On-chip Flash (Flash)

I. Learning Objectives

1. Learn and understand basic flash knowledge.
2. Learn to read and write the flash of MSPM0G3507.

The DAP-LINK debugging tool used in this tutorial section needs to be prepared by yourself

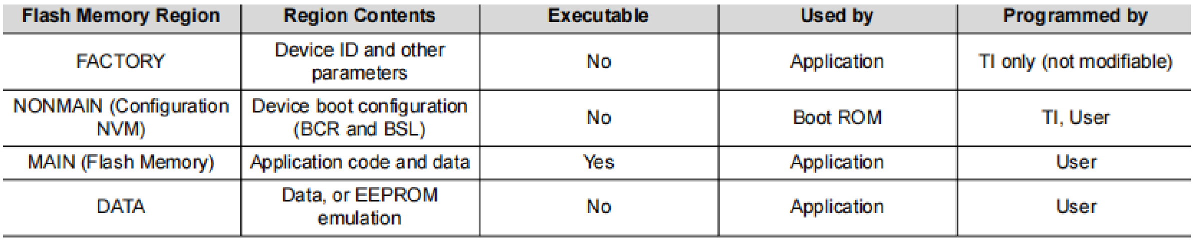
II. Flash Introduction

First, we need to understand several unit terms

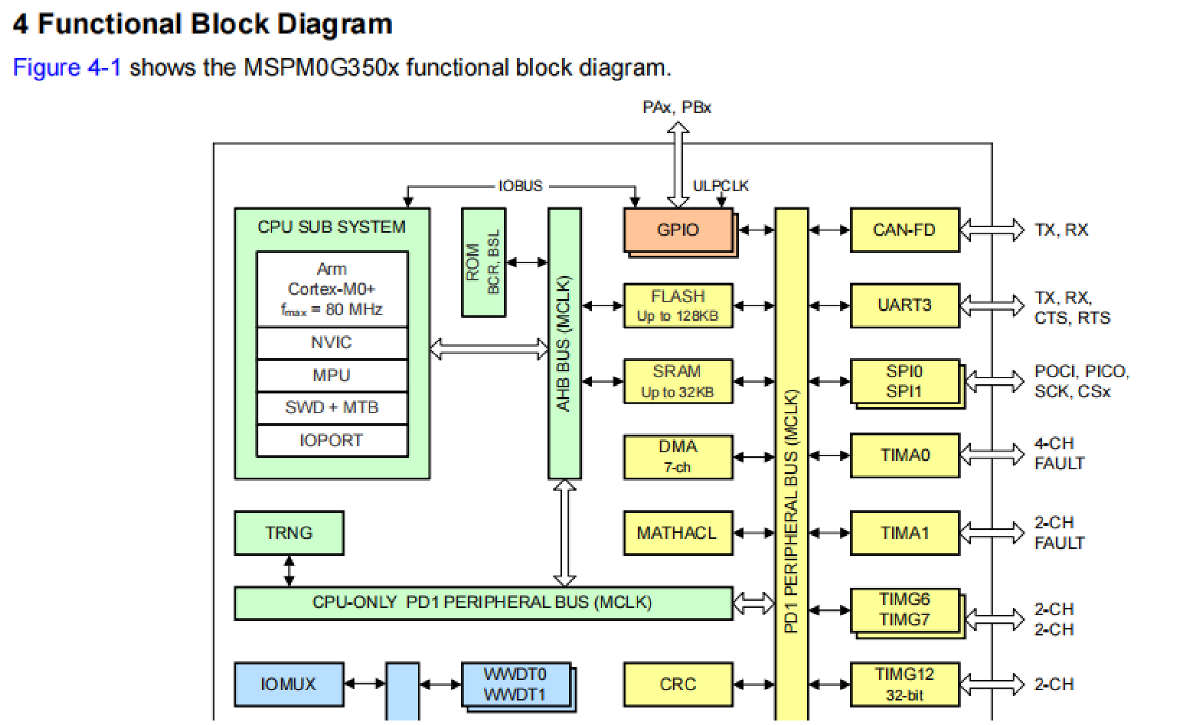
Sector: Sector (1KB)

BANK: A group of sectors that can be batch erased in a single operation; only one operation (read, program, erase, or verify) can run simultaneously on the same Bank, maximum 256 sectors

MSPM0G350x has flash memory with error correction code (ECC) and up to 128KB, where the first 32KB has 100,000 programming/erasing cycles (addresses 0x0000.0000 to 0x0000.8000), and up to 10,000 programming/erasing cycles can be performed on the remaining flash. Flash is used to store application code and data, device startup configuration, and parameters pre-programmed by TI from the factory. In this section, we will write specified data to flash starting from address 0x00001000 and then view it through simulation tools.

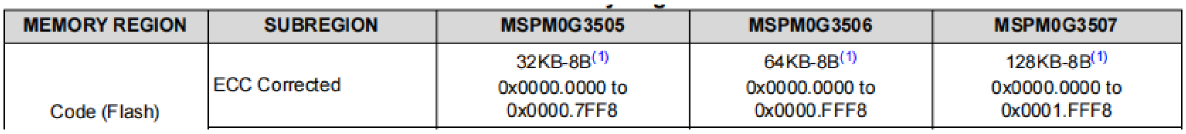


Flash Memory Region	Region Contents	Executable	Used by	Programmed by
FACTORY	Device ID and other parameters	No	Application	TI only (not modifiable)
NONMAIN (Configuration NVM)	Device boot configuration (BCR and BSL)	No	Boot ROM	TI, User
MAIN (Flash Memory)	Application code and data	Yes	Application	User
DATA	Data, or EEPROM emulation	No	Application	User



4 Functional Block Diagram

Figure 4-1 shows the MSPM0G350x functional block diagram.

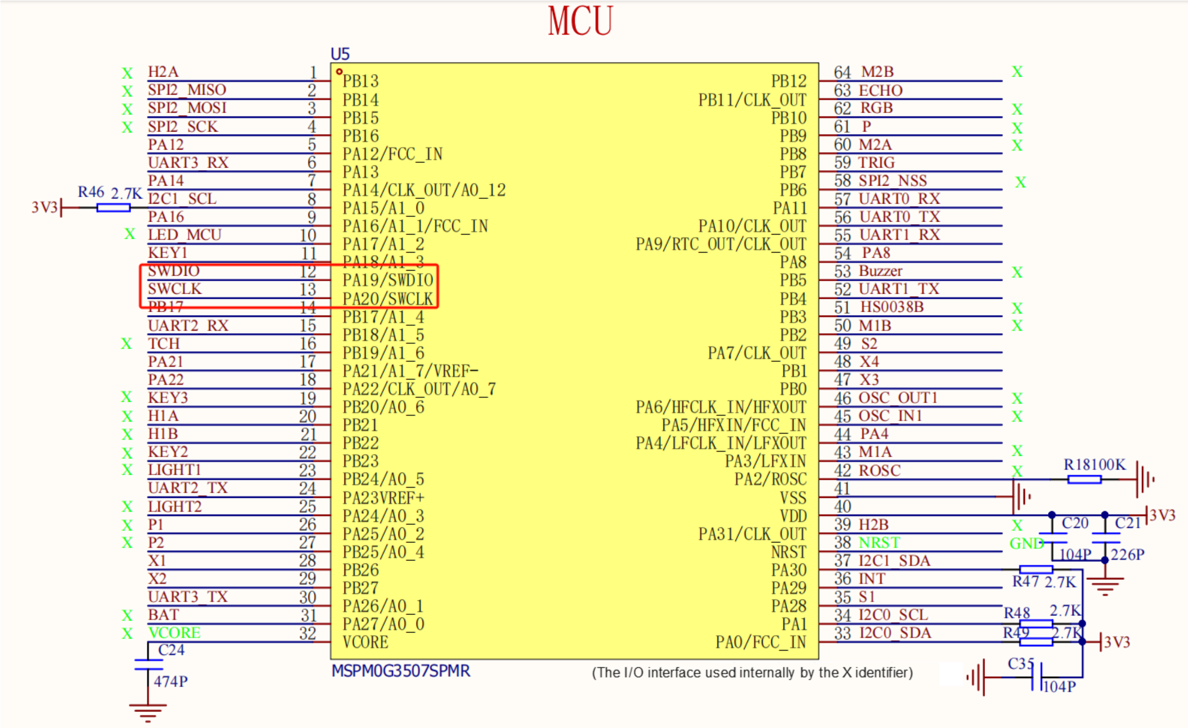


MEMORY REGION	SUBREGION	MSPM0G3505	MSPM0G3506	MSPM0G3507
Code (Flash)	ECC Corrected	32KB-8B[1] 0x0000.0000 to 0x0000.7FF8	64KB-8B[1] 0x0000.0000 to 0x0000.FFF8	128KB-8B[1] 0x0000.0000 to 0x0001.FFF8

III. Hardware Setup

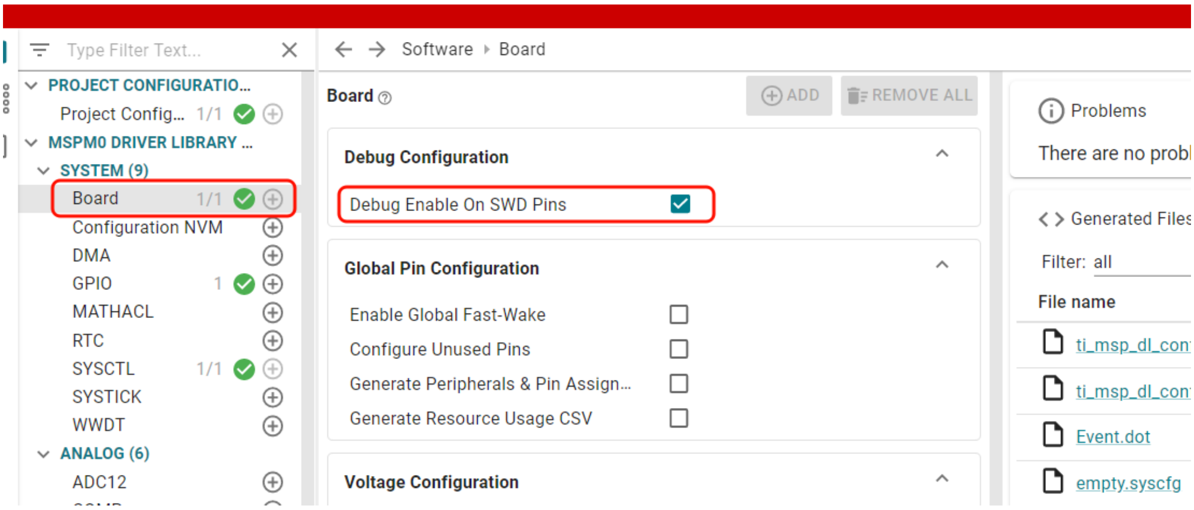
This course requires connecting a debugging tool, using DAP-Link as an example. For details, please refer to [M0 Car Development Board Instructions] SWD Debug Interface

DAP-LINK	MSPM0G3507
GND	GND
VDD or 3V3	3V3
SWCLK/CLK	SWCLK(PA20)
SWDIO/DIO	SWDIO(PA19)



IV. Software Environment Setup

In this tutorial section, we will build based on the LED project. First, we open SWD through sysconfig, then save and generate code



Board

Debug Configuration

Debug Enable On SWD Pins ☑

Global Pin Configuration

Enable Global Fast-Wake □

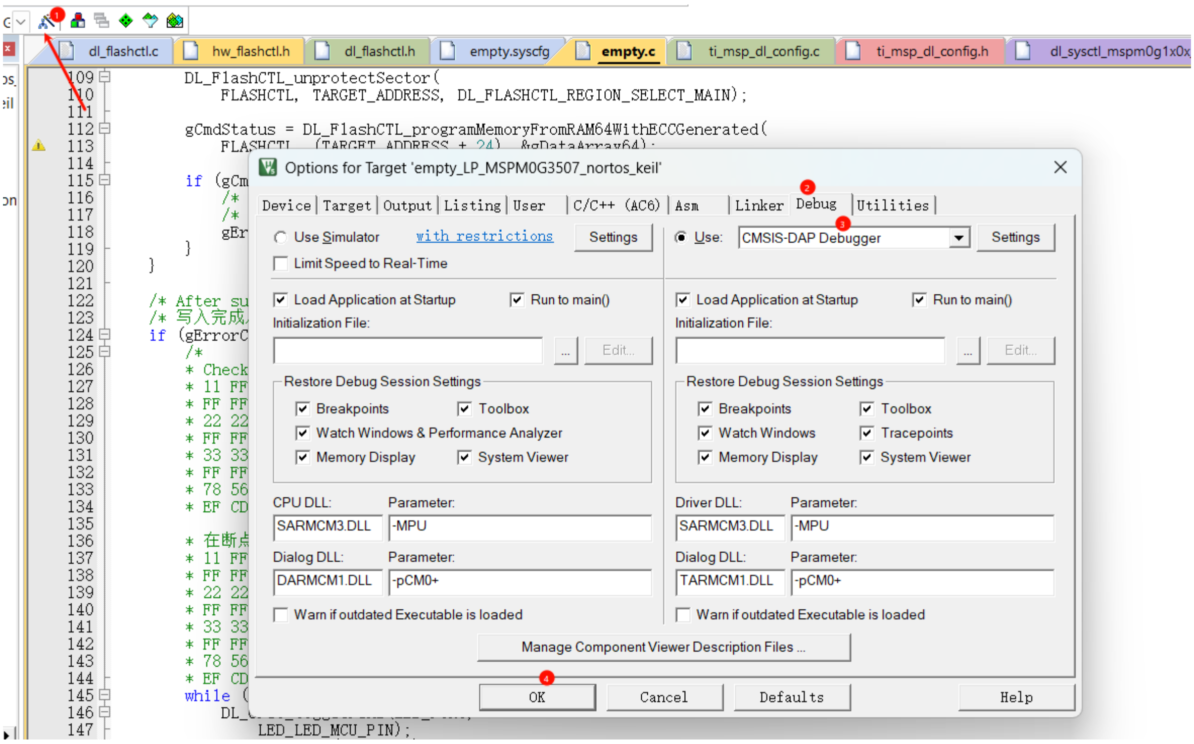
Configure Unused Pins □

Generate Peripherals & Pin Assign... □

Generate Resource Usage CSV □

Voltage Configuration

Configure the programmer, using DAP-LINK as an example here. Other programmers can be selected according to your own configuration.



```
        DL_FlashCTL_unprotectSector(
            FLASHCTL, TARGET_ADDRESS, DL_FLASHCTL_REGION_SELECT_MAIN);

        gCmdStatus = DL_FlashCTL_programMemoryFromRAM64WithECCGenerated(
            FLASHCTL, (TARGET_ADDRESS + 24), &gDataArray64);
```

Options for Target 'empty_LP_MSPM0G3507_nortos_keil'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

○ Use Simulator   with restrictions   Settings
□ Limit Speed to Real-Time
☑ Load Application at Startup   ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints   ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display   ☑ System Viewer
CPU DLL: SARMCM3.DLL   Parameter: -MPU
Dialog DLL: DARMCM1.DLL   Parameter: -pCM0+
□ Warn if outdated Executable is loaded

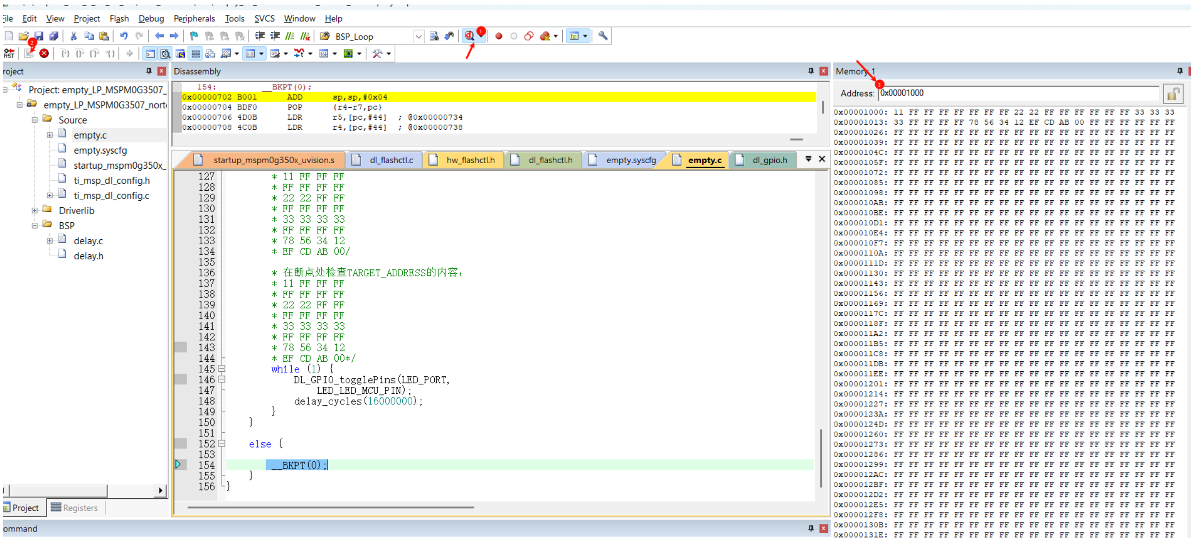
● Use: CMSIS-DAP Debugger   Settings
☑ Load Application at Startup   ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints   ☑ Toolbox
☑ Watch Windows   ☑ Tracepoints
☑ Memory Display   ☑ System Viewer
Driver DLL: SARMCM3.DLL   Parameter: -MPU
Dialog DLL: TARMCM1.DLL   Parameter: -pCM0+
□ Warn if outdated Executable is loaded

V. Code Analysis

```
Flash writing process: Unprotect -> Erase Sector/Bank -> Write specified data
    /* Unprotect sectors in main memory (hardware generates ECC) */
    DL_FlashCTL_unprotectSector(
        FLASHCTL, TARGET_ADDRESS, DL_FLASHCTL_REGION_SELECT_MAIN);

    /* Erase sectors in main memory */
    gCmdStatus = DL_FlashCTL_eraseMemoryFromRAM(
        FLASHCTL, TARGET_ADDRESS, DL_FLASHCTL_COMMAND_SIZE_SECTOR);

        if (gCmdStatus == DL_FLASHCTL_COMMAND_STATUS_FAILED) {
        /* If command execution fails, set error flag */
        gErrorCode = ERROR_ERASE;
    }

    if (gErrorCode == NO_ERROR) {
        /* Perform 8-bit write to main memory flash */
        DL_FlashCTL_unprotectSector(
            FLASHCTL, TARGET_ADDRESS, DL_FLASHCTL_REGION_SELECT_MAIN);

        gCmdStatus = DL_FlashCTL_programMemoryFromRAM8WithECCGenerated(
            FLASHCTL, TARGET_ADDRESS, &gData8);
        if (gCmdStatus == DL_FLASHCTL_COMMAND_STATUS_FAILED) {
            /* If command execution fails, set error flag */
            gErrorCode = ERROR_8BIT_W;
        }
    }
```

VI. Experiment Phenomenon

After the program is flashed, write one byte, two bytes, three bytes, and four bytes of data to the specified position sequentially. After writing is completed, the MCU indicator light will flash once every 1 second. We can view the written values through debugging tools



```
   154:          __BKPT(0);
0x00000702 B001      ADD      sp,sp,#0x04
0x00000704 BDF0      POP      {r4-r7,pc}
0x00000706 4D0B      LDR      r5,[pc,#44]  ; @0x00000734
0x00000708 4C0B      LDR      r4,[pc,#44]  ; @0x00000738
```

```
        * 11 FF FF FF
        * FF FF FF FF
        * 22 22 FF FF
        * FF FF FF FF
        * 33 33 33 33
        * FF FF FF FF
        * 78 56 34 12
        * EF CD AB 00/

        * 在断点处检查TARGET_ADDRESS的内容:
        * 11 FF FF FF
        * FF FF FF FF
        * 22 22 FF FF
        * FF FF FF FF
        * 33 33 33 33
        * FF FF FF FF
        * 78 56 34 12
        * EF CD AB 00*/
        while (1) {
            DL_GPIO_togglePins(LED_PORT,
                LED_LED_MCU_PIN);
            delay_cycles(16000000);
        }
    }

    else {

        __BKPT(0);
    }
}
```

```
0x00001000: 11 FF FF FF FF FF FF FF 22 22 FF FF FF FF FF FF 33 33 33
0x00001013: 33 FF FF FF FF 78 56 34 12 EF CD AB 00 FF FF FF FF FF FF
0x00001026: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001039: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000104C: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000105F: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001072: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001085: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001098: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000010AB: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000010BE: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000010D1: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000010E4: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000010F7: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000110A: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000111D: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001130: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001143: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001156: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001169: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000117C: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000118F: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000011A2: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000011B5: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000011C8: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000011DB: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000011EE: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001201: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001214: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001227: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000123A: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000124D: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001260: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001273: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001286: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x00001299: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000012AC: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000012BF: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000012D2: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000012E5: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x000012F8: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000130B: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0x0000131E: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```