

Contents

1、	Introduction	1
2、	Bootloader's implementation	1
3、	Run the demo	4

1、 Introduction

Previously we used sd card to upgrade the program. We have to insert the sd card into the computer every time, copy the program to the sd card, and then insert it into the sd card slot of mcu to update the program. This method seems to be a bit troublesome, so we implemented a more convenient method. It no longer needs to insert or remove the SD card from PC and MCU. Use the usb function of mcu to recognize mcu as a storage U disk. When we need to update the program, connect the MCU's usb interface to PC. After the computer recognizes it, copy the program that needs to be burned in. Then the bootloader will recognize the file and then upgrade the application. Bootloader detects changes of the file, not the existence of the file. In other words, if the a000.bin file has already existed in the sd card, the application will not be updated. When this a000.bin is overwritten with another a000.bin, the operation of updating the application will be performed.

2、 Bootloader's implementation

The schematic for SD card is shown below. The board uses SDHC module to communicate with SD card.

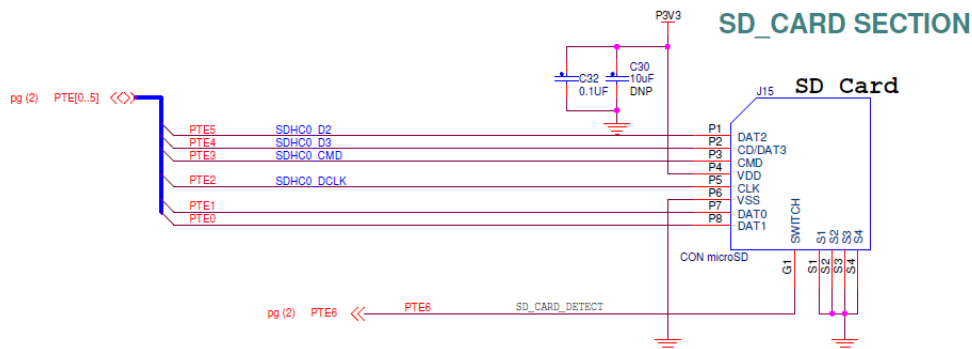


Figure 1. Figure 1.Schematic for SD card

We use the 2.6.0 version of FRDM-K64F's SDK. You can download the SDK in our website. The link is "mcuxpresso.nxp.com".

The schematic for USB is shown below.

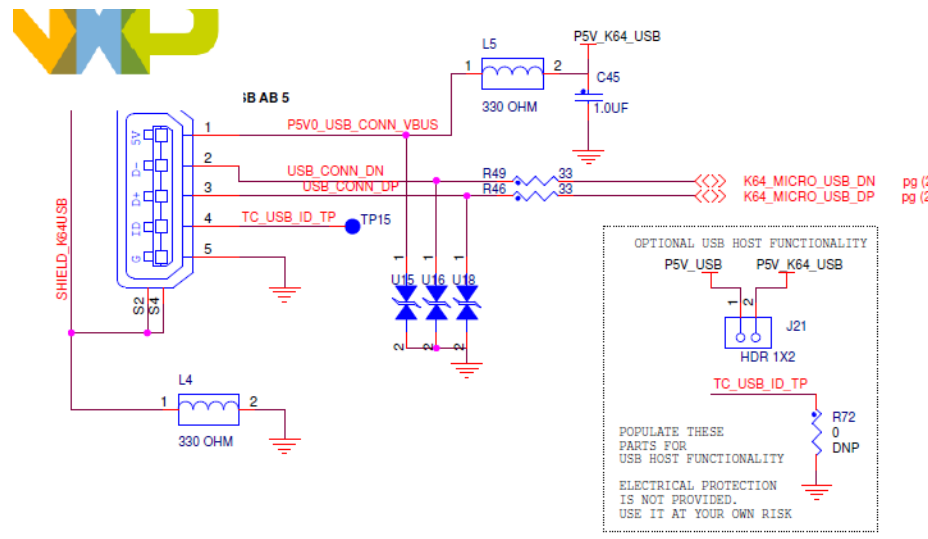


Figure 2. Schematic of USB

Bootloader uses SDHC, fatfs, usb, flash, So we should add files to support them. Our code is based on the example "usb_device_msc_sdcard_lite" that belongs to usb example.

In main code, the program will initialize the usb, sd and fatfs. Then the computer will communicate with MCU. Finally, PC will recognize the mcu as a u-disk.

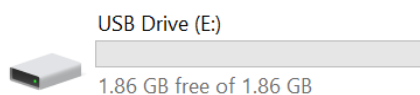


Figure 3.u-disk

The method of how to update the program and prepare the application has been written in this document. You can refer to it. <https://community.nxp.com/docs/DOC-344903>

Use a variable "wrFlag" to check the modification of the file. When we put a file into the u-disk, this variable will be set.

```

usb_status_t USB_DeviceMscProcessUfiCommand(usb_device_msc_struct_t *mshandle)
{
    usb_status_t error = kStatus_USB_Error;
    usb_device_msc_ufi_struct_t *ufi = NULL;
    ufi = &mshandle->mscUfi;
    if (USB_DEVICE_MSC_REQUEST_SENSE_COMMAND != mshandle->mscCbw->cbwcb[0])
    {
        ufi->requestSense->senseKey = USB_DEVICE_MSC_UFI_NO_SENSE;
        ufi->requestSense->additionalSenseCode = USB_DEVICE_MSC_UFI_NO_SENSE;
        ufi->requestSense->additionalSenseQualifier = USB_DEVICE_MSC_UFI_NO_SENSE;
    }
    ufi->thirteenCase.hostExpectedDataLength = mshandle->mscCbw->dataTransferLength;
    ufi->thirteenCase.hostExpectedDirection = (uint8_t)(mshandle->mscCbw->flags >> USB_DEVICE_MSC_CBW_DI);
    /*The first byte of all ufi command blocks shall contain an Operation Code, refer to ufi spec*/

    switch (mshandle->mscCbw->cbwcb[0])
    {
        /* ufi command operation code*/
        case USB_DEVICE_MSC_INQUIRY_COMMAND: /*operation code : 0x12*/
            error = USB_DeviceMscUfiInquiryCommand(mshandle);
            break;
        case USB_DEVICE_MSC_READ_10_COMMAND: /*operation code : 0x28 */
        case USB_DEVICE_MSC_READ_12_COMMAND: /*operation code : 0xA8 */
            error = USB_DeviceMscUfiReadCommand(mshandle);
            break;
        case USB_DEVICE_MSC_REQUEST_SENSE_COMMAND: /*operation code : 0x03*/
            error = USB_DeviceMscUfiRequestSenseCommand(mshandle);
            break;
        case USB_DEVICE_MSC_TEST_UNIT_READY_COMMAND: /*operation code : 0x00 */
            error = USB_DeviceMscUfiTestUnitReadyCommand(mshandle);
            break;
        case USB_DEVICE_MSC_WRITE_10_COMMAND: /*operation code : 0x2A */
        case USB_DEVICE_MSC_WRITE_12_COMMAND: /*operation code : 0xAA */
            error = USB_DeviceMscUfiWriteCommand(mshandle);
            break;
        case USB_DEVICE_MSC_PREVENT_ALLOW_MEDIUM_REM_COMMAND: /*operation code :0x1E */
            error = USB_DeviceMscUfiPreventAllowMediumCommand(mshandle);
            break;
        case USB_DEVICE_MSC_WRITE_10_COMMAND: /*operation code : 0x2A */
        case USB_DEVICE_MSC_WRITE_12_COMMAND: /*operation code : 0xAA */
            wrFlag = 1;
            break;
    }
}

```

Figure 4. Modification of flag

When this variable is set, the program will open the "a000.bin". Then update the application. Finally, go to the application.

```

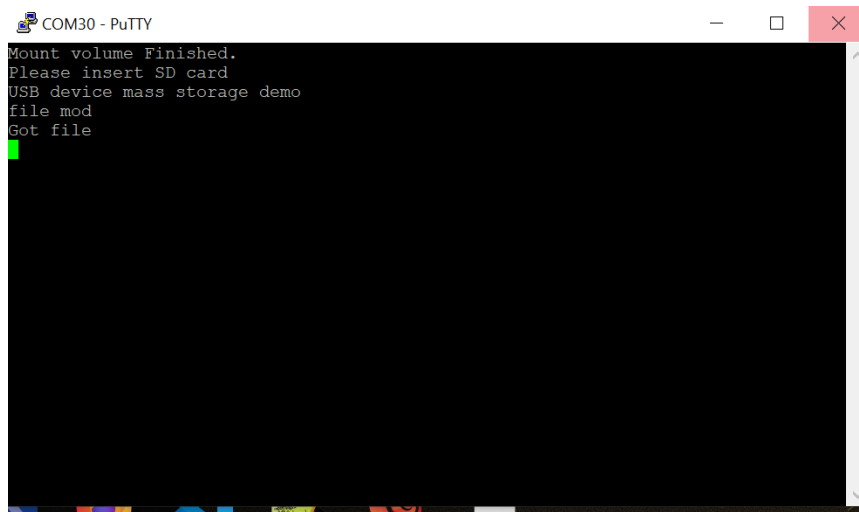
FRESULT USB_DeviceMscApp(void)
{
    FRESULT error = FR_OK;
    if(wrFlag == 1)
    {
        SDMMCHOST_Delay(300); //because When put file into the disk, the commar
        wrFlag = 0;
        PRINTF("file mod\r\n");
        error = f_open(&g_fileObject, _T("/a000.bin"), (FA_WRITE | FA_READ));
        if(FR_NO_FILE == error)
        {
            PRINTF("No file\r\n");
        }
        if(FR_OK == error)
        {
            PRINTF("Got file\r\n");
            update_bootloader(&g_fileObject);
            f_close(&g_fileObject);
            USB_DeviceStop(g_msc.deviceHandle);
            USB_DeviceDeinit(g_msc.deviceHandle);
            //jump to App();
            deinit();
            SD_Deinit(&g_sd);
            static void (*farewellBootloader)(void) = 0;
            farewellBootloader = (void (*)(void))(APP_VECTOR_TABLE[1]);
            __set_MSP(APP_VECTOR_TABLE[0]);
            __set_PSP(APP_VECTOR_TABLE[0]);
            SCB->VTOR = (uint32_t)APP_VECTOR_TABLE;
            farewellBootloader();
            while(1);
        }
    }
    // f_close(&g_fileObject);
}
/*TO DO*/
/*add user code*/
return error;
}

```

Figure 5. Update the application

3、 Run the demo

- 1、 Download this bootloader
- 2、 Prepare a user application program. We use the “led blinky” as an example. Use it to generate the binary file. Name it as “a000.bin”. Put it into the u-disk. You will see some log in the uart.



```
COM30 - PuTTY
Mount volume Finished.
Please insert SD card
USB device mass storage demo
file mod
Got file
```

- 3、 The application will execute automatically

