Implementing USB Disk and FatFs function on M032

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses the M032 series microcontroller (MCU) to implement USB Disk and FatFs function.
BSP Version	M031 Series BSP CMSIS V3.05.000
Hardware	NuMaker-M032LD V1.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

This example code uses the M032 series microcontroller (MCU) to run USB mass storage functions and FatFs file system functions, and is applicable to the M032 USB series with chip capacity not less than 64 Kbytes. The M032 series can drive file read and write operations through FatFs file system after copying files from PC to mass storage device.

1.1 Principle

The NuMicro M032LD2AE microcontroller has 64 Kbytes of APROM capacity. Here, Data Flash is used to simulate the USB mass storage space and will occupy 32 Kbytes.

As shown in Table 1-1, if the APROM Flash size of M032 series is less than 64 Kbytes, this example code cannot run because the Data Flash size must not be less than 32 Kbytes.

M032 Series	APROM Flash Size				
	32 Kbytes	64 Kbytes	128 Kbytes	256 Kbytes	512 Kbytes
Whether 32 Kbytes space can be allocated to Data Flash for USB disk?	NO	YES	YES	YES	YES

Table 1-1 Map of the APROM Flash Size for M032 Series

1.2 FatFs file system and related functions:

Official website: http://elm-chan.org/fsw/ff/00index_e.html

FatFs is a FAT file system written in C language. Its application layer and driver layer are completely separated. The driver layer has been mainly modified to be associated with the read and write function of M032's Data Flash. The application interface and driver functions of the FatFs are listed as follows.

1.2.1 Application Interface:

- f_mount - Register/Unregister a work area
- f_opendir - Open a directory
- f_open - Open/Create a file
- f_read - Read file
- f_write - Write file
- f_close - Close an open file
- .....

The above six functions are mainly used. There are many other functions in the application interface, and more details can be found on the official website.

1.2.2 Disk I/O Interface:

- disk_initialize - Initialize disk drive
- disk_status - Get disk status
- disk_read - Read sector(s)
- disk_write - Write sector(s)
- disk_ioctl - Control device dependent features
- get_fattime - Get current time

1.3 ICP Download

Connect to the ICP tool and download the firmware to the chip. Download USBD_Mass_Storage_Flash_Fatfs.bin and chip configuration is shown in Figure 1-1 and Figure 1-2.

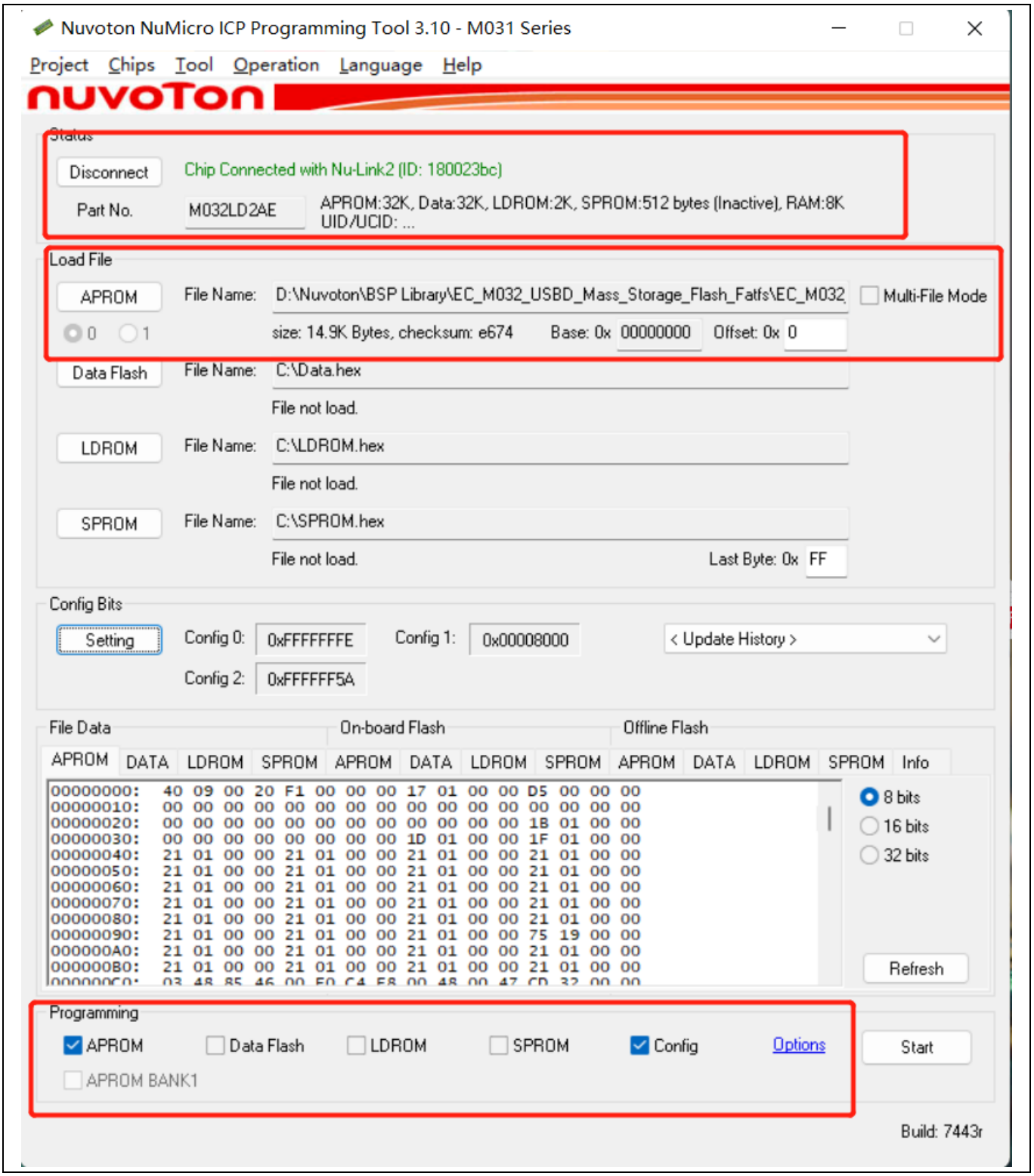

Figure 1-1 ICP Configuration

Chip settings are shown as follows:

