使用SPI Flash 作為 USB 儲存裝置

NuMicro® 32位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼實現通過大容量儲存接口將檔案寫入 SPI Flash
BSP 版本	M032_Series_BSP_CMSIS_V3.05.000
開發平台	NuMaker-M032KI V1.0

1. 概述

此範例程式使用微控制器 M032 作為 SPI Flash 燒錄器，通過 USB 接口更新 SPI Flash。M032 透過 USB 通訊埠連接電腦，扮演大容量儲存裝置類別，並連接外部的 SPI Flash Memory 作為儲存裝置的媒介。通過使用大容量儲存接口，不需要為 PC 提供任何工具或驅動程序即可實現將檔案儲存於 SPI Flash。將 USB 數據線插入設備， PC 將識別它為隨身碟，使用者只需將檔案 “Update.bin” 複製到隨身碟即可達到 “將檔案寫入 SPI Flash” 。檔案位置的訊息儲存於 SRAM，所以此磁碟會在程式重啟後回復為預設的狀態。

1.1 原理

本次實驗採用 USB Mass Storage Device Class (MSC) Bulk-Only Transfer 協議，根據該協議規範，Host 與 Device 交換資料的過程可分為三個狀態階段，分別為 Command Transport (CBW)，Data-Out / Data-In，以及 Status Transport (CSW)，其狀態切換流程如圖 1-1 所示。

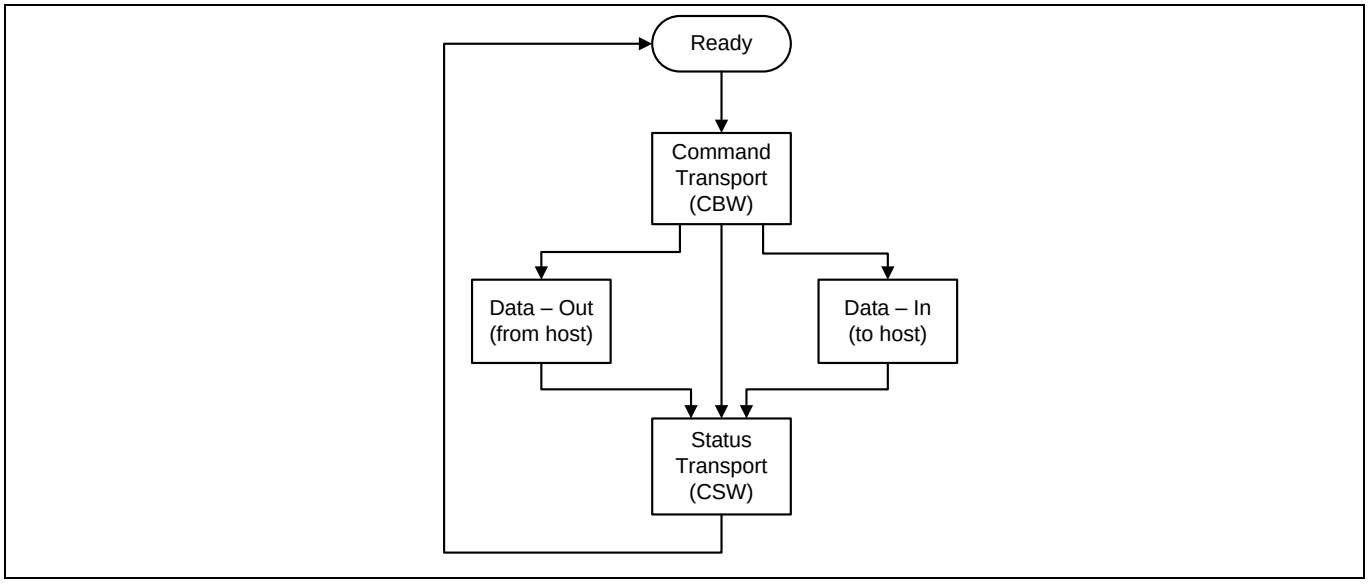

圖 1-1 大容量存儲類別 Command/Data/Status 協議

MSC 同時也定義了兩個專屬的類別請求，分別為 Bulk-Only Mass Storage Reset，以及 Get Max LUN 命令，Bulk-Only Mass Storage Reset 命令用以重置 Device 回到 Ready 狀態，通知裝置準備接收 CBW 命令；Get Max LUN 則用以取得 Device 所支援的邏輯單元 (儲存槽) 數量，其命令的詳細內容與格式可參閱 USB Mass Storage Class Bulk-Only Transport Specification 文件 (https://www.usb.org/sites/default/files/usbmassbulk_10.pdf)。

1.1.1 Command Transport (CBW)

Command Transport 內含有一個 Transaction，資料方向為 OUT，由 Host 透過 Bulk-Out Endpoint 對 Device 送出指定命令的資料封包，並以 Command Block Wrapper (CBW) 之格式封裝，封包格式如表 1-1所示。在 CBW 封包當中，欄位 CBWCB 存放 Host 對於 Device 的儲存媒體進行存取或控制的指令，該指令類型會決定下一個狀態為 Data-In 或是 Data-Out，以及雙方該傳遞的內容，詳細的指令描述可參閱 UFI Command Specification 文件規範 (https://www.usb.org/sites/default/files/usbmass-ufi10.pdf)。

Byte \ bit	7	6	5	4	3	2	1	0
0-3	*dCBWSignature*							
4-7	*dCBWTag*							
8-11	*dCBWDataTransferLength*							
12	*bmCBWFlags*							
13	*Reserved(0)*				*bCBWLUN*			
14	*Reserved(0)*				*bCBWCBLengh*			
15-30	*CBWCB*							

表 1-1 Command Block Wrapper (CBW) 封包格式

1.1.2 Data-In / Data-Out

Data-In / Data-Out 狀態下的行為受 CBW 內的命令所影響，舉例來說，若 CBWCB 內的 UFI 指令為 WRITE，則代表由 Host 希望向 Device 的儲存媒介寫入檔案資料，同時雙方將狀態切換為 Data-Out；反之若 UFI 指令為 READ，則表示 Host 向 Device 讀取儲存裝置內的檔案，由 Device 向 Host 送出資料封包，此時狀態切換至 Data-In。Data-In / Data-Out 階段允許包含多個 Transaction，由 CBWCB 內的 dCBWDataTransferLength 欄位所影響。順帶一提，本次實驗的儲存媒介是採用外接的 SPI Flash Memory，微控制器接收到讀取或寫入的指令後，必須驅動周邊 SPI 對外部 Flash Memory 的指定區塊進行資料的讀寫。

1.1.3 Status Transport (CSW)

Status Transport 的用途是 Device 向 Host 回覆本次 CBW 命令執行的狀態，含有一個 Transaction，資料方向為 IN，此階段的資料封包以 Command Status Wrapper (CSW) 之格式封裝，封包格式如表 1-2所示。Host 可透過 CSW 內的 bCSWStatus 欄位判斷 Device 在執行 CBW 命令的過程是否出現錯誤，bCSWStatus 數值意義如表 1-3所示。

Byte \ bit	7	6	5	4	3	2	1	0
0-3	*dCSWSignature*							
4-7	*dCSWTag*							
8-11	*dCSWDataResidue*							
12	*bCSWStatus*							

表 1-2 Command Status Wrapper (CSW) 封包格式

Value	Descrition
00h	Command Passed (“good status”)
01h	Command Failed
02h	Phase Error
03 and 04h	Reserved (Obsolete)
05h to FF	Reserved

表 1-3 bCSWStatus 欄位內容

1.1.4 USB 端點配置

本範例需要配置四個端點給予 USB 介面使用，如下所示：

- Endpoint 0: Control-In
- Endpoint 1: Control-Out
- Endpoint 2: Bulk-In
- Endpoint 3: Bulk-Out

1.1.5 Serial Flash Memory

本範例代碼使用 Winbond W25Q32JVSIG SPI Flash 作為 Mass Storage Device 的儲存媒介，儲存容量為 4 MB，最小的寫入單位為一個 Page (256 B)，每 16 Page 可組成一個 Sector (4 KB)，每 256 Page 則組成一個 Block (64 KB)，其中 Sector 為最小的抹除單位。SPI Flash 的控制方式非常簡便，微控制器首先透過以下的命令集合對 SPI Flash 下達指令，包含 Read、Program 或是 Erase 等行為，接著雙方會依據指令的類別，決定後續傳遞的資料的內容與長度。

Data Input Output	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Number of Clock(1-1-1)	8	8	8	8	8	8	8
Write Enable	06h						
Volatile SR Write Enable	50h						
Write Disable	04h						
Release Power-down / ID	ABh	Dummy	Dummy	Dummy	(ID7-ID0)(2)		
Manufacturer/Device ID	90h	Dummy	Dummy	00h	(MF7-MF0)	(ID7-ID0)	
JEDEC ID	9Fh	(MF7-MF0)	(ID15-ID8)	(ID7-ID0)			
Read Unique ID	4Bh	Dummy	Dummy	Dummy	Dummy	(UID63-0)	
Read Data	03h	A23-A16	A15-A8	A7-A0	(D7-D0)		
Fast Read	0Bh	A23-A16	A15-A8	A7-A0	Dummy	(D7-D0)	
Page Program	02h	A23-A16	A15-A8	A7-A0	D7-D0	D7-D0(3)	
Sector Erase (4KB)	20h	A23-A16	A15-A8	A7-A0			
Block Erase (32KB)	52h	A23-A16	A15-A8	A7-A0			
Block Erase (64KB)	D8h	A23-A16	A15-A8	A7-A0			
Chip Erase	C7h/60h						
Read Status Register-1	05h	(S7-S0)(2)					
Write Status Register-1(4)	01h	(S7-S0)(4)					
Read Status Register-2	35h	(S15-S8)(2)					
Write Status Register-2	31h	(S15-S8)					
Read Status Register-3	15h	(S23-S16)(2)					
Write Status Register-3	11h	(S23-S16)					
Read SFDP Register	5Ah	A23-A16	A15-A8	A7–A0	dummy	(D7-0)	
Erase Security Register(5)	44h	A23-A16	A15-A8	A7-A0			
Program Security Register(5)	42h	A23-A16	A15-A8	A7-A0	D7-D0	D7-D0(3)	
Read Security Register(5)	48h	A23-A16	A15-A8	A7-A0	Dummy	(D7-D0)	
Global Block Lock	7Eh						
Global Block Unlock	98h						
Read Block Lock	3Dh	A23-A16	A15-A8	A7-A0	(L7-L0)		
Individual Block Lock	36h	A23-A16	A15-A8	A7-A0			
Individual Block Unlock	39h	A23-A16	A15-A8	A7-A0			
Erase / Program Suspend	75h						
Erase / Program Resume	7Ah						
Power-down	B9h						
Enable Reset	66h						
Reset Device	99h						

表 1-4 Winbond SPI Flash 命令表

表 1-4為 Winbond W25Q32JVSIG 指令表，以 Page Program (02h) 與 Read Status Register (05h) 指令為例，可參閱圖 1-2與圖 1-3之時序，Master 首先送出 1 Byte 的指令 02h，接續 24-Bit 的資料寫入位址與 256 Byte 的資料後，便完成 Page Program 的指令傳輸。之後，微控制器可再使用 Read Status Register (05h) 指令來詢問 SPI Flash 的狀態，來檢查資料是否已經寫入完畢。使用者可參閱 Winbond W25Q32JVSIG Datasheet 文件，內有各個指令詳細的使用方法與說明。

