实验三十：USB_MSC 实验——读/写 U 盘（大容量存储器）

一、 实验目的与意义

1、 了解 STM32 USB HOST 结构
2、 了解 STM32 USB HOST 特征
3、 掌握 USB HOST MSC 的使用方法
4、 掌握 STM32 HAL 库中 USB HOST 属性的配置方法
5、 掌握 KEILMDK 集成开发环境使用方法

二、 实验设备及平台

1、 iCore4T 双核心板
2、 JLINK（或相同功能）仿真器
3、 Micro USB 线缆
4、 Keil MDK 开发平台
5、 STM32CubeMX 开发平台
6、 装有 WIN XP（及更高版本）系统的计算机

三、 实验原理

1. USB 简介

USB，是英文 UniversalSerialBUS（通用串行总线）的缩写，而其中文简称为“通串线，是一个外部总线标准，用于规范电脑与外部设备的连接和通讯。是应用在 PC 领域的接口技术。USB 接口支持设备的即插即用和热插拔功能。USB 是在 1994 年底由英特尔、康柏、IBM、Microsoft 等多家公司联合提出的。

USB 发展到现在已经有 USB1.0/1.1/2.0/3.0 等多个版本。目前用的最多的就是 USB1.1 和 USB2.0，USB3.0 目前已经开始普及。STM32H750 自带的 USB 符合 USB2.0 规范。

标准 USB 共四根线组成,除 VCC/GND 外，另外为 D+和 D-，这两根数据线采用的是差分电压的方式进行数据传输的。在 USB 主机上，D-和 D+都是接了 15K 的电阻到地的，所以在没有设备接入的时候，D+、D-均是低电平。而在 USB 设备中，如果是高速设备，则会在 D+上接一个 1.5K 的电阻到 VCC，而如果是低速设备，则会在 D-上接一个 1.5K 的电阻到 VCC。这样当设备接入主机的时候，主机就可以判断是否有设备接入，并能判断设备是高速设备还是低速设备。接下来，我们简单介绍一下 STM32 的 USB 控制器。

STM32H750 系列芯片自带有 2 个 USBOTG，其中 USB1 是高速 USB（USB1OTGHS）；USB2 是全速 USB（USB2OTGFS），高速 USB（HS）需要外扩高速 PHY 芯片实现。

2. OTG 主要特性

主要特性可分为三类：通用特性、主机模式特性和从机模式特性。

(1) 通用特性

OTG_HS 接口的通用特性如下：

- 经 USB-IF 认证，符合通用串行总线规范第 2.0 版
- OTGHS 支持以下 PHY 接口：
  - 片上全速 PHY
  - 连接外部全速 PHY 的 I2C 接口
  - 连接外部高速 PHY 的 ULPI 接口
- 模块内嵌的 PHY 还完全支持定义在标准规范 OTG 补充第 2.0 版中的 OTG 协议
  - 支持 A-B 器件识别（ID 线）
  - 支持主机协商协议(HNP)和会话请求协议(SRP)

- 允许主机关闭 VBUS 以在 OTG 应用中节省电池电量
- 支持通过内部比较器对 VBUS 电平采取 OTG 监控
- 支持主机到从机的角色动态切换

- 可通过软件配置为以下角色：
  - 具有 SRP 功能的 USBHS 从机（B 器件）
  - 具有 SRP 功能的 USBHS/LS 主机（A 器件）
  - USBOn-The-Go 全速双角色设备
- 支持 HSSOF 和 LSKeep-alive 令牌
  - SOF 脉冲可通过 PAD 输出
  - SOF 脉冲通过内部连接到定时器(TIMx)
  - 可配置的帧周期
  - 可配置的帧结束中断
- OTG_HS 内嵌 DMA，并可软件配置 AHB 的批量传输类型。
- 具有省电功能，例如在 USB 挂起期间停止系统、关闭数字模块时钟、对 PHY 和 DFIFO 电源加以管理。
- 具有采用高级 FIFO 控制的 4KB 专用 RAM：
  - 可将 RAM 空间划分为不同 FIFO，以便灵活有效地使用 RAM
  - 每个 FIFO 可存储多个数据包
  - 动态分配存储区
  - FIFO 大小可配置为非 2 的幂次方值，以便连续使用存储单元
- 一帧之内可以无需要应用程序干预，以达到最大 USB 带宽。
- 它支持电池充电规范第 1.2 版中介绍的充电端口检测（仅限 FSPHY 收发器

(2) 主机（Host）模式特性

OTG_HS 接口在主机模式下具有以下主要特性和要求：

- 通过外部电荷泵生成 VBUS 电压。
- 多达 16 个主机通道（又称之为管道）：每个通道都可以动态实现重新配置，可支持任何类型的 USB 传输。
- 内置硬件调度器可：
  - 在周期性硬件队列中存储多达 16 个中断加同步传输请求
  - 在非周期性硬件队列中存储多达 16 个控制加批量传输请求
- 管理一个共享 RxFIFO、一个周期性传输 TxFIFO 和一个非周期性传输 TxFIFO，以有效使用 USB 数据 RAM。

(3) 从机（Slave/Device）模式特性

OTG_HS 接口在从机模式下具有以下特性：

- 1 个双向控制端点 0
- 8 个 IN 端点(EP)，可配置为支持批量传输、中断传输或同步传输
- 8 个 OUT 端点，可配置为支持批量传输、中断传输或同步传输
- 管理一个共享 RxFIFO 和一个 Tx-OUTFIFO，以高效使用 USB 数据 RAM
- 管理多达 9 个专用 Tx-INFIFO（分别用于每个使能的 INEP），以降低应用程序负荷
- 支持软断开功能。

3. OTG 框图

STM32H7 中存在两个 OTG_HS 模块（OTG_HS1 和 OTG_HS2）。尽管他们都可以编程为 HS 操作，但只有 OTG_HS1 具有可访问的 ULPI 接口，因此允许使用外部 HS 收发器进行高速操作。

图 740. 高速 OTG 模块框图 (OTG_HS1)

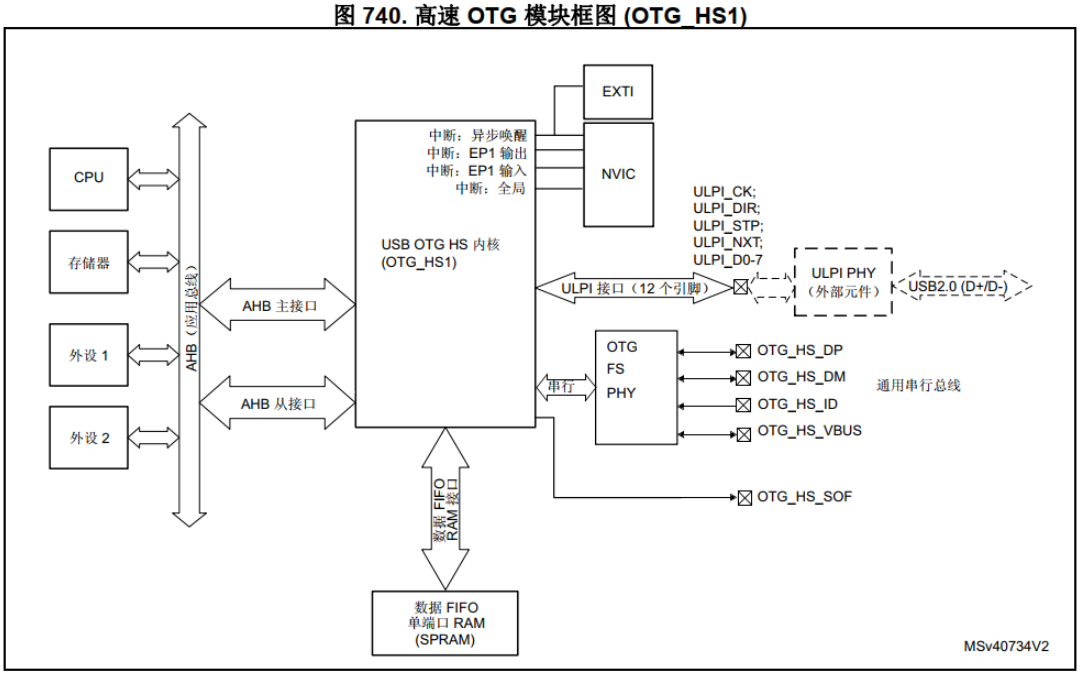
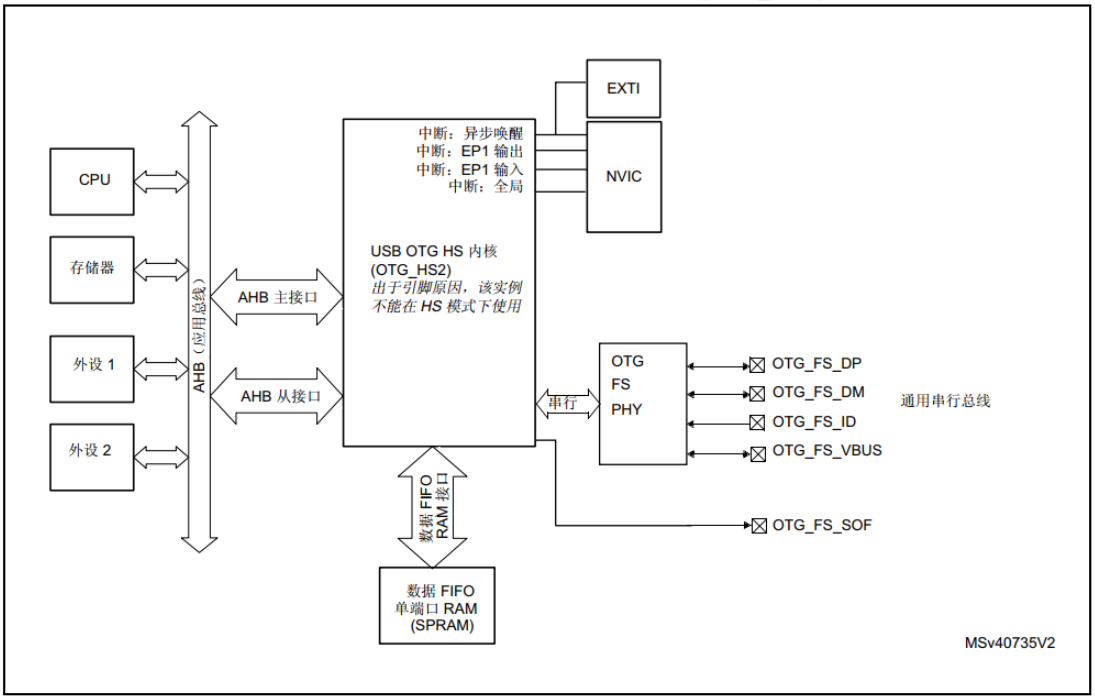


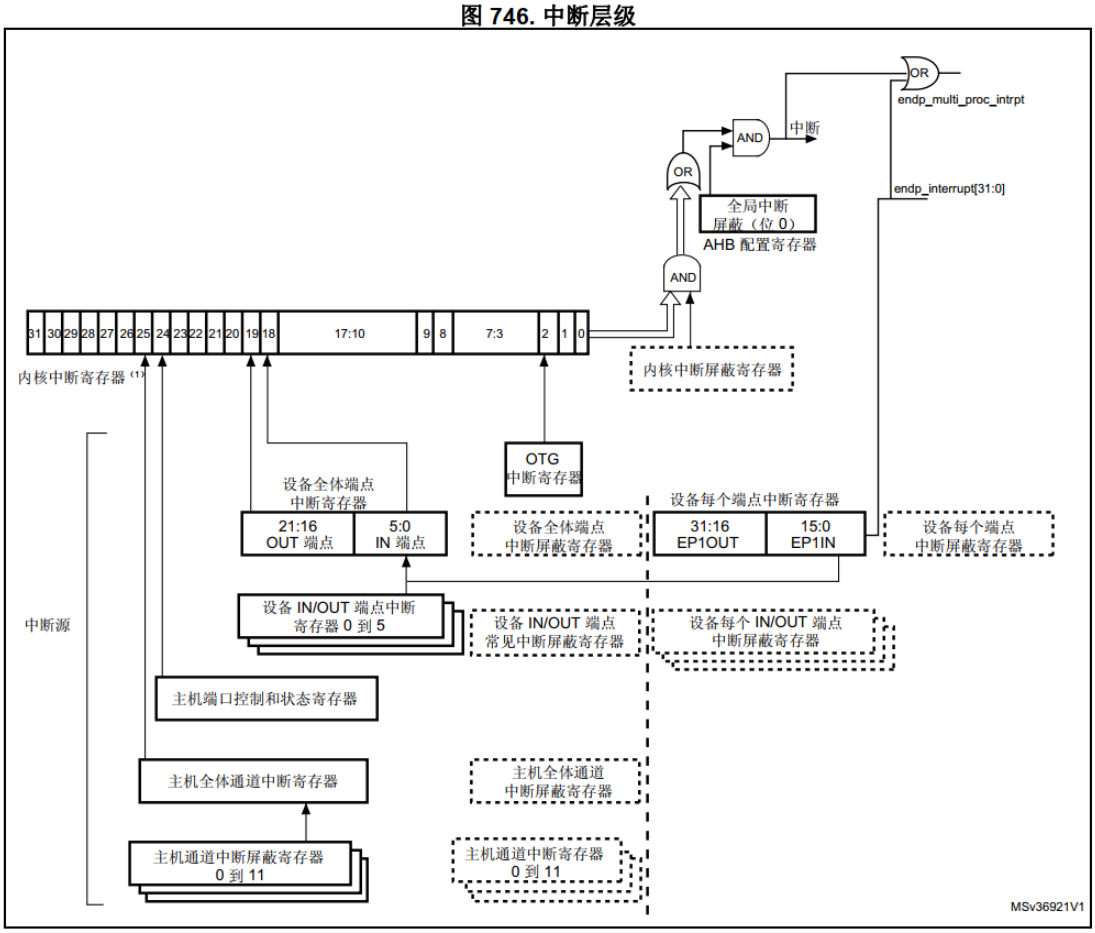
图 741. 高速 OTG 模块框图 (OTG_HS2)



4. OTG_HS 中断

当 OTG_HS 控制器在一种模式下（设备模式或主机模式）工作时，应用程序不得以另一种角色模式访问寄存器。如果发生了非法访问，将会产生模式不匹配中断并在模块中断寄存器（OTG_GINTSTS 寄存器中的 MMIS 位）中反映。当模块从一种角色模式切换到另一种角色模式时，新工作模式下的寄存器必须重新编程为上电复位后的状态。如图显示了中断层级：

图 746. 中断层级



四、 实验程序

1. 主函数

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    i2c.initialize();
    axp152.initialize();
    axp152.set_dcdc1(3500);//[ARM & FPGA BK1/2/6 &OTHER]
    axp152.set_dcdc2(1200);//[FPGA INT & PLL D]
    axp152.set_aldo1(2500);//[FPGA PLL A]
    axp152.set_dcdc4(3300);//[POWER_OUTPUT]
    axp152.set_dcdc3(3300);//[FPGA BK4][Adjustable]
    axp152.set_aldo2(3300);//[FPGA BK3][Adjustable]
    axp152.set_dldo1(3300);//[FPGA BK7][Adjustable]
    axp152.set_dldo2(3300);//[FPGA BK5][Adjustable]

    HAL_Delay(200);
    MX_GPIO_Init();
    MX_FATFS_Init();
```

```
    MX_USB_HOST_Init();
    MX_USART2_UART_Init();
    LED_ON;
    usart2.printf("\x0c");                    //清屏
    usart2.printf("\033[1;32;40m");           //设置终端字体为绿色
    usart2.printf("\r\nHello, I am iCore4T.\r\n");
    while (1)
    {
      MX_USB_HOST_Process();
    }
}
```

2. USB_HOST 初始化函数

