FRDMK64F SD Bootloader

Contents

[1、Introduction 1](#_Toc20669674)

[2、Bootloader’s implementation 1](#_Toc20669675)

[3、Memory relocation 7](#_Toc20669676)

[4、Run the demo 9](#_Toc20669677)

[5、Reference 9](#_Toc20669678)

# 1、Introduction

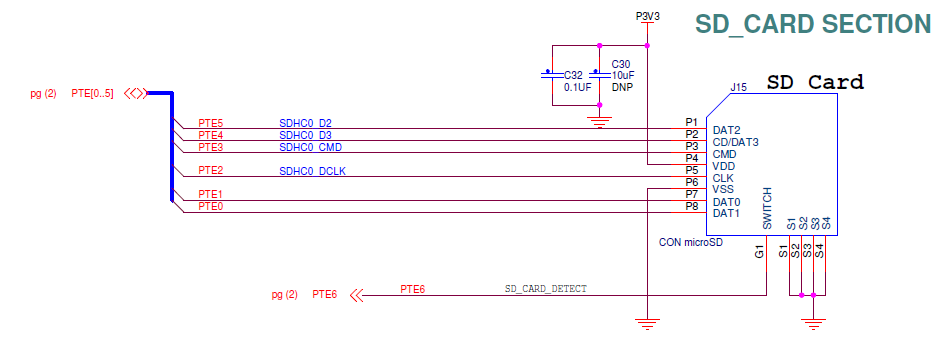
As is known to all, we use debugger to download the program or debug the device.

FRDMK64 have the opsenSDA interface on the board, so wo do not need other’s debugger. But if we want to design a board without debugger but can download the program, we can use the bootloader. The bootloader is a small program designed to update the program with the interface such as UART,I2C,SPI and so on.

This document will describe a simple bootloader based on the FRDMK64F.The board uses SD card to update the application. User can put the binary file into the card. When the card insert to the board ,the board will update the application automatically. The bootloader code and application code are all provided so that you can test it on your own board.

# 2、Bootloader’s implementation

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

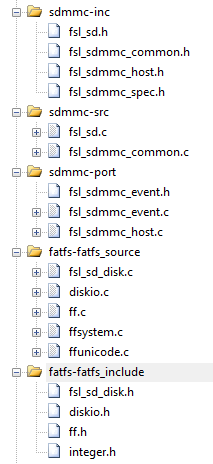


**Figure 1.Schematic for SD card**

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”.

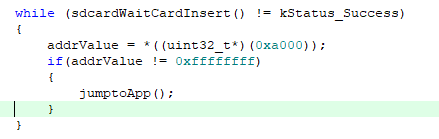
The bootloader uses SDHC and fafts file system. So we should add files to support it.



**Figure 2.The support file**

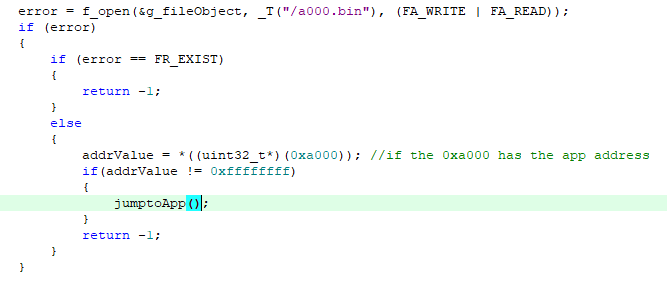
In main code, the program will wait until the card has inserted. Then it will find the file named “a000.bin” in sd card to update the application. If the file do not exist, the board will directly execute the application. If there is no application, the program will end.

The following code shows how the program wait for inserting sd card. It will also check if the address has the application’s address.



**Figure 3.The code -- wait for inserting card**

The following code shows how the program opens the binary file. If sd card doesn’t have the file, the program will go to the application.



**Figure 4.Open the binary file**

If the program opens the file normally, the update will begin. It will erase 200k’s space from 0xa000. You can adjust it according to your project.

Now I will explain update’s method in detail. Our data is written to the buffer called “rBUff”. The buffer size is 4K. Before write data to it, it is cleared.

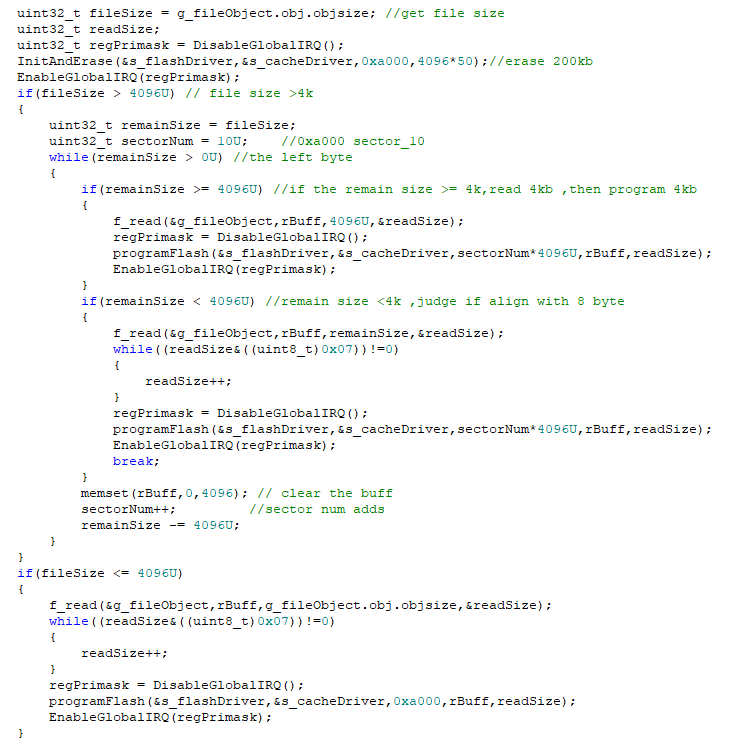
Please note that when we erase or program the flash, we should disable all interrupts and when the operations finish we should enable the interrupts.

The file size will decide which way to write the data to flash.

1、If the size < 4k ,we just read the file’s data to buffer and judge if its size aligned with 8 byte. If not , we increase the size of “readSize” to read more data in our data buffer called “rBuffer”. The more data we read is just 0.

2、If the size > 4K, we use “remainSize” to record how much data is left. We read 4k each time until its size is smaller than 4k and then repeat step 1. When finish the operation at a

time, we should clear the buffer and increase the sector numer to prepare the next transmission.



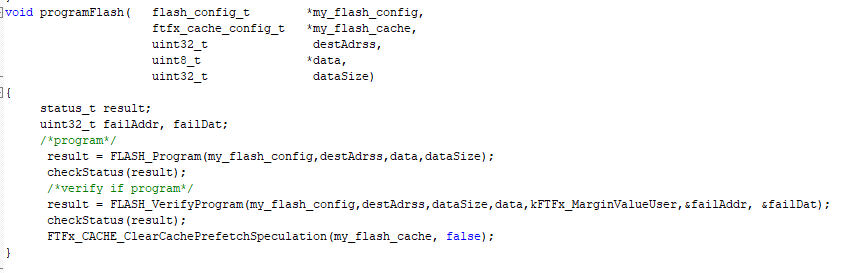
**Figure 5.Write flash operation code**

The way to clear the space is shown in the figure. It will initialize the flash and erase the given size from the given address. “SectorNum” is used to show which sector to erase.



**Figure 6.Erase operation code**

The following figure shows how to write the data to flash.

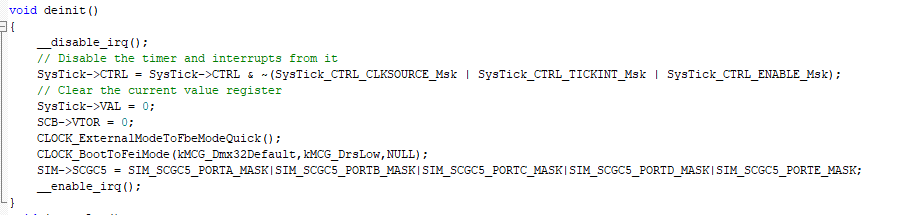


**Figure 7.Program operation code**

Before we go to the application, we should modify the configuration we did in the bootloader.

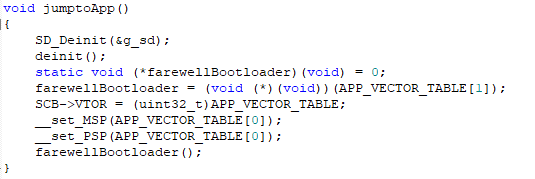
1. Close the systick, clear its value.
2. Set the VTOR to default value.
3. Our bootloader runs in PEE mode. So we should change it to FEI mode.
4. Disable the all pins.

You should disable the global interrupt when run these codes. And don’t forget to enable the global interrupt.



**Figure 8.Deinitalization code**

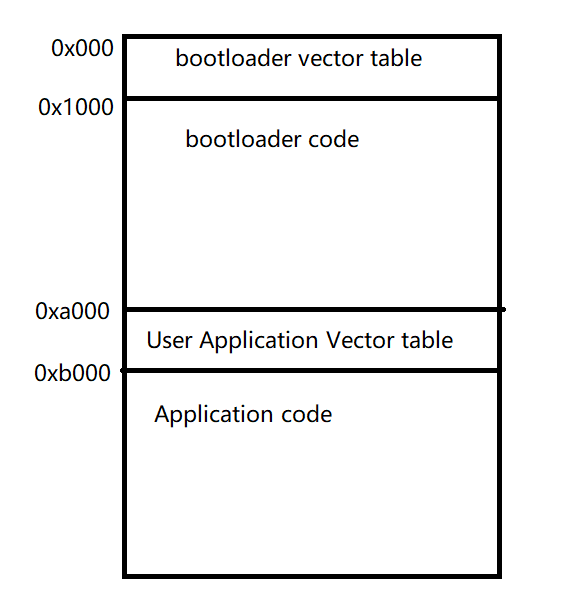
Then we can go to the application.



**Figure 9.Go to Application**

# 3、Memory relocation

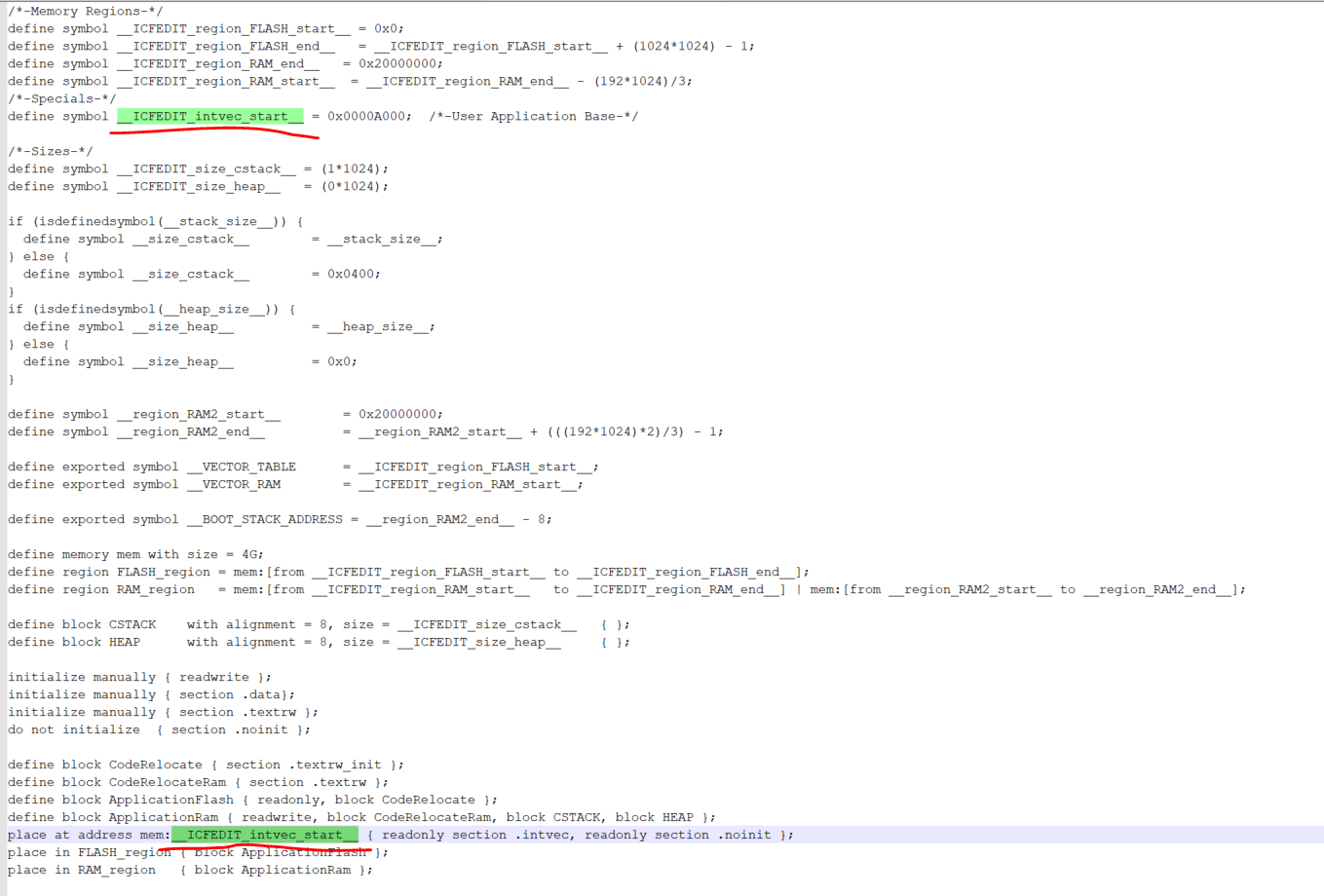
The FRDMK64 has the 1M flash, from 0x00000000 to 0x00100000.As shown in figure 10,we use the 0xa000 as the application’s start address.



**Figure 10.The memory map**

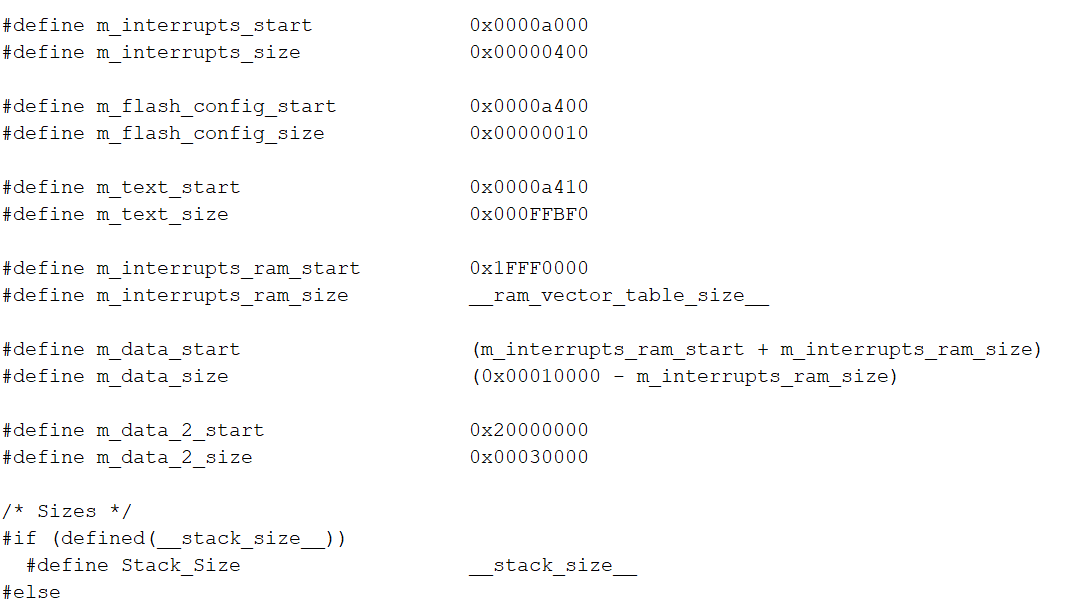
Now, I will show you how to modify the link file for user application in different IDE.

In IAR



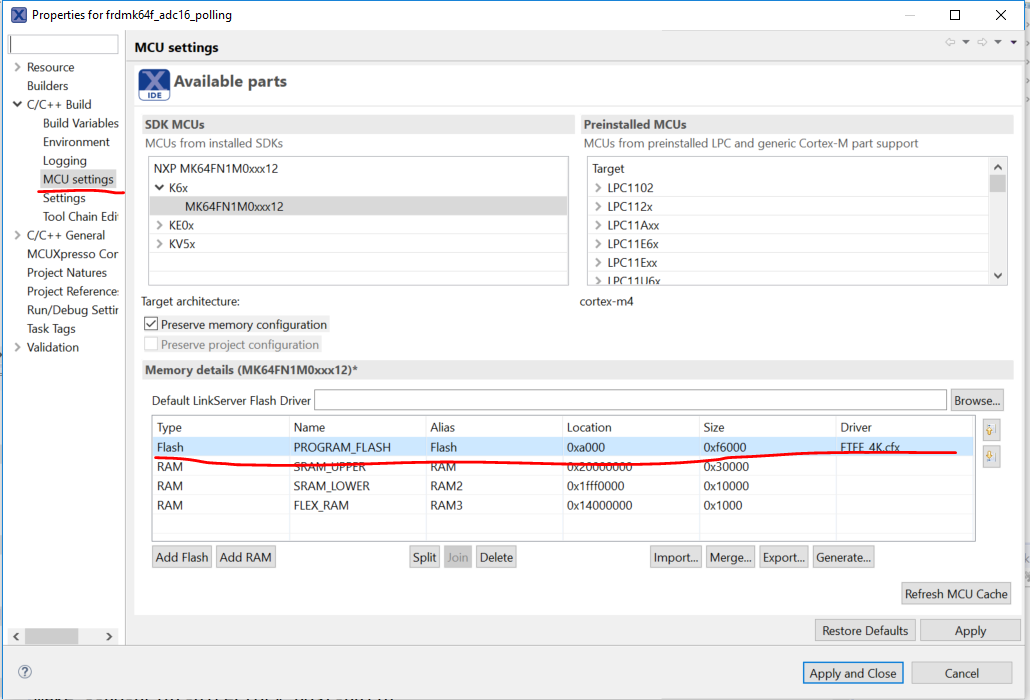
**Figure 11.IAR’s ICF**

In MDK



**Figure 12.MDK’s SCF**

In MCUXpresso



**Figure 13.MCUXpresso’s flash configuration**

# 4、Run the demo

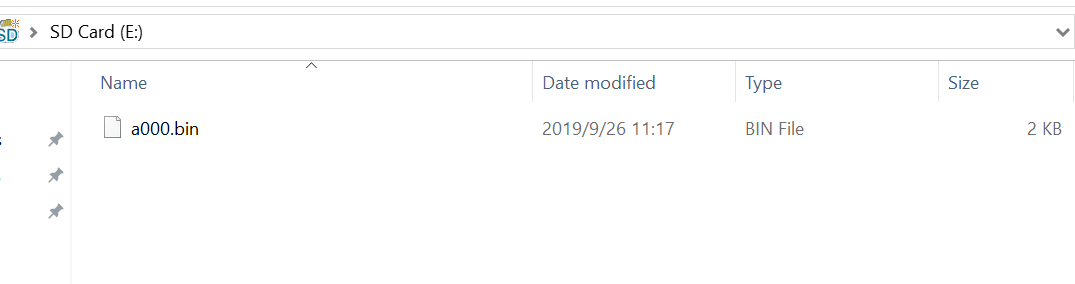
1) Download the bootloader first.

2) Prepare a user application program. We use the “led blinky” as an example.

3) Modify the Link file.

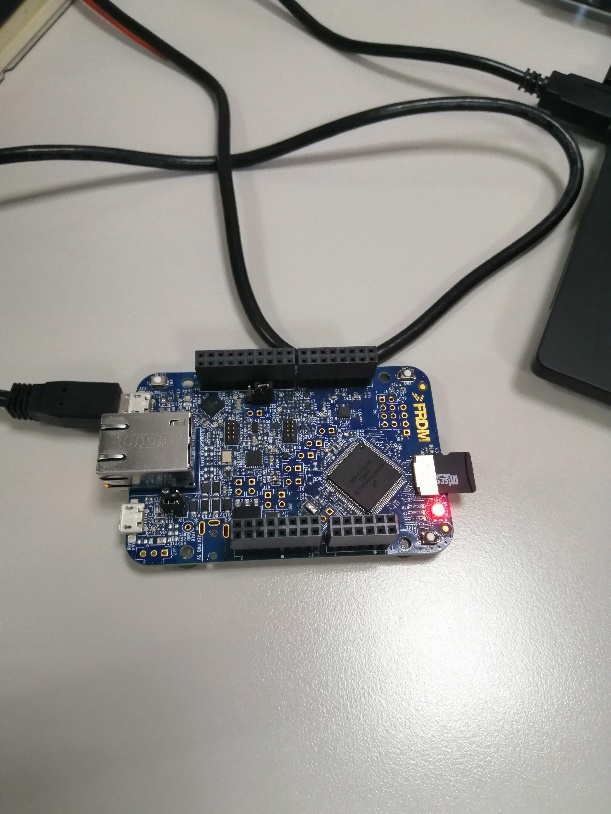
4) Generate the binary file with your IDE, please name it as “a000.bin”.

5) Put it into the sd card like figure 5.



**Figure 14.SD card’s content**

6) Insert the card. And power on. Wait for a moment, the application will execute automatically.



# 5、Reference

1) Kinetis MCU的bootloader解决方案

2) KEA128\_can\_bootloader