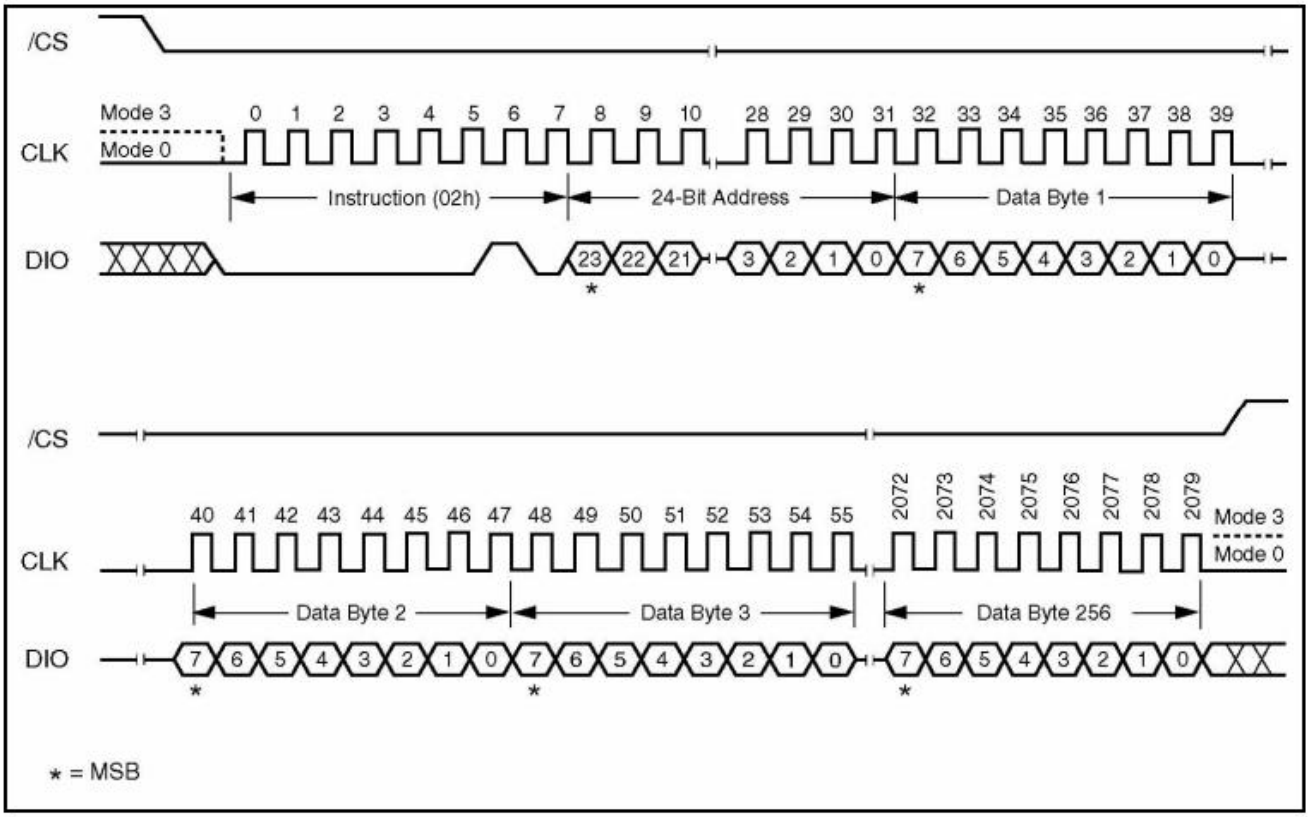

圖 1-2 Page Program 命令時序圖

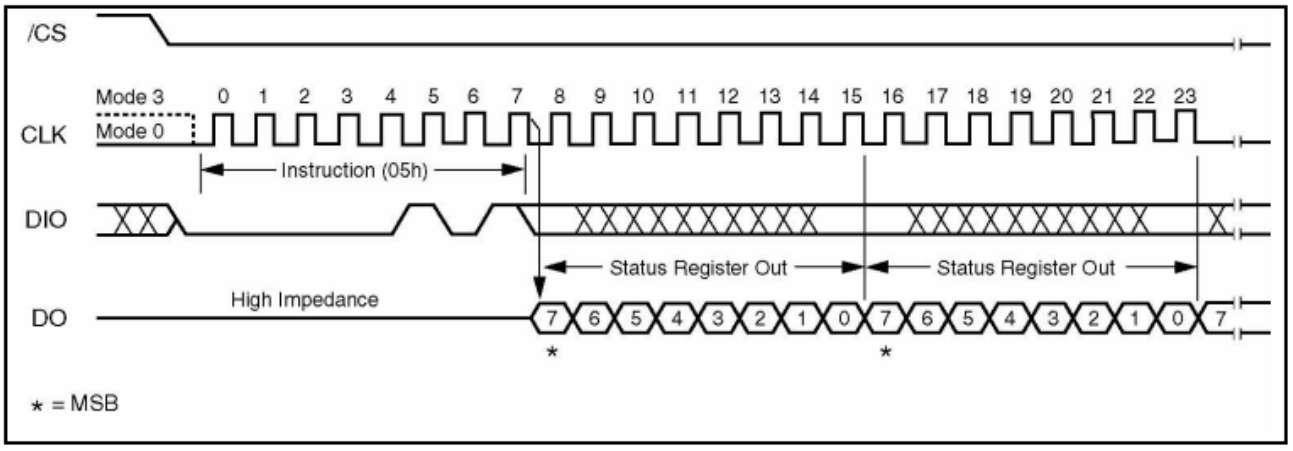

圖 1-3 Read Status Register 命令時序圖

1.2 執行結果

1.2.1 虛擬磁碟

當 M032 插入 PC USB 通訊埠時，PC會辨識出儲存裝置，DataFlash 顯示容量為3.98 MB的隨

身碟。使用者需要將檔案命名為 Update.bin並將該檔案拖曳或複製並貼上到此隨身碟。

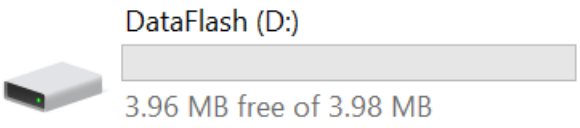

圖 1-4 M032 虛擬磁碟

1.2.2 讀取資料

從M032虛擬磁碟以無緩衝的方式讀取Update.bin, 並將其與先前寫入的資料比對結果一致。

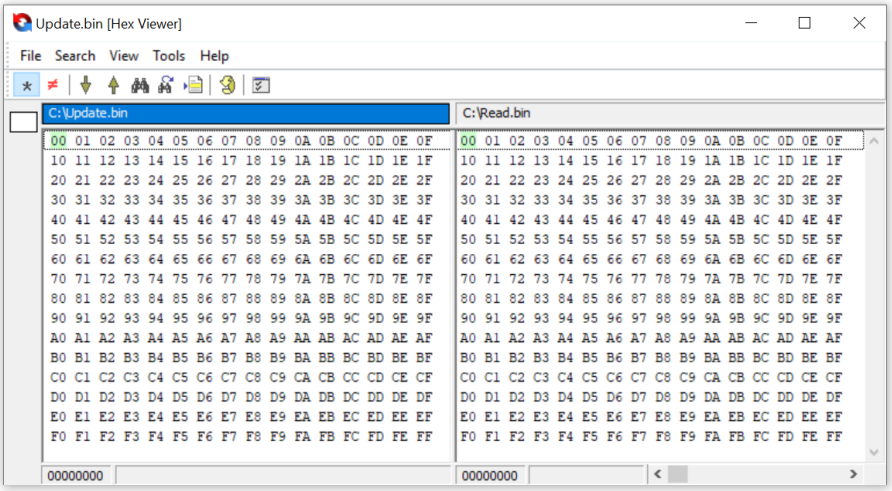

圖 1-5 比較結果

1.2.3 無緩衝的 I/O 複製檔案

1.2.3.1 Windows

Windows command - Xcopy

Xcopy Command	
參數	**敘述**
/j	使用無緩衝的 I/O 複製，建議使用於非常大的檔案。此功能首次支援於 Window 7。

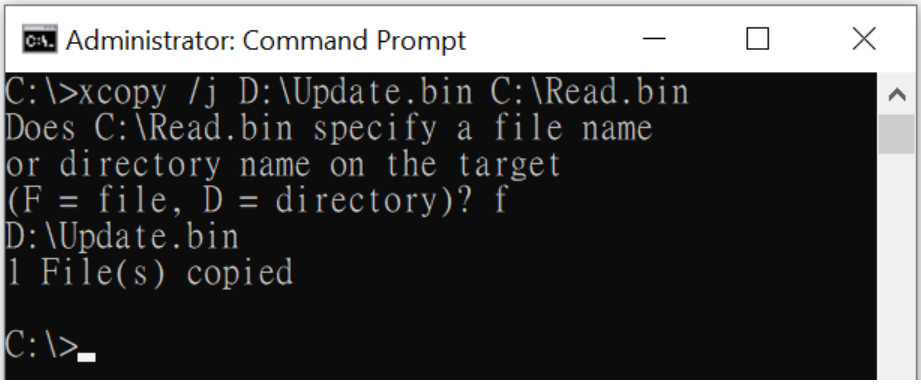

圖 1-6 使用 xcopy 以無緩衝的 I/O 複製檔案

1.2.3.2 Linux

Linux command - dd

dd Command	
參數	**敘述**
iflg=direct	使用direct I/O的方式

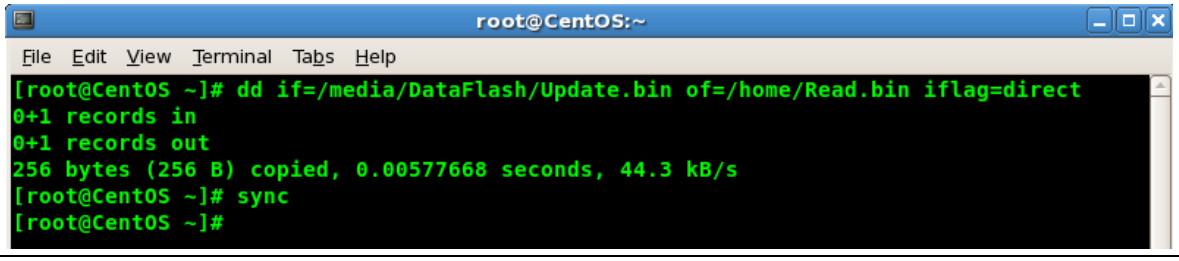

圖 1-7 使用 dd 命令以無緩衝的 I/O 複製檔案

2. 代碼介紹

2.1 虛擬磁碟配置

配置虛擬磁碟的內容，代碼位於 DataFlashProg.c，以下是虛擬磁碟的啟動磁區內容

```
uint8_t u8BootSectorData[512] =
{
    /* Instructions to jump to boot code */
    0xEB, 0x3C, 0x90,
    /* Name string (MSDOS5.0) */
    0x4D, 0x53, 0x44, 0x4F, 0x53, 0x35, 0x2E, 0x30,
    /* Bytes/sector (0x0200 = 512) */
    0x00, 0x02,
    /* Sectors/cluster */
    0x08,
    /* Size of reserved area */
    (RSVD_SEC_CNT & 0xFF), (RSVD_SEC_CNT >> 8),
    /* Number of FATs */
    NUM_FAT,
    /* Byte 17 & 18 (Max. number of root directory entries) */
    (ROOT_ENT_CNT & 0xFF) , (ROOT_ENT_CNT >> 8),
    /* Byte 19 & 20 (Total number of sectors) */
    (SECTOR_CNT & 0xFF), (SECTOR_CNT >> 8),
    /* Media type (removable) */
    0xF8,
    /* Byte 22 & 23 (FAT size) */
    (FAT_SZ & 0xFF), (FAT_SZ >> 8),
    /* Sectors/track */
    0x01, 0x00,
    /* Number of heads */
    0x01, 0x00,
    /* Number of sector before partition */
    0x00, 0x00, 0x00, 0x00,
    /* Total number of sectors */
    0x00, 0x00, 0x00, 0x00,
    /* Drive number */
    0x80,
    /* Unused */
    0x00,
    /* Extended boot signature */
    0x29,
    /* Volume serial number */
    0x2F, 0x44, 0x83, 0x7A,
    /* Volume label - "NO NAME " */
    0x4E, 0x4F, 0x20, 0x4E, 0x41, 0x4D, 0x45, 0x20, 0x20, 0x20, 0x20,
    /* File system type label ("FAT12 ") */
    0x46, 0x41, 0x54, 0x31, 0x32, 0x20, 0x20, 0x20,
    ...
    /* Signature value (0xaa55) */
    0x55, 0xAA
};
```

