

EB037 MMC Programming Strategy

The EB037 MMC E-Block comes complete with test code and sample code to help get you started with programming your own MMC memory operations.

This document will explain how to use the sample C functions to allow you to make your own MMC programs and routines.

Note – The example MMC software will either remove or corrupt any data currently stored on your MMC card. Please make sure to backup all of your files before inserting the MMC card into the E-Block system.

Contents of the MMC programmer support package

Here is a list of the files that you receive to support your MMC E-Block.

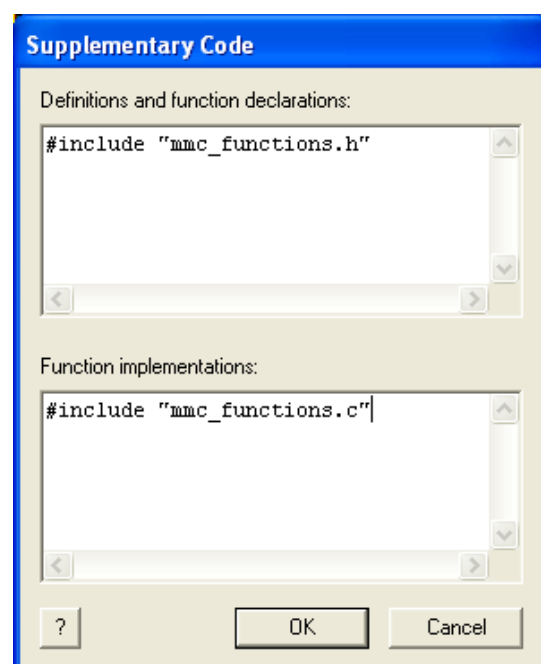
- mmc_functions.c – C code file with simple to use ready made functions.
- mmc_functions.h – Header file with the definitions of the C functions.
- ADC Sample and Store.fcf – Flowcode file demonstrating MMC functions.

Including the C functions in your program

To include the C functions into your programs you will need to copy the mmc_functions.c and the mmc_functions.h files into the directory containing your program and then do the following.

Flowcode Users:

Add the following lines into the supplementary code window available from the Edit menu.



BoostC Users:

Add the following lines to the beginning of your program.

```
#include "mmc_functions.h"  
#include "mmc_functions.c"
```

Other PIC C Users:

The Functions and header files will need to be ported into your version of C. This should be a relatively easy job and should be fairly self evident if you are familiar with your particular version of C.

MMC Commands

Here is a brief example of some of the more common MMC commands.

Command	Command ID	Addr1H	Addr1L	Addr2H	Addr2L	CRC
MMC Start-up	0x40	0x00	0x00	0x00	0x00	0x95
MMC Ready	0x41	0x00	0x00	0x00	0x00	0xFF
512 Byte Write	0x58	Addr1H	Addr1L	Addr2H	Addr2L	0xFF
512 Byte Read	0x51	Addr1H	Addr1L	Addr2H	Addr2L	0xFF

There are hundreds of different MMC messages to choose from so for a complete set of data please refer to documents detailing the MMC standard.

Description of C functions

Here is the list of functions with details of their operation, listings of their inputs and expected responses.

SPI Bus Initialise Function

The MMC card is accessed via a protocol named SPI. This protocol uses 4 I/O pins to represent data in, data out, data clock and chip select. The example MMC functions use the onboard hardware dedicated to the SPI bus. The dedicated hardware pins are referred to, as SDO, SDI, CLK and the chip select pin can be any general I/O pin. The SPI bus can be emulated in software but due to the complex nature of the MMC E-Block, the example file uses the hardware SPI pins.

```
//*****  
//Define SPI Bus Pins 16F877A  
//*****  
#define SPI_Port    portc  
#define SPI_Trisc   trisc  
#define SPI_CS      2  
#define SPI_SCK     3  
#define SPI_SDI     4  
#define SPI_SDO     5
```

To initialise the SPI bus, use the `init_SPI` function. This function sets up the hardware to SPI mode and configures the direction of data on the I/O pins. The `init_SPI` function is designed for use with 16F877A and other pin compatible PICmicros. If a PICmicro that uses different pins for the SPI hardware is being used then the pin definitions in `mmc_functions.h` will need to be edited accordingly.

Example `init_SPI` function call:
[init_SPI\(\);](#)

MMC Initialise Function

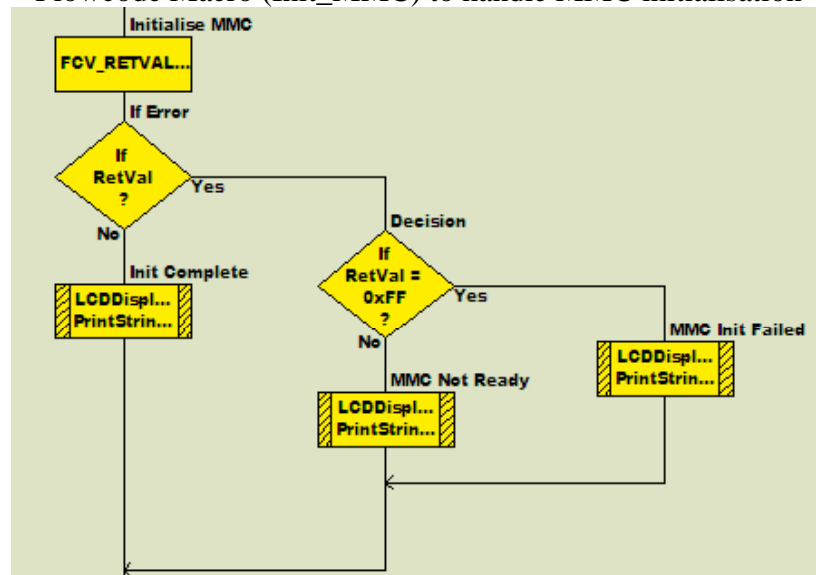
To allow access to the MMC card it must be configured in a particular way. This includes things like taking time to allow for the card to come online, clocking though data to synchronise the card and allowing the card to use the SPI bus and the 512-byte protocol.

Example `init_MMC` function call:
[FCV_RETVAL = init_MMC\(\);](#)

The return byte is stored into a Flowcode variable currently named `RetVal`. This return byte allows for the status of the MMC to be monitored.

Returned Value	Definition
0x00	MMC Initialised Correctly
0xFE	MMC Not Ready For Operation
0xFF	MMC Not Reset Correctly

Flowcode Macro (Init_MMC) to handle MMC initialisation



MMC Buffer Functions

The minimum amount of data that can be read from or written to the MMC card is 512-bytes. This means that to write or read one byte of data to or from the card involves handling 511-bytes of unwanted data. The sample read and write functions use a section of memory that is 256-bytes thus making it easier to manipulate smaller chunks of data. The 256-bytes are then duplicated twice onto the MMC card, allowing the PICmicro to determine whether the data is accurate or corrupted.

Prototypes for Buffer memory

```
write_Buffer(char address, char data);  
char read_Buffer(char address);
```

Example write_Buffer function call, stores the value contained in Flowcode variable 'data' into the memory location contained in Flowcode variable 'address'.

```
write_Buffer(FCV_ADDRESS, FCV_DATA);
```

Example read_Buffer function call, collects the value of the memory location contained in Flowcode variable 'address' and returns the value into Flowcode variable 'data'.

```
FCV_DATA = read_Buffer(FCV_DATA);
```

Warning – Upon calling a read_MMC function (see below) the MMC data buffer is wiped clean. To avoid losing data you should first make sure that the buffer has been written to the MMC card.

MMC Write / Read Functions

The example MMC_write function uses the 256-byte buffer and copies the data from the buffer into the MMC card at the chosen address. The write function then repeats the 256-bytes to achieve the complete 512-byte write needed by the MMC card. There are two reasons why the data is copied or striped this way:

- 1) The amount of physical memory available of a PICmicro is limited at best therefore having a buffer that was 512 bytes in size would severely limit the available program size and complexity.
- 2) Writing the data twice allows for data redundancy, which is utilized in the example read_MMC function by checking to see if the data has been corrupted.

The read and write functions use a combination of four address bytes to address the locations available on the MMC card. The limits to these address locations depend on the size and format of your particular MMC card.

Warning – Any computer data present on the MMC card is likely to be corrupted when using the example functions. Make sure you have a backup of your files before performing any of the example data operations.

Prototype for write_MMC function

`char write_MMC(char add1h, char add1l, char add2h, char add2l);`

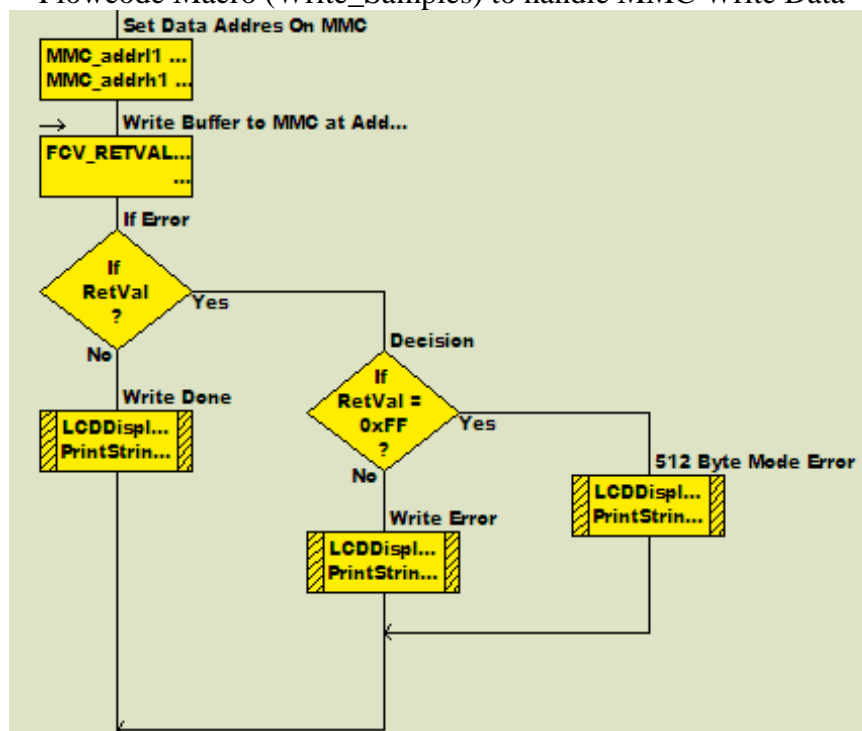
Example write_MMC function call using Flowcode variables ah1, al1, ah2 and al2 to signify the address and variable RetVal to contain the return value.

`FCV_RETVAL = write_MMC(FCV_AH1, FCV_AL1, FCV_AH2, FCV_AL2);`

The return byte RetVal allows for the status of the MMC card to be monitored.

Returned Value	Definition
0x00	MMC Write Complete
0xFE	MMC Write Error
0xFF	MMC 512-byte Mode Error

Flowcode Macro (Write_Samples) to handle MMC Write Data



Prototype for read_MMC function

`char read_MMC(char add1h, char add1l, char add2h, char add2l);`

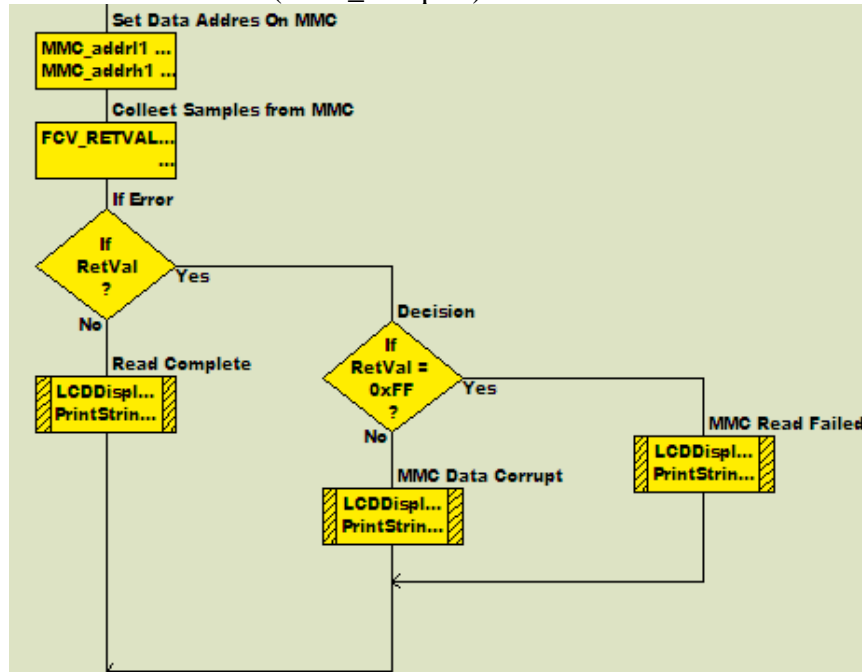
Example read_MMC function call using Flowcode variables ah1, al1, ah2 and al2 to signify the address and variable RetVal to contain the return value.

`FCV_RETVAL = read_MMC(FCV_AH1, FCV_AL1, FCV_AH2, FCV_AL2);`

The return byte RetVal allows for the status of the MMC card to be monitored.

Returned Value	Definition
0x00	MMC Read Complete
0xFE	MMC Data Pair Mismatch (Corrupt)
0xFF	MMC Read Mode Error

Flowcode Macro (Read_Samples) to handle MMC Read Data



Example Flowcode Program

There is an example Flowcode program designed to test your MMC E-Block and provide an example of how to utilize the MMC functions provided.

ADC Sample and Store.fcf

Provides an example of how to store up to 256 values into the buffer, write the buffer memory to the MMC, read the MMC memory back into the buffer and finally use the retrieved values.

Example Flowcode Macros

There are also three Flowcode macros to allow easy implementation with the MMC card. To add the macros to your programs simply use the Macro menu and click import.

INIT_MMC.fcm

Configures and initialises the SPI bus and the MMC card.

Read_Samples.fcm

Reads from the MMC card and stores the result into the 256-byte buffer.

Write_Samples.fcm

Writes to the MMC card using the data stored in the 256-byte buffer.

Implementing FAT16 (Windows: File Allocation Table)

To allow data that is written to the MMC card by the PICmicro to be compatible with Windows and to allow data written by Windows to be compatible with the PICmicro the FAT disk standard has to be followed. The FAT standard is not implemented in the supplied example functions as so the PICmicro is using the MMC card rather like a large flash memory depository. Implementing the FAT protocol involves a lot of large calculations that would eat up a lot of program memory and processor time. See the datasheets regarding FAT (File Allocation Table) for further information.

Troubleshooting

If you are having problems reading data from the card then the problem is most likely caused by the type of card you are using. At the moment only MMC type memory cards are supported. SD memory cards are slightly different in shape, connectivity and also have an encryption layer on the data. Unfortunately this means that SD cards are not currently supported by the MMC card reader board.