```
void MX_USB_HOST_Init(void)
{
  /* 初始化主机库，添加支持的类并启动该库 */
  if (USBH_Init(&hUsbHostHS, USBH_UserProcess, HOST_HS) != USBH_OK)
  {
    Error_Handler();
  }
  if (USBH_RegisterClass(&hUsbHostHS, USBH_MSC_CLASS) != USBH_OK)
  {
    Error_Handler();
  }
  if (USBH_Start(&hUsbHostHS) != USBH_OK)
  {
    Error_Handler();
  }
}
```

3. USBH_UserProcess 函数

```
static void USBH_UserProcess  (USBH_HandleTypeDef *phost, uint8_t id)
{
    int i,j;
    static FRESULT res;
    unsigned char write_buffer[512];
    unsigned char read_buffer[512];
    unsigned int counter;

    switch(id)
    {
    case HOST_USER_SELECT_CONFIGURATION:
    break;
```

```
    case HOST_USER_DISCONNECTION:
    Appli_state = APPLICATION_DISCONNECT;
    break;
    case HOST_USER_CLASS_ACTIVE:
    Appli_state = APPLICATION_READY;
    //f_mount
    res = f_mount(&fatfs,"0:",1);
    if(res != RES_OK){
        USBH_UsrLog("\r\nf_mount error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
        USBH_UsrLog("\r\nf_mount successful!");
    }
    //f_open
    for(i = 0; i < 512 ; i ++)write_buffer[i] = i % 256;
    res = f_open(&file,"0:/test.txt",FA_READ | FA_WRITE | FA_OPEN_ALWAYS)
;   //打开驱动器 0 上的源文件
    if(res != RES_OK){
        USBH_UsrLog("f_open error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
        USBH_UsrLog("f_open successful!");
    }
    //f_lseek
    res = f_lseek(&file,0);
    if(res != RES_OK){
        USBH_UsrLog("f_lseek error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
```