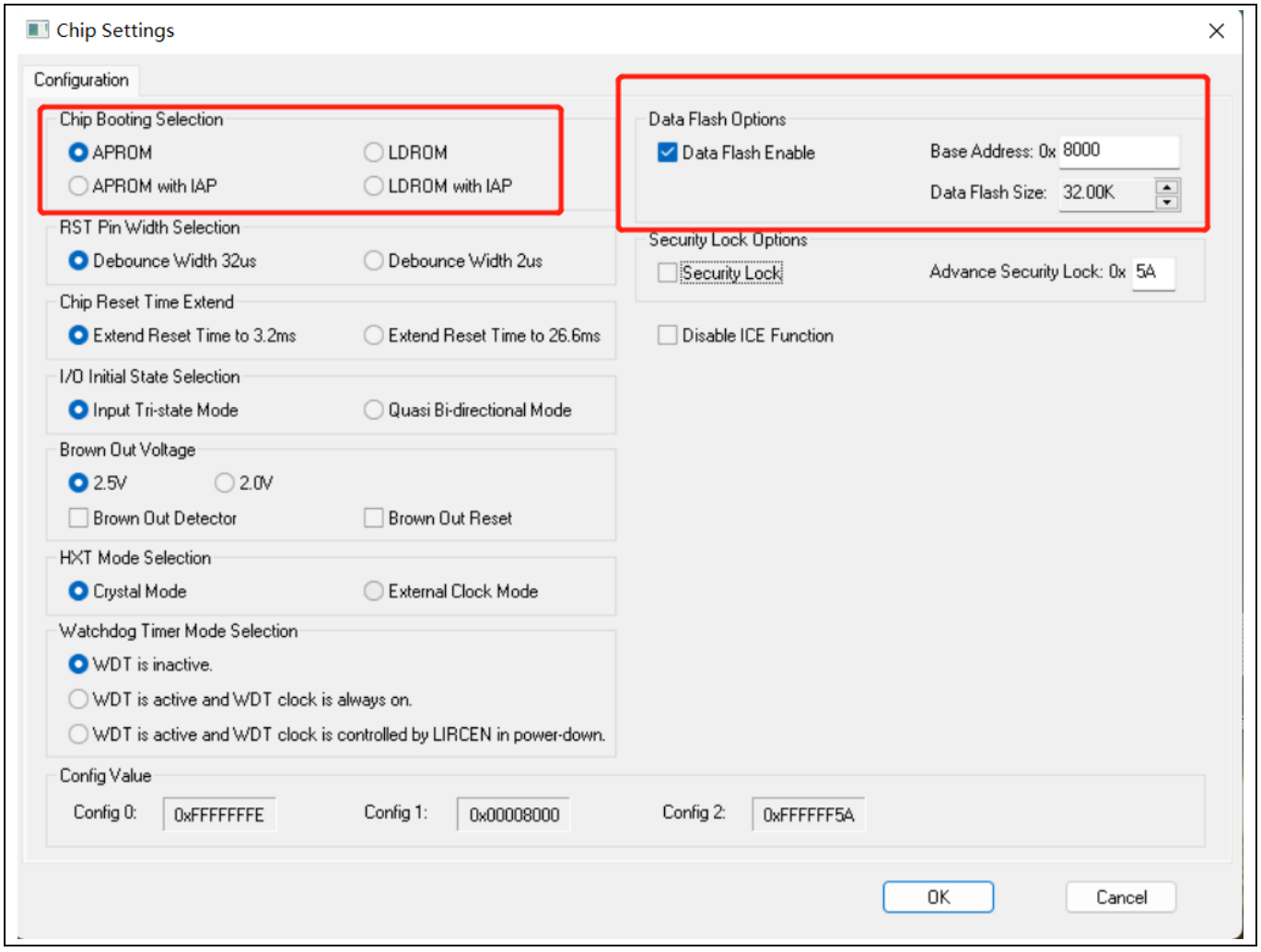

Figure 1-2 Chip Settings

If other part numbers in the M032 series are used, the capacity of Data Flash should be allocated 32 Kbytes because the USB disk needs to occupy 32 Kbytes of Data Flash area. It is recommended to upgrade the Nu-link2-Me V1.0 firmware to the version 3.10 or above.

1.4 Demo Result

When the sample code runs, the USB interface on the evaluation board is plugged into the USB interface on the computer through the MicroUSB adapter cable, then the drive code is displayed on PC side, and the disk is formatted in FAT/512byte format. On the PC side a file named "Rtest.txt" with the contents of "0123456789" is created, and the file Rtest.txt is copied to the disk, as shown in Figure 1-3 :

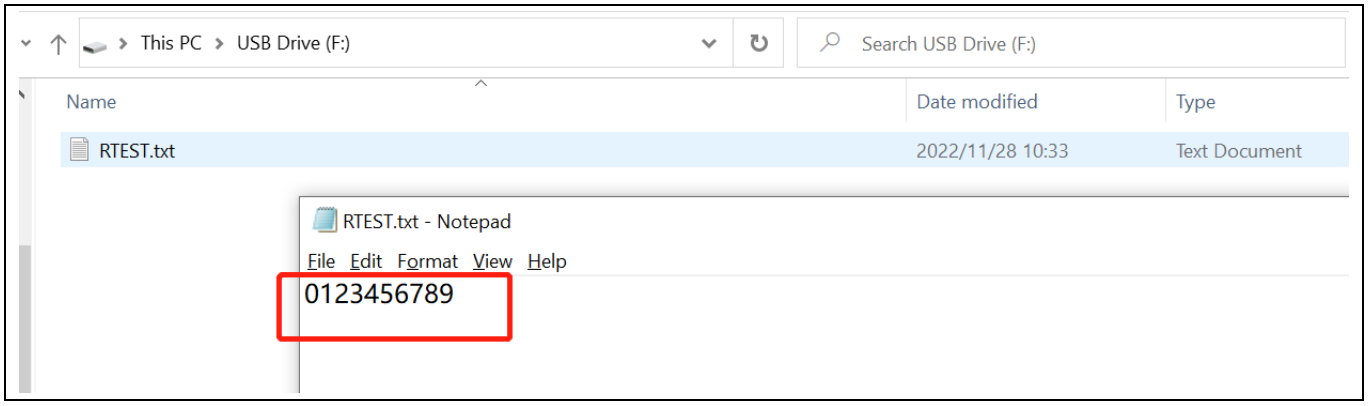

Figure 1-3 Contents of the Txt Format File Copied to USB Disk

Press the reset key once to reset the MCU. After reset successfully, the M032 creates the "Wtest.txt" file and executes a string of characters "USBD MSC+FATFS Demo \r\n www.nuvoton.com \r\n", and then the printed information is as Figure 1-4:

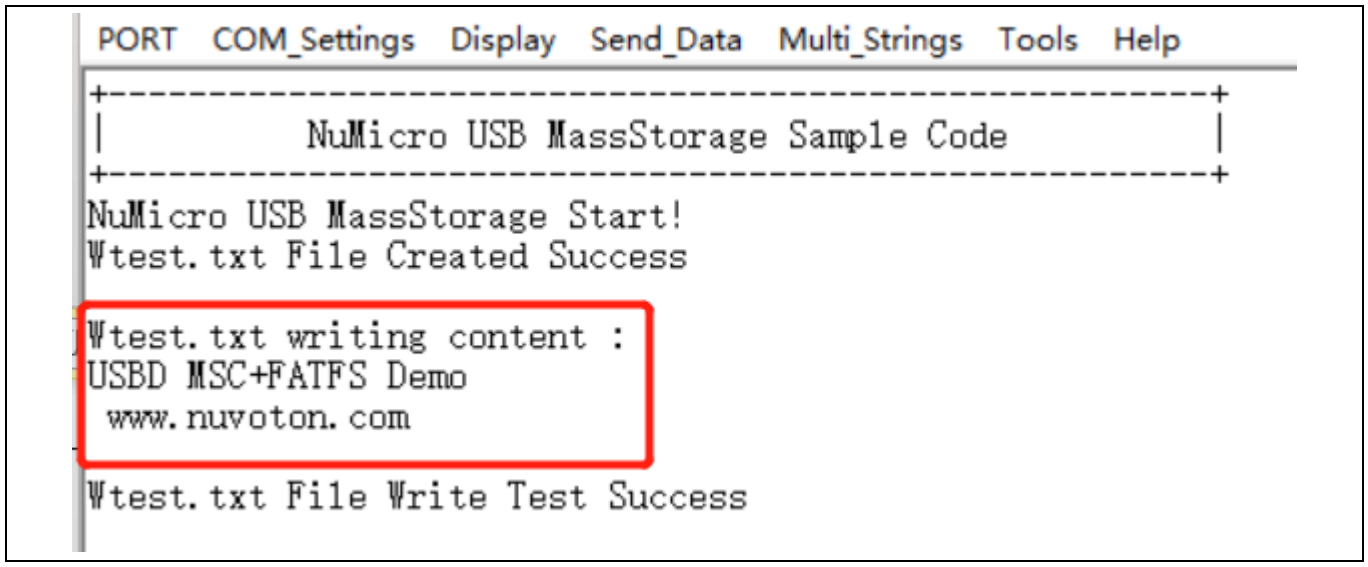

Figure 1-4 Write the File to USB Disk for Information Printing

Double-click on the computer to open the file "Wtest.txt" in the USB drive, and the contents of the file are shown in Figure 1-5:

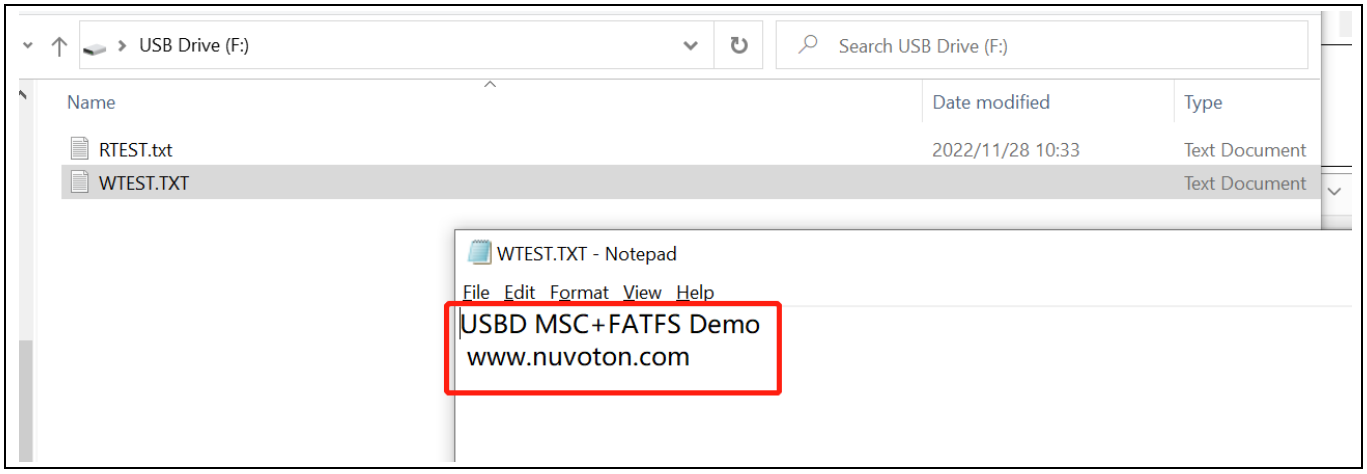

Figure 1-5 Show the Contents of the File in USB Disk

It can be seen that the contents printed by the "Wtest.txt" file are consistent with the contents written by the M032 in Figure 1-4, which proves that M032 writes files to the USB Flash drive through the FatFs file system correctly.

The M032 checks the contents of Rtest.txt on the USB disk through the FatFs file system driver and prints them out, the printing information is shown in Figure 1-6:

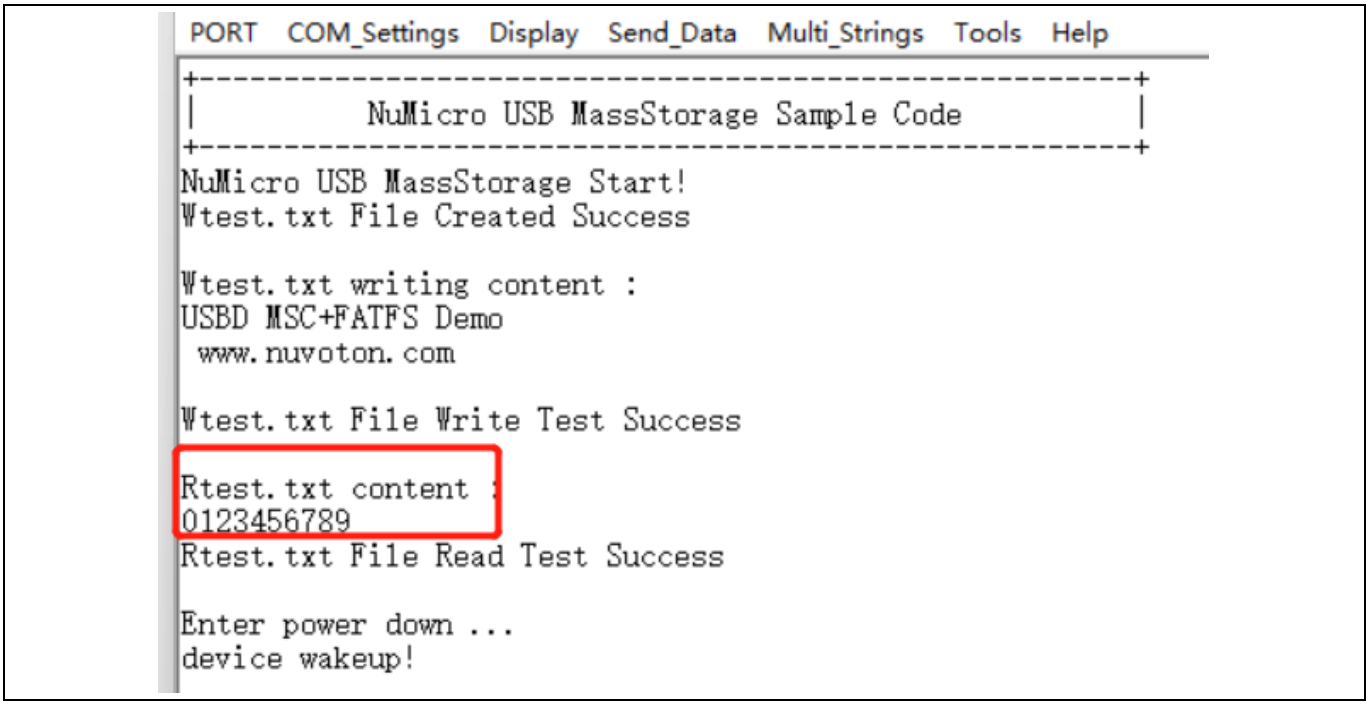

Figure 1-6 Print the Contents of the USB Disk File

It can be seen that the contents printed from the "Rtest.txt" file are "0123456789", which is consistent with the actual contents of Rtest.txt. The M032 reads that files from the FatFs file system to the USB disk are running correctly.

Note that if the following print prompt appears, the compiler option PAGE_SIZE_2048 needs to be enabled in the code. This is because the Flash format of some part numbers in the M032 series is 2048 bytes per page, so the option needs to be enabled.

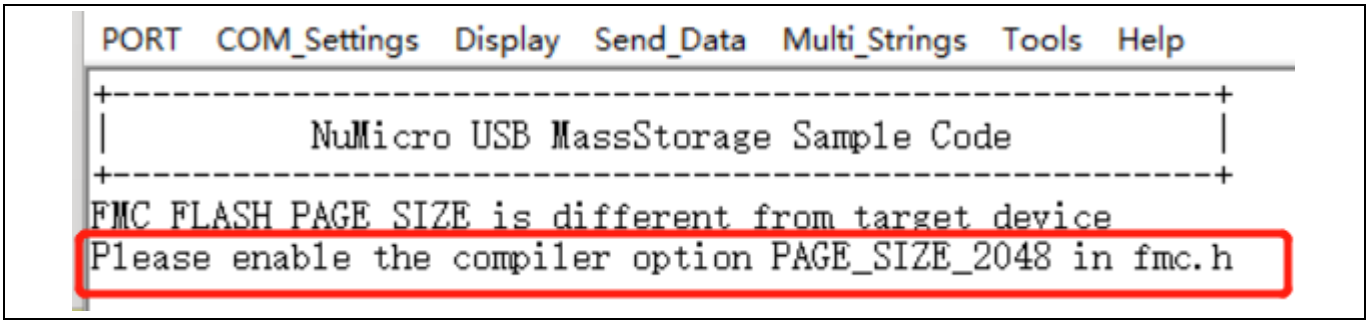

Figure 1-7 Pointer to Enable the Compiler Option PAGE_SIZE_2048

If the following print prompt appears, the compiler option PAGE_SIZE_2048 needs to be disabled in the code. This is because the Flash format of some part numbers in the M032 series is 512 bytes per page, so the option needs to be disabled.

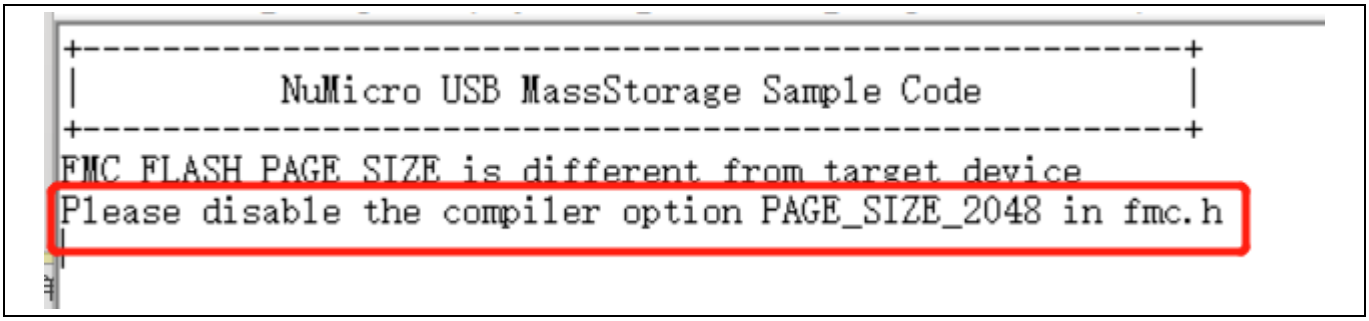

Figure 1-8 Pointer to Disable the Compiler Option PAGE_SIZE_2048

If the following print prompt appears, the CONFIG0 and CONFIG1 need to be changed in the ICP. This is because the Data Flash is used to simulate the USB mass storage space and will occupy 32 Kbytes.

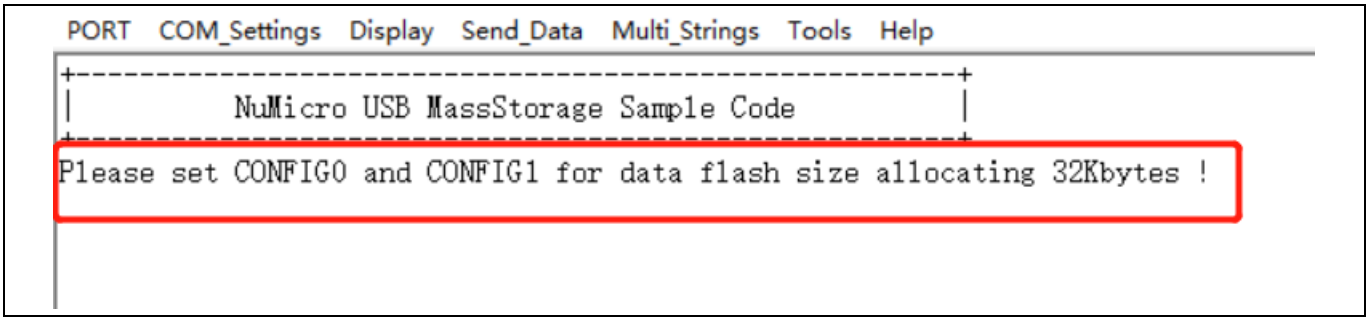

Figure 1-9 Pointer to Change the Configuration

2. Code Description

Create the ReadFileData function in main. c to read and print the contents of the "Rtest.txt" file in the root directory of the USB disk. Change the parameter "Rtest.txt" to the new file name in the f_open() function for reading the contents of the new file.

```
/**
  * @brief    Read file data from usb_msc in FatFs file systerm
  *
  *
  * @details  Read max 128 characters of "Rtest.txt" and print the content by Uart.
  *
  */
void ReadFileData(void)
{
    uint8_t buf[128] = {0}; /*max read 128 bytes content*/

    /* mount the file system */
    result = f_mount(&fs, " ", 1);
    if (result != FR_OK)
    {
        /* File System Mounted Failed (result)*/
        return;
    }

    /* open root directory */
    result = f_opendir(&DirInf, "/");
    if (result != FR_OK)
    {
        /*Root Directory is Open Error (result) */
        return;
    }
    else
    {
        /* open file name "Rtest.txt" */
        result = f_open(&file, "Rtest.txt", FA_OPEN_EXISTING | FA_READ);
        if (result !=  FR_OK)
        {
            /*Don't Find File : Rtest.txt*/
            return;
        }
        else
        {
            /* read file Rtest.txt */
            result = f_read(&file, &buf, sizeof(buf) - 1, &bw);
            if (bw > 0)
            {
                buf[bw] = 0;

                /*compare buffers */
                if(memcmp(g_au8Backup_buf, buf, sizeof(buf)) != 0)
                {
                    printf("\r\nRtest.txt content : \r\n%s\r\n", buf);
                    /*renew backup_buf */
```

```
                    memcpy(g_au8Backup_buf, buf, sizeof(buf));
                    printf("Rtest.txt File Read Test Success\r\n");
                }
            }
            else  /*no content*/
            {
                /*no content,clear buffer*/
                memcpy(g_au8Backup_buf, buf, sizeof(buf));
                printf("Rtest.txt File Read Nothing\r\n");
            }
        }
    }

    /* colse file*/
    f_close(&file);
}
```

Create the WriteFileData function in main.c to write a string of characters "USBD MSC+FATFS Demo \r\n www.nuvoton.com \r\n" to the "Wtest.txt" file in the root directory of the USB disk.

```
/**
  * @brief    Write data to usb_msc in FatFs file systerm
  *
  *
  * @details  Write a string of characters to "Wtest.txt".
  *
  */
void WriteFileData(void)
{
uint8_t buf[128] = {0};    /*max read 128 bytes content*/

    /* mount the file system */
    result = f_mount(&fs, " ", 1);          /* Mount a logical drive */
    if (result != FR_OK)
    {
        /* File System Mounted Failed (result)*/
        return;
    }

    /* open root directory */
    result = f_opendir(&DirInf, "/");
    if (result != FR_OK)
    {
        /*Root Directory is Open Error (result) */
        return;
    }

    /* open file name "Wtest.txt" */
    result = f_open(&file, "Wtest.txt", FA_OPEN_EXISTING | FA_READ);

    if (result !=  FR_OK)              /*Don't Find File : Wtest.txt*/
    {
        /* Create file name "Wtest.txt" */
        result = f_open(&file, "Wtest.txt", FA_CREATE_ALWAYS | FA_WRITE);
```

```
        if (result !=  FR_OK)
        {
            /* Create file faild */
            return;
        }
        else
        {
            /* Create file success */
            printf("Wtest.txt File Created Success\r\n");
        }
    }
    else                    /*Find File : Wtest.txt*/
    {
        /* read file Wtest.txt */
        result = f_read(&file, &buf, sizeof(buf) - 1, &bw);

        if (bw > 0)
        {
            buf[bw] = 0;
            printf("\r\nWtest.txt content : \r\n%s\r\n", buf);
            printf("Wtest.txt File Write Test Success\r\n");
            return;
        }
        else  /*no content*/
        {
            printf("Wtest.txt File Read Nothing\r\n");
            printf("Wtest.txt File Start Writing\r\n");
            /* Open file name "Wtest.txt" in write mode*/
            result = f_open(&file, "Wtest.txt", FA_CREATE_ALWAYS | FA_WRITE);

            if (result !=  FR_OK)
            {
                /* Open file faild */
                return;
            }
        }
}

    /* Write a string of data */
    result = f_write(&file, "USBD MSC+FATFS Demo \r\n www.nuvoton.com \r\n", 38, &bw);
    if (result == FR_OK)
{
        printf("\r\nWtest.txt writing content : \r\n%s\r\n",
"USBD MSC+FATFS Demo \r\n www.nuvoton.com \r\n");
        printf("Wtest.txt File Write Test Success\r\n");
    }
    else
    {
        printf("Wtest.txt File Write Failed");
    }

    /* colse file*/
    f_close(&file);
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M031_Series_BSP_CMSIS_V3.05.000
- IDE version
  - Keil uVersion 5.28

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M032LD V1.1
- Pin Connect

The hardware uses the M032 board to connect the computer, as shown in Figure 3-1.

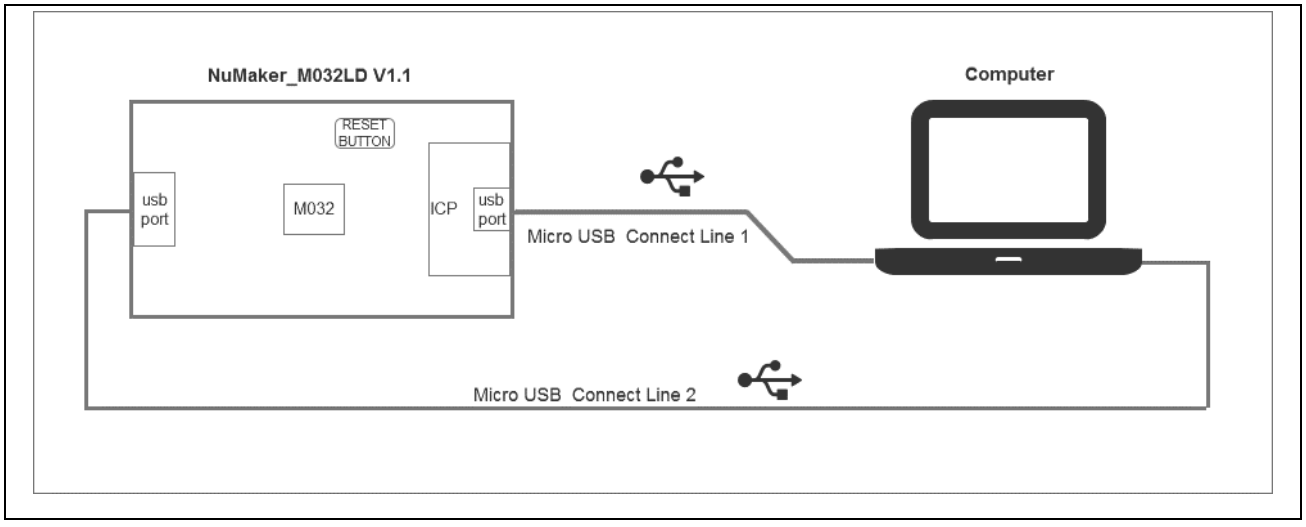

Figure 3-1 Hardware Connection Diagram

As Figure 3-1, the MicroUSB Connection Line 1 is connected with the ICP debugging circuit of the evaluation board and the USB port of computer, and mainly plays the role of serial port for printing and program debugging. The MicroUSB Connection Line 2 is connected with the USB port of the M032 and the other USB port of computer, and mainly used for USB communication between the M032 and computer when the sample code is running.

4. Directory Information

The directory structure is shown below.

EC_M032_USBD_Mass_Storage_Flash_FatFs_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
ThirdParty	
FatFs	A generic FAT file system module for small embedded systems. Its official website is: http://elm-chan.org/fsw/ff/00index_e.html.

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M032_USBD_Mass_Storage_Flash_FatFs.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2022.11.11	1.00	Initial version.

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.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.