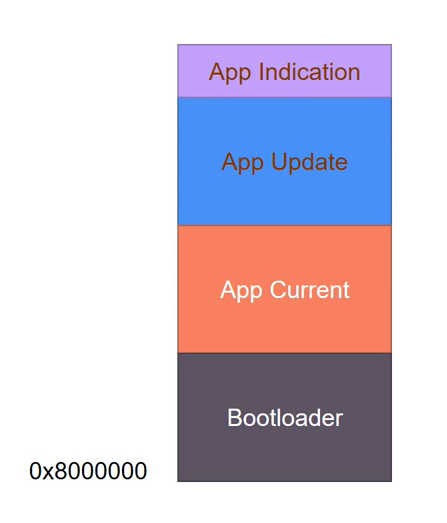
**Quá trình Update firmware:**

Khi đã tung một sản phẩm thiết bị ra thị trường, quá trình thực hiện nâng cấp phần mềm cho các thiết bị được mô tả một cách khái quát cụ thể sẽ có 4 bước sau đây:

* Bước 1: Sau khi cung cấp, lắp đặt thiết bị cho khách hàng thì công ty sản xuất phải thu thập những ý kiến đóng về thiếu sót, lỗi thường gặp khi triển khai ra thực tế. Sau đó đội ngũ kỹ sư lập trình sẽ sửa chữa và tạo một Firmware mới cho thiết bị để khắc phục những lỗi trên.
* Bước 2: Phát triển và tạo ra tệp tin chương trình mới. Đưa tệp tin này lên Server và đặt tên cho nó bằng một Version mới.
* Bước 3: Người dùng có thể tự tay điều khiển thiết bị đi vào quá trình cập nhật từ xa. Sau khi đã kiểm tra version của Firmware mới và thiết bị sẽ cho tải tệp tin chứa Firmware về.
* Bước 4: Khi quá trình tải Firmware đã hoàn tất thì thiết bị sẽ tự động kiểm tra và nạp chương trình vào bộ nhớ. Sau đó khởi chạy chương trình mới. Kết thúc quá trình cập nhật!

Đương nhiên việc cập nhật này không thể thực hiện bằng cách yêu cầu người dùng cắm các thiết bị hỗ trợ nạp Firmware như Stlink đối với STM32. Các nhà sản xuất sẽ ưu tiên sử dụng thêm 1 thiết bị để nhận file cập nhật từ trang Web, Cloud hoặc các kết nối khác như Wifi, bluetooth. Sau đó Firmware mới có thể dễ dàng được chuyển vào Flash để cho thiết bị hoạt động.

Khi này, dữ liệu của Firmware mới sẽ được lưu vào trong 1 vùng nhớ riêng. Ví dụ như hình:

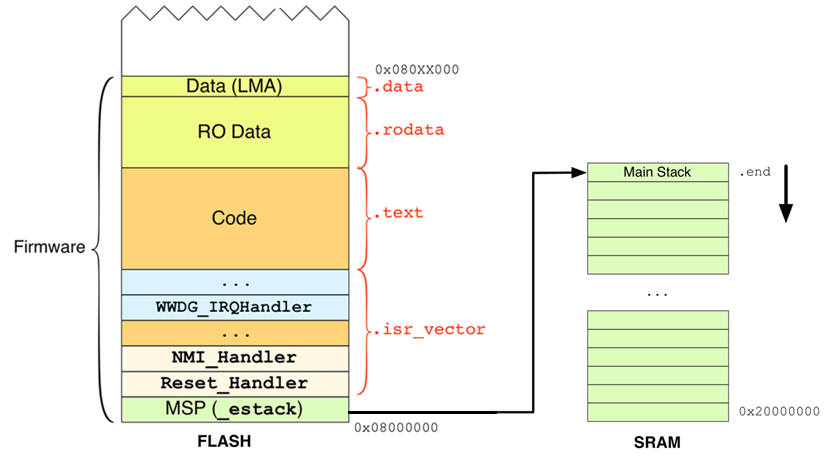


| App Indication | Vùng Flash chứa các cờ chỉ thị. |
| --- | --- |
| App Update | App cập nhật nhận được từ các thiết bị khác. |
| App Current | App chạy hiện tại trên Stm32 |
| BootLoader | Chương trình đầu tiên được chạy |

Khi này, sau khi reset MCU sẽ chọn 1 trong các chương trình được lưu ở vùng nhớ và thực thi. Tuy nhiên trước đó, Bootloader luôn luôn được nạp sẵn và được chạy trước. Đây là chương trình có chức năng quản lý và chỉ định MCU sẽ thực thi nhưng thiết lập hệ thống bán đầu.

Ví dụ: Bootloader kiểm tra vùng App Indication để xác nhận có file firmware Update trong vùng App Update hay không, từ đó quyết định sẽ thực thi Firmware nào, cùng như tiến hành cập nhật lại App Curent cho Chip.

**Quá trình Reset của Vi điều khiển:**

****

Địa chỉ bắt đầu của bộ nhớ (đối với Vi điều khiển STM32 là bộ nhớ FLASH) là **0x0000.0000**. Phần đầu tiên trong bộ nhớ của 1 chương trình đó là ***"Vector Table"*** - Chứa các vector ngắt, điển hình nhất chính là Vector Reset. Quá trình Reset gồm các bước sau:

1. CPU sẽ tìm kiếm bảng Vector Table của chương trình trên bộ nhớ .
2. Bộ xử lý tìm nạp giá trị tại ô nhớ đầu tiên của Vector Table, vào thanh ghi **MSP - Main Stack Pointer** (MSP ở trạng thái reset). Mặc định đối với STM32F1 là 0x08000000.
3. Tiếp theo, bộ xử lý sẽ lấy địa chỉ của **Reset Handler** được đặt trong ô nhớ thứ 2 của Vector Table load vào **PC** để thực hiện hàm **Reset Handler**.
4. Sau khi nhảy đến Reset Handler, bộ xử lý bắt đầu thực hiện các lệnh trong Reset Handler. Reset Handler thường được viết bằng ngôn ngữ Assembly (có thể là C), và đảm nhận công việc khởi tạo ban đầu của hệ thống.
5. Bước cuối cùng sau của Reset Handler là gọi đến hàm **main()** cũng chính là hàm main() trong file main.c. Và sau đó thực hiện các tác vụ trong hàm main().

Trong thực tế, 1 thiết bị có thể lưu được nhiều chương trình khác nhau trên các vùng khác nhau của bộ nhớ. Quá trình Reset lúc này sẽ chọn 1 trong các chương trình để thực hiện. Quá trình thực hiện 1 chương trình tương tự 5 bước trên.

## **Chương trình Bootloader**

Bootloader là một chương trình và là chương trình đầu tiên chạy khi Chip hoạt động.

Nhiệm vụ của Bootloader là kiểm tra các điều kiện để lựa chọn thực thi một trong các chương trình: Firmware update mới nhất, Firmware được nạp vào do hãng cung cấp trước khi xuất ra thị trường hoặc Current Firmware bản Firmware hiện tại đang chạy mỗi khi CPU reset.

Bootloader có mục tiêu chính là nâng cấp hoặc sửa đổi phần mềm hệ thống mà không cần sự can thiệp của các công cụ nâng cấp chương trình cơ sở chuyên dụng. Ví dụ như cài đặt phần mềm từ xa, nâng cấp phần mềm mà không cần kết nối các thiết bị nạp.

Bootloader có thể có nhiều chức năng, nhưng nó chủ yếu được sử dụng để quản lý ứng dụng. Nó cũng có thể sử dụng các giao thức khác nhau như UART, CAN, I2C, I2S, Ethernet hoặc USB để thiết lập giao tiếp và bắt đầu nâng cấp firmware.

Vị trí của chương trình Bootloader thường được bắt đầu tại địa chỉ đầu tiên của bộ nhớ Flash, đây là địa chỉ mặc định sẽ được CPU thực thi sau khi Reset. Với dòng vi điều khiển STM32 thì vị trí bắt đầu của bộ nhớ Flash là 0x08000000.

**Nhìn chung, hoạt động của CPU được mô tả như sau:**

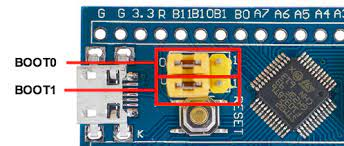
* Bắt đầu, hoặc mỗi khi CPU reset, PC nhảy đến địa chỉ bắt đầu (với boot 0-0 là thực thi vùng nhớ tại Flash, PC-> 0x08000000) , đây là vị trí lưu chương trình bootloader. vì vậy chương trình này sẽ được chạy đầu tiên.
* Lấy địa chỉ của App Current được lưu trên Flash.
* Hàm boot nhảy đến vị trí lưu App Current.
* Thực thi Firmware tại vị trí nhảy đến.
* Lặp lại mỗi khi khởi động hoặc có Reset.

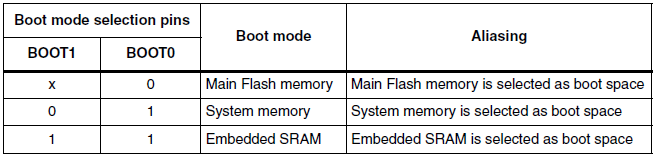
**Hoạt động của hàm Boot:**

* Lấy địa chỉ để chương trình nhảy đến.
* Xóa hết các ngắt, các trình xử lý lỗi.
* Set Main Stack Pointer (Địa chỉ bắt đầu của Firmware) đến địa chỉ mới.
* Set PC nhảy đến hàm Reset của Firmware mới.
* Gọi ngắt Reset để thực thi Firmware mới.

**Quá trình boot trên STM32:**

STM32F1 cung cấp cho người dùng chọn thực thi chương trình được lưu từ các vùng nhớ khác nhau. Việc này có thể thực hiện bằng cách thay đổi kết nối 2 Jumper trên Kit theo bảng bên dưới:



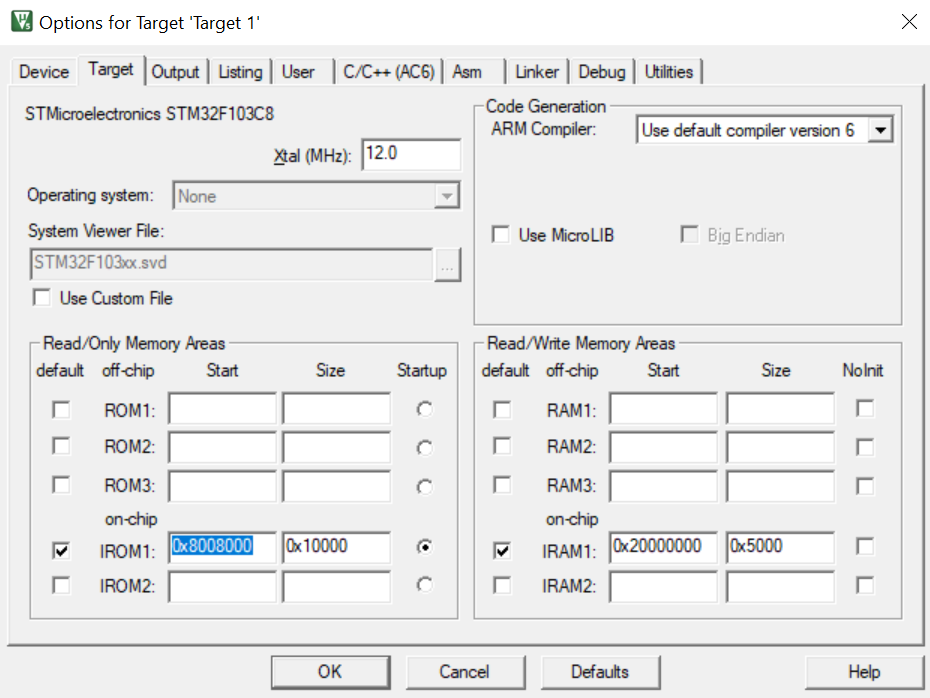


Thông thường, 2 jumper sẽ được đặt ở 0-0, tức là thực thi chương trình lưu ở vùng nhớ Flash. Sau khi khởi động, vi điều khiển sẽ tìm đến vùng nhớ Flash và boot chương trình được lưu và thực hiện nó.

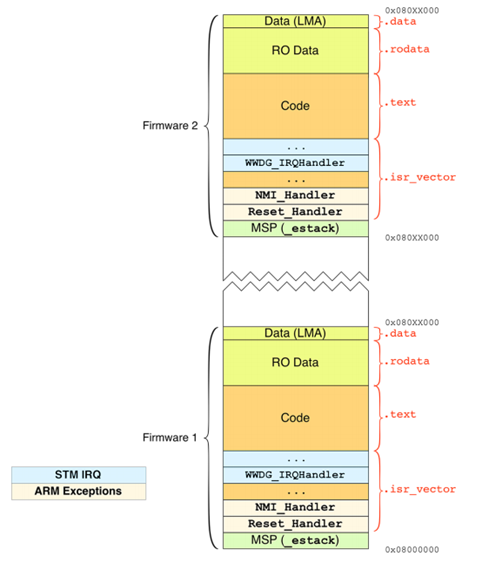
Ở bài này, mình sẽ minh họa quá trình boot bằng cách viết 1 App Blinkled và lưu nó ở 1 vùng nhớ khác 0x08000000 (cụ thể là 0x08008000)trên Flash, sau đó mình sẽ viết thêm 1 chương trình boot nạp vào tại 0x08000000 để điều khiển nhảy đến vùng nhớ chưa App Blinkled và thực thi.

Đầu tiên, App Blinkled mình sẽ lưu ở địa chỉ 0x08008000 trong Flash, có nghĩa là chia cho chương trình bootloader vùng nhớ từ 0x08000000 đến 0x08008000, kích thước 32Kb. Còn vùng Flash từ 0x08008000 trở đi sẽ dành để lưu chương trình App Blinkled.

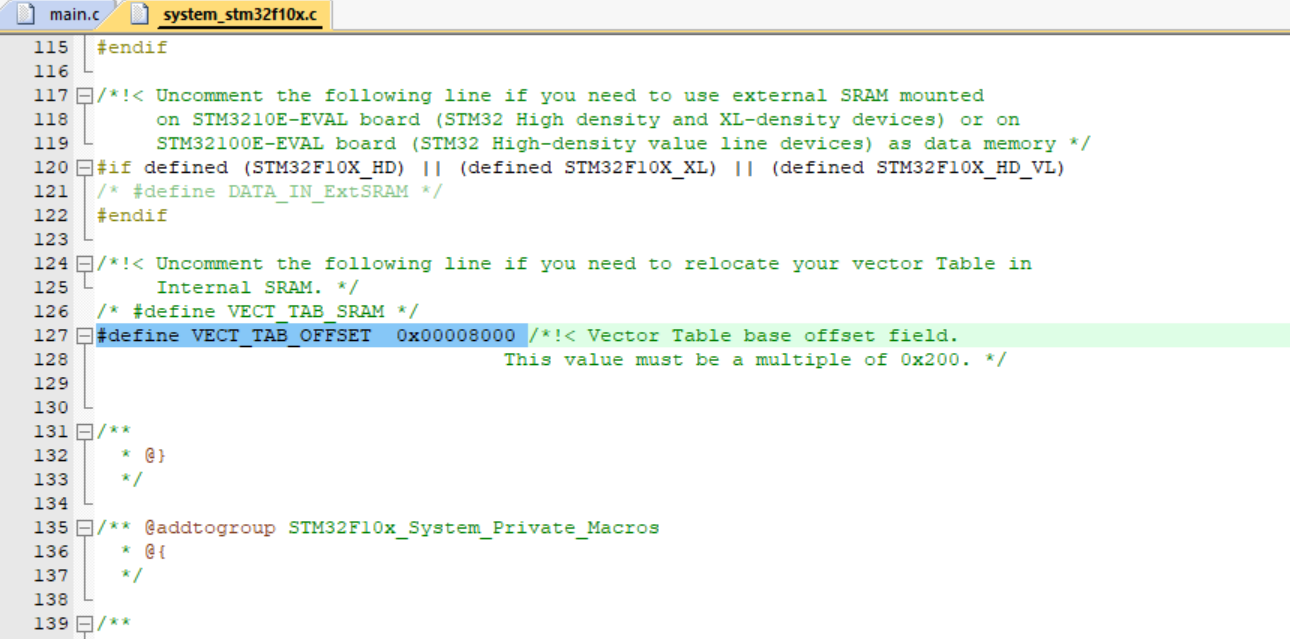
Khi đó, ta phải thay đổi giá trị địa chỉ để báo cho STlink nạp xuống vùng nhớ tương ứng.



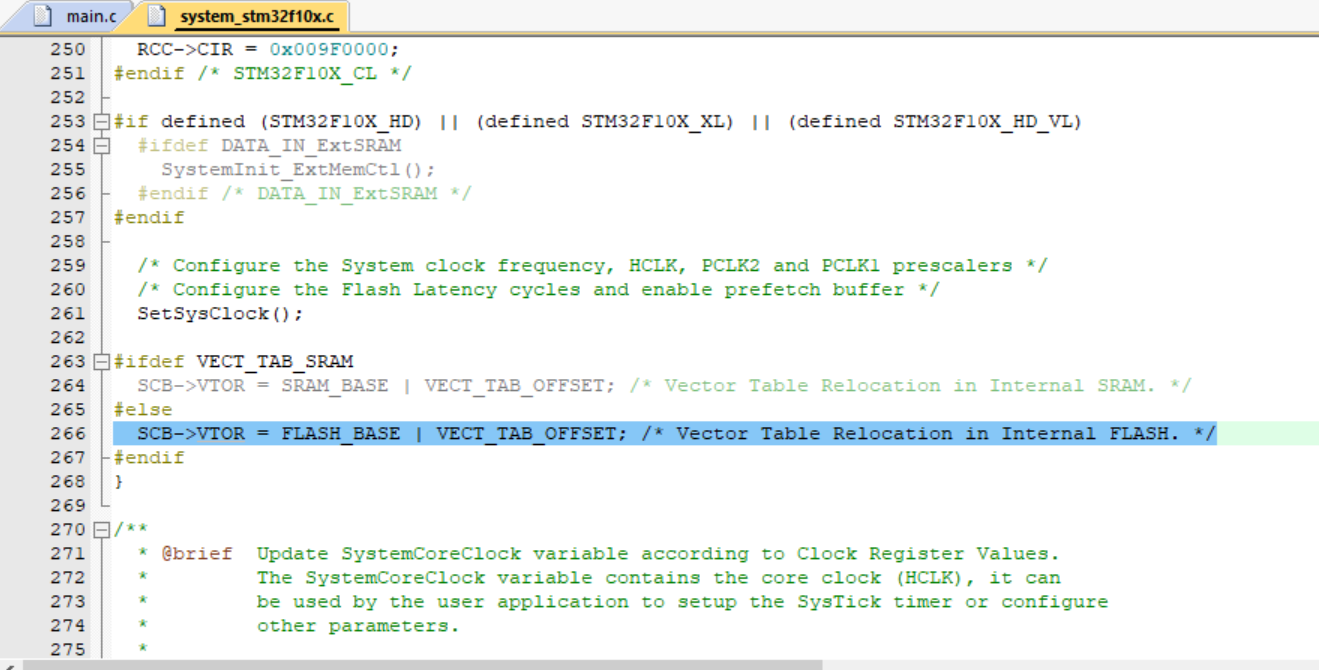
Một chương trình Firmware sẽ bao gồm cả phần code(chương trình thực thi), các vùng nhớ đi kèm và 1 bảng vector tương ứng. vì vậy ta phải dời bảng vector đúng với địa chỉ firmware được nạp:x



Ở đây ta dời bảng vector lên ứng với MSP ở 0x08008000. Sửa đổi giá trị Offset của table này trong file system.c



Khi đó bảng vector sẽ được dời bởi phép or, có nghĩa là or giữ 0x08000000 và 8000, kết quả sẽ là 0x08008000.



Tiến hành Build và nạp code như bình thường. Ở đây trước khi nạp, có thể xóa Flash để tránh ghi đè data bằng cách chọn Flash->Erase.

Như vậy là ta đã nạp thành công 1 APP vào Flash cho STM32.

Vì MCU sẽ mặc định thực hiện chương trình bắt đầu tại địa chỉ 0x08000000 trong Flash, đây là vùng ta sẽ lưu chương trình Bootloader, tức là khi khởi động hoặc Reset, mặc định chương trình bootloader được lưu tại đây sẽ chạy đầu tiên.

Chương trình bootloader sẽ được viết theo sơ đồ đã nêu ở phần trước:

int main(){

RCC\_DeInit();

SCB->SHCSR &= ~(SCB\_SHCSR\_USGFAULTENA\_Msk| SCB\_SHCSR\_BUSFAULTENA\_Msk| SCB\_SHCSR\_MEMFAULTENA\_Msk);

\_\_set\_MSP(\*(\_\_IO uint32\_t\*)(ADDR\_STR\_BLINK));

uint32\_t JumpAddress = \*(\_\_IO uint32\_t\*)(ADDR\_STR\_BLINK+4);

void (\*reset\_handler)(void) = (void \*)JumpAddress;

reset\_handler();

while(1){

}

}

* Chương trình sẽ chạy tuần tự vào trong main, hàm RCC\_Deinit(); reset bộ RCC về trạng thái reset mặc định.
* Vô hiệu hóa các bit kiểm tra lỗi trong thanh ghi SCB->SHCSR.
* Gọi hàm \_\_set\_MSP() để điều khiển Main Stack Pointer đến địa chỉ Firmware Blinkled đã nạp trước đó.
* Đặt địa chỉ có PC là ô nhớ tiếp theo sau MSP, tức là PC nhảy đến hàm Reset\_Handler() của Firmware Blinkled.
* Cấu hình hàm reset\_handler(); để sau khi reset, PC sẽ nhảy đến vị trí PC đã chọn.
* Gọi hàm reset để di chuyển PC.

Như vậy, qua ví dụ này có thể mô tả được quá trình boot. Với các ứng dụng phức tạp hơn, chương trình boot có thể thực hiện nhiều chức năng khác.

Ví dụ khi cập nhật chương trình từ data nhận về bằng UART, chương trình có thể đảm nhiệm việc xử lý ghi data(chương trình mới ) vào Flash, sau đó mới tiến hành gọi hàm boot để thực thi chương trình mới cập nhật này.