配置虛擬磁碟內容的定義，代碼位於 DataFlashProg.h，修改 “MB_SIZE” 就可以修改虛擬磁碟的大小。

```
#define MB_SIZE                 4
#define DISK_SIZE               (MB_SIZE*1024 * 1024)
...
#define BYTE_PER_SEC            512
#define ROOT_ENT_CNT            512
#define ROOT_ENT_SEC_CNT        ((32 * ROOT_ENT_CNT) / BYTE_PER_SEC)
#define NUM_FAT                 2
#define FAT_SEC                 RSVD_SEC_CNT
#define FAT_SEC_ADDR            (RSVD_SEC_CNT * BYTE_PER_SEC)
#define ROOT_SEC_ADDR           (FAT_SEC_ADDR + FAT_SZ * NUM_FAT * BYTE_PER_SEC)
#define DATA_SEC_ADDR           (ROOT_SEC_ADDR + ROOT_ENT_SEC_CNT * BYTE_PER_SEC)
#define SECTOR_CNT              (DISK_SIZE / BYTE_PER_SEC)
```

2.2 檔案配置表(FAT)與讀寫 Flash

DataFlashRead() 負責回覆 PC 端儲存裝置的檔案配置表(FAT)並且根據 CBW 內的 Logic Block Address 以及 dCBWDataTransferLength 欄位來讀取指定的磁區位址以及資料長度。

```
void DataFlashRead(uint32_t addr, uint32_t size, uint32_t buffer)
{
    /* This is low level read function of USB Mass Storage */
    uint32_t * pu32Buf = (uint32_t *)buffer;
    memset((uint8_t *)pu32Buf, 0, STORAGE_BUFFER_SIZE);

    if (addr == 0x00000000)
    {
#ifdef __FAT_INFO__
        if(g_ShowFat && GET_FAT_SZ != 0)
        {
            printf("\n");
            printf("Default FAT_SEC        0x%08X ", FAT_SEC);
            printf("Current FAT_SEC       0x%08X\n", GET_FAT_SEC);
            printf("Default FAT_SZ         0x%08X ",FAT_SZ);
            printf("Current FAT_SZ        0x%08X\n", GET_FAT_SZ);
            printf("Default ROOT_SEC_ADDR  0x%08X ",ROOT_SEC_ADDR);
            printf("Current ROOT_SEC_ADDR 0x%08X\n", GET_ROOT_SEC_ADDR);
            printf("Default DATA_SEC_ADDR  0x%08X ",DATA_SEC_ADDR);
            printf("Current DATA_SEC_ADDR 0x%08X\n", DATA_SEC_ADDR);
            g_ShowFat = 0;
        }
#endif
        USBD_MemCopy((uint8_t *)buffer, u8BootSectorData, sizeof(u8BootSectorData));
    }
    else
    {
        if (addr >= ROOT_SEC_ADDR && addr < DATA_SEC_ADDR)     /* Root Directory */
            USBD_MemCopy( (uint8_t *)buffer, u8DirData + (addr - ROOT_SEC_ADDR), 512);
        else if (addr >= FAT_SEC_ADDR && addr < ROOT_SEC_ADDR) /* File Allocation Table */
            USBD_MemCopy((uint8_t *)buffer, u8FAT + ((addr - FAT_SEC_ADDR)/* % (FAT_SZ *
                        BYTE_PER_SEC)*/), 512);
```

```
        else if(addr > DATA_SEC_ADDR) /* Data */
        {
            SpiRead(u32SPI_RAddress, size, (uint32_t)pu32Buf);
            if(u32SPI_RAddress % 0x40000 == 0)
                DbgPrintf("Read from SPI 0x%08X - 0x%08X\n",u32SPI_RAddress, *(uint32_t
                          *)pu32Buf);
            u32SPI_RAddress+=size;
        }
    }
}
```

DataFlashWrite()呼叫底層函數 SpiWrite()執行寫入操作，將檔案儲存到 SPI Flash 的指定區域。寫入操作完成後，還會呼叫 SpiRead() 來驗證儲存在 SPI Flash 中的數據是否正確。

```
void DataFlashWrite(uint32_t addr, uint32_t size, uint32_t buffer)
{
    int k,l;
    /* This is low level write function of USB Mass Storage */
    if(addr >= DATA_SEC_ADDR)  /* Data */
    {
        if(addr == DATA_SEC_ADDR)
        {
            if(g_UpdateEnable == 0 && g_u8Windows == 0)
            {
                DbgPrintf("OS : Windows\n");
                g_u8Windows = 1;
            }
        }
        if(g_UpdateEnable || (g_u8Windows == 0 && g_u8MACOS == 0))
        {
            if(u32SPI_WAddress == 0)
            {
                if(g_u8Windows == 0 && g_u8MACOS == 0)
                    DbgPrintf("OS : Linux\n");
            }

            addr -= DATA_SEC_ADDR;

            SpiWrite(u32SPI_WAddress, STORAGE_BUFFER_SIZE, (uint32_t)buffer);

            SpiRead(u32SPI_WAddress, STORAGE_BUFFER_SIZE, (uint32_t)Storage_Block_Verify);

            if(memcmp((void *)buffer, (void *)Storage_Block_Verify,
               STORAGE_BUFFER_SIZE) !=0)
                DbgPrintf("Verify fail 0x%08X\n",u32SPI_WAddress);

            u32SPI_WAddress += STORAGE_BUFFER_SIZE;

            if(u32SPI_WAddress % 0x40000 == 0)
            {
                DbgPrintf("\rData Transferred %d KB",u32SPI_WAddress >> 10);
                if(g_file_size != 0)
                    DbgPrintf(" / %d KB",*g_file_size >>10);
            }
            if(g_file_size != 0 && u32SPI_WAddress >= *g_file_size)
```

```
            {
                DbgPrintf("\rData Transferred %d KB",u32SPI_WAddress >> 10);
                if(g_file_size != 0)
                    DbgPrintf(" / %d KB",*g_file_size >>10);

                g_TransferDone = 1;
                u32SPI_RAddress = 0;
                g_UpdateEnable = 0;
                DbgPrintf("\nTransferred Done\n");
            }
        }
        else
        {
            uint8_t *pu8Data = (uint8_t *)buffer;

            if(pu8Data[8] == 'M' && pu8Data[9] == 'a' && pu8Data[10] == 'c')
            {
                if(g_u8MACOS == 0)
                {
#if (ENABLE_DEBUG_MSG)
                    char *puchar = (char *)(&pu8Data[8]);
#endif
                    DbgPrintf("OS : %s\n",puchar);
                    g_u8MACOS = 1;
                }
                /* Finder progresses a copy of a file - HFS type code 'brok' & HFS creator
                    code 'MACS' */
                if(pu8Data[50] == 'b' && pu8Data[51] == 'r' && pu8Data[52] == 'o' &&
                    pu8Data[53] == 'k' && pu8Data[54] == 'M' && pu8Data[55] == 'A' &&
                    pu8Data[56] == 'C' && pu8Data[57] == 'S')
                {
                    g_u8MACOS_Update = 1;
                }
            }
        }
    }
    else if (addr == 0x00000000)
    {
        USBD_MemCopy(u8BootSectorData,(uint8_t *) buffer, 512);
#ifdef __FAT_INFO__
        g_ShowFat = 1;
#endif
    }
    else if (addr >= ROOT_SEC_ADDR && addr < DATA_SEC_ADDR) /* Root Directory */
    {
        USBD_MemCopy(u8DirData + (addr - ROOT_SEC_ADDR), (uint8_t *)buffer, 512);

        if((g_UpdateEnable == 0 && g_TransferDone == 0) || ((g_u8MACOS == 1 || g_u8Windows
           == 0)&& g_u8ShowFile != 1))           /* Check File name (When Update Function
           is not Enabled or MAC OS) */
        {
            for(k =0; k<32; k++)                 /* Check Root Directory */
            {
                for(l=0; l<FILE_NAME_LENGTH; l++)
                {
                    if(u8FileName[l] != 0)       /* Need to Check File Name - Character */
```

```
                        if(u8DirData[k*16+i8FileIndex[l]] != u8FileName[l])      /* Check
                           File Name */
                            break;
                }

                if(l == FILE_NAME_LENGTH)                                       /* Match */
                {
                    g_file_size = (uint32_t *)&u8DirData[k *16 + 60];  /* Get File Size */

                    if(*g_file_size == 0)
                    {
                        g_file_size = 0;
                        continue;
                    }
                    g_UpdateEnable = 1;

                    DbgPrintf("\nFile Name:");
                    for(l=0; l<FILE_NAME_LENGTH; l++)   /* Display the Update File Name */
                    {
                        if((u8DirData[k *16 + i8FileIndex[l]] != 0) && (u8DirData[k *16 +
                           i8FileIndex[l]+1] == 0) )
                            DbgPrintf("%c",u8DirData[k *16 + i8FileIndex[l]]);
                        else
                            break;
                    }

                    if(g_u8MACOS == 1 || g_u8Windows == 0)
                    {
                        g_u8ShowFile = 1;
                    }
                    DbgPrintf("\nFile Size: %dB\n",*g_file_size);
                    if(u32SPI_WAddress >= *g_file_size)
                    {
                        u32SPI_RAddress = 0;
                        g_TransferDone = 1;
                        DbgPrintf("\nTransferred Done\n");
                    }
                    break;
                }
            }
        }
    }
    else if (addr >= FAT_SEC_ADDR && addr < ROOT_SEC_ADDR)  /* File Allocation Table */
        USBD_MemCopy(u8FAT + ((addr - FAT_SEC_ADDR) /*% (FAT_SZ * BYTE_PER_SEC)*/),
                     (uint8_t *)buffer, 512);
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M032_Series_BSP_CMSIS_V3.05.000
- IDE 版本
  - Keil uVersion 5.28

3.2 硬體需求

- 電路元件
  - NuMaker-M032KI V1.0
  - Level1-Training
  - Micro USB cable
- 示意圖

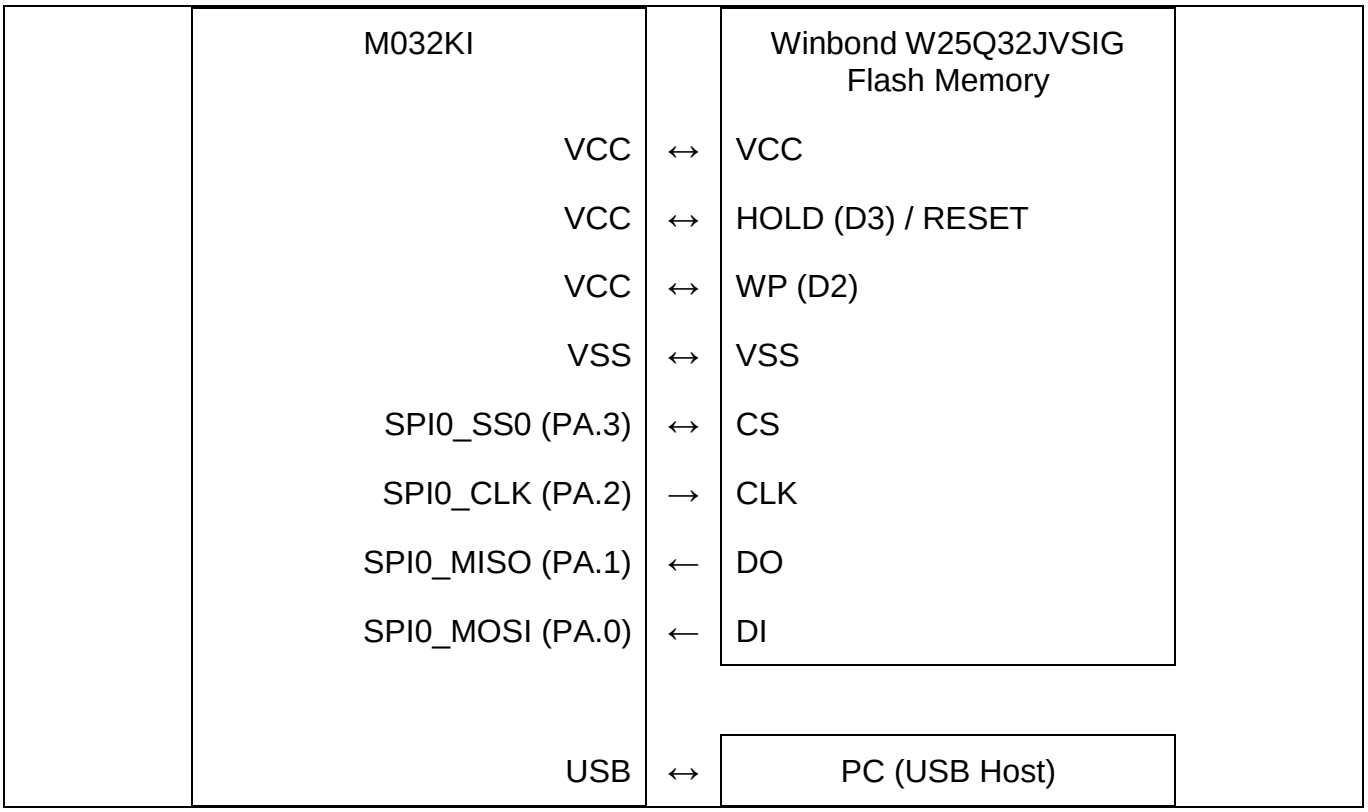

圖 3-1 系統連接圖

4. 目錄資訊

EC_M032_USBD_MSC_SPI_Flash_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *M032_USBD_MSC_SPI_Flash.uvproj*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2023.06.07	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.