```
        USBH_UsrLog("f_lseek successful!");
    }
    //f_write
    res = f_write(&file,write_buffer,512,&counter);
    if(res != RES_OK || counter != 512){
        USBH_UsrLog("f_write error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
        USBH_UsrLog("f_write successful!");
    }
    //f_lseek
    res = f_lseek(&file,0);
    if(res != RES_OK){
        USBH_UsrLog("f_lseek error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
        USBH_UsrLog("f_lseek successful!");
    }
    //f_read
    res = f_read(&file,read_buffer,512,&counter);
    if(res != RES_OK || counter != 512){
        USBH_UsrLog("f_read error!");
        while(1){
            LED_ON;
            HAL_Delay(500);
            LED_OFF;
            HAL_Delay(500);
        }
    }else{
        USBH_UsrLog("f_read successful!");
    }
    f_close(&file);
    USBH_UsrLog("read data:");
    for(i = 0;i < 32;i++){
```

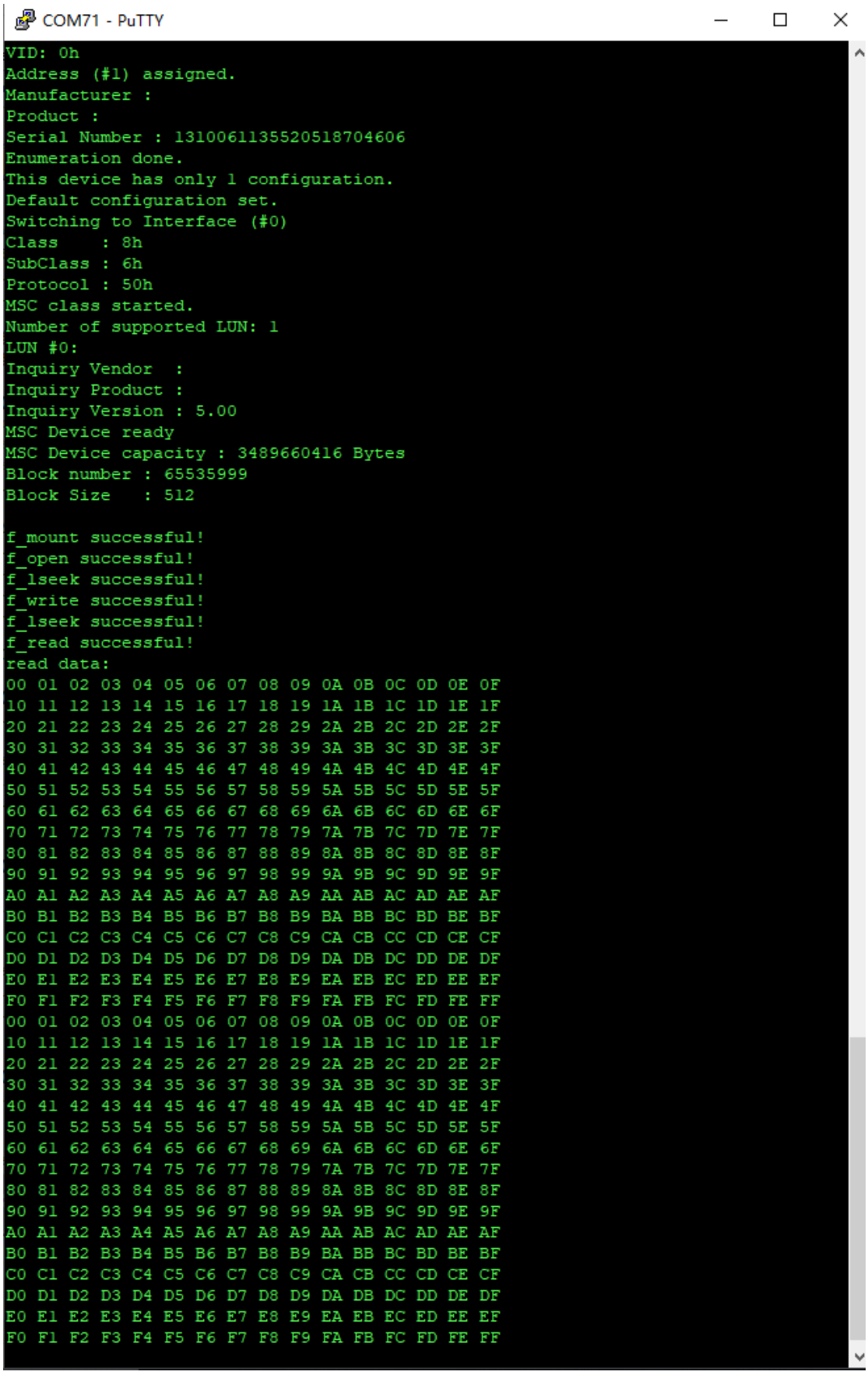
```
          for(j = 0; j < 16; j ++)
             usart2.printf("%02X ",read_buffer[i*16+j]);
          usart2.printf("\r\n");
      }
    break;
    case HOST_USER_CONNECTION:
    Appli_state = APPLICATION_START;
    break;
    default:
    break;
    }
}
```

五、 实验步骤

1、 把仿真器与 iCore4T 的 SWD 调试口相连（直接相连或者通过转接器相连）；
2、 把 iCore4T 通过 Micro USB 线与计算机相连，为 iCore4T 供电；
3、 打开 PuTTY 串口终端；
4、 打开 Keil MDK 开发环境，并打开本实验工程；
5、 烧写程序到 iCore4T 上；
6、 也可以进入 Debug 模式，单步运行或设置断点验证程序逻辑。

六、 实验现象

将 U 盘插入高速 USB A 接口，将会被自动挂载，从终端打印出来 U 盘的相关信息。对向 U 盘中写入测试文件并将文件中数据读取出来。


COM71 - PuTTY
VID: 0h
Address (#1) assigned.
Manufacturer :
Product :
Serial Number : 1310061135520518704606
Enumeration done.
This device has only 1 configuration.
Default configuration set.
Switching to Interface (#0)
Class : 8h
SubClass : 6h
Protocol : 50h
MSC class started.
Number of supported LUN: 1
LUN #0:
Inquiry Vendor :
Inquiry Product :
Inquiry Version : 5.00
MSC Device ready
MSC Device capacity : 3489660416 Bytes
Block number : 65535999
Block Size : 512
f_mount successful!
f_open successful!
f_lseek successful!
f_write successful!
f_lseek successful!
f_read successful!
read data:
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF