed.