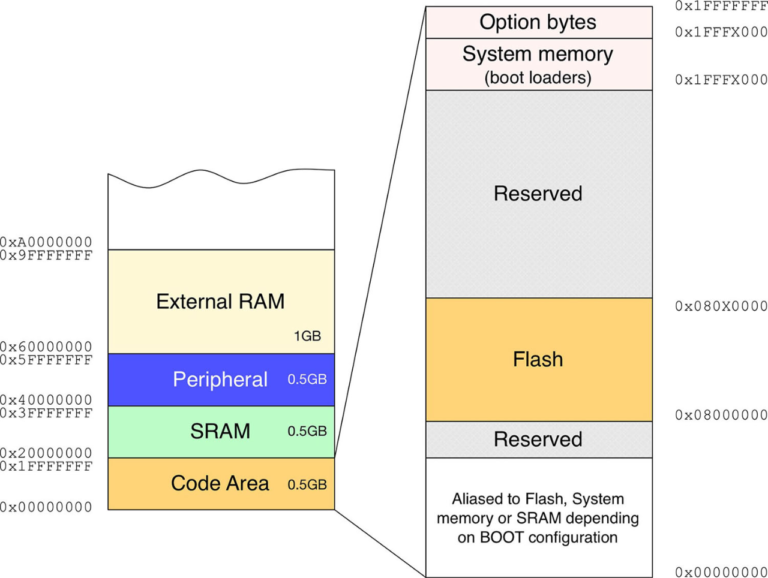
Hiểu được tổ chức bộ nhớ của vi điều khiển STM32 là một điều rất quan trọng khi tìm hiểu về Bootloader. Chúng ta cần phải nắm được chương trình Boot của chúng ta nằm ở địa chỉ nào, chương trình ứng dụng nằm ở địa chỉ nào, và cách phân chia các phần của [**bộ nhớ Flash**](https://www.laptrinhdientu.com/2022/01/EmbeddedMemory.html).

Bộ nhớ (tiếng Anh: memory) là nơi lưu trữ chương trình hoặc là nơi chứa các thông tin mà CPU đang làm việc.

Có 2 kiểu bộ nhớ cơ bản:

* [RAM](https://vi.wikipedia.org/wiki/RAM) (Random access memory) là bộ nhớ lưu các dữ liệu mà CPU đang làm việc. Dữ liệu trong RAM sẽ bị xóa khi mất điện.
* [ROM](https://vi.wikipedia.org/wiki/B%E1%BB%99_nh%E1%BB%9B_ch%E1%BB%89_%C4%91%E1%BB%8Dc)/EPROM/EEPROM hoặc flash: là bộ nhớ lưu trữ chương trình vận hành của vi điều khiển, chúng được ghi khi vi điều khiển được nạp chương trình. Nội dung trong các loại bộ nhớ này không bị mất khi mất điện hoặc reset.

| **RAM** | **FLASH** | **EPROM** |
| --- | --- | --- |
| * Tốc độ đọc/ghi nhanh. * Dữ liệu bị mất khi ngưng cấp nguồn. | * Tốc độ ghi chậm. * Tốc độ đọc nhanh. * Dữ liệu không bị mất khi ngưng cấp điện. * Phải xóa Flash trước khi tiến hành ghi lại dữ liệu. Không thể xóa từng byte mà phải xóa theo Page (1 vùng nhớ được phân chia kích thước rõ ràng). * Chỉ có thể đọc hoặc ghi theo khối 2/4 byte. | * Tương tự FLASH, tuy nhiên có thể đọc/ghi theo từng byte. |

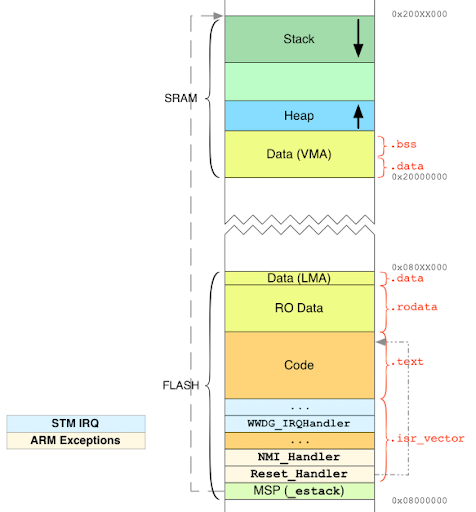


**Bộ nhớ chương trình:**

* 512MB đầu tiên được dành riêng cho vùng code.
* System Memory – Bộ nhớ hệ thống : Bộ nhớ hệ thống là vùng ROM dành riêng cho bộ nạp khởi động (Boot Loader). Mỗi họ STM32 cung cấp một số Boot Loader khác nhau được lập trình sẵn vào chip trong quá trình sản xuất. Những Boot Loader này có thể được sử dụng để tải code từ một số thiết bị ngoại vi, bao gồm USART, USB và CAN Bus.
* Option Bytes – Vùng tùy chọn byte : Vùng byte tùy chọn chứa một loạt cờ bit có thể được sử dụng để định cấu hình một số khía cạnh của MCU (như bảo vệ đọc ghi FLASH, v.v…

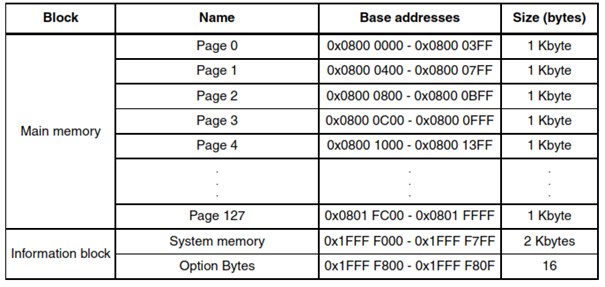
**Bộ nhớ FLash:**

Địa chỉ bộ nhớ Flash bắt đầu từ 0x0000.0000, nhưng trong Vi điều khiển STM32, vùng nhớ code được lưu từ địa chỉ 0x0800.0000, khi chúng ta nạp xuống, nó sẽ mặc định nạp chương trình từ địa chỉ này, với MSP - Main Stack Pointer ở địa chỉ 0x0800.0000 và Vector Table bắt đầu từ địa chỉ 0x0800.0004 (**Reset\_Handler**).



Bộ nhớ Flash được tổ chức gồm một khối chính (Main Block) có dung lượng lên đến 128kB (hoặc 64kB). V ìsố lần ghi/xóa ảnh hưởng tới tuổi thọ của Flash cũng như để dễ quản lý, bộ nhớ Flash trong STM32 được chia nhỏ thành các Page

Flash của STM32F1 được chia thành 128 Page (hoặc 64 Page đối với 64kB). Và một khối thông tin (information block). Có địa chỉ bắt đầu từ 0x0800 0000.



Bộ nhớ Flash có thể được thao tác **ghi trên từng Word** (4 bytes) hoặc **halfword** (2byte) , nhưng lại chỉ có thể **xóa theo từng Page**.

Vì bộ nhớ Flash còn được dùng để lưu trữ chương trình chính nên khi muốn lưu dữ liệu vào Flash, khuyến nghị là nên lưu vào những Page cuối (như Page 127 hoặc 64).

Để đọc dữ liệu từ Flash, đơn giản chỉ cần truy cập đến địa chỉ của vùng Flash và lấy dữ liệu cần thiết.

Quá trình ghi dữ liệu sẽ phức tạp hơn, vì Flash chỉ cho phép đoc/ ghi 2/4 byte và phải yêu cầu xóa trước khi ghi. Ngoài ra không thể xóa từng byte mà phải xóa cả page bộ nhớ tại đó.

Vì bộ nhớ Flash là nơi lưu chương trình, để tránh trường hợp vô tình làm thay đổi dữ liệu chương trình, Flash có 1 cơ chế đơn giản là LOCK và UNLOCK vùng nhớ.

Cụ thể Flash sẽ được đưa vào trạng trái LOCK sau khi reset để bảo vệ dữ liệu. Khi người dùng cần thao tác với vùng nhớ thì cần phải UNLOCK Flash để sử dụng.

Như đã đề cập ở trên, bộ nhớ FLash không thể cập nhật dữ liệu như RAM hay ROM, vì vậy muốn ghi dữ liệu thì trước hết phải xóa vùng nhớ Flash đó trước.

Thư viện SPL cung cấp thư viện "stm32f10x\_flash.h", có hỗ trợ sẵn các hàm phục vụ cho các thao tác trên Flash.

* **Các hàm LOCK và UNLOCK FLash**:
  + void FLASH\_Unlock(void): Unlock bộ điều khiển xóa Flash. Hàm này Unlock cho tất cả vùng nhớ trong Flash.
  + void FLASH\_UnlockBank1(void): Tương tự hàm trên, tuy nhiên hàm này chỉ Unlock cho Bank đầu tiên. Vì SMT32F103C8T6 chỉ có 1 Bank duy nhất nên chức năng tương tự hàm trên.
  + void FLASH\_UnlockBank2(void): Unlock cho Bank thứ 2.
  + \*Các hàm Unlock được gọi trước quá trình xóa hay ghi dữ liệu vào Flash để đảm bảo trạng thái của Flash là Unlock.
  + void FLASH\_Lock(void): Lock bộ điều khiển xóa Flash cho toàn bộ vùng nhớ Flash.
  + void FLASH\_LockBank1(void) và void FLASH\_LockBank2(void): Lock bộ điều khiển xóa Flash cho Bank 1 hoặc 2 tương ứng (STM32F103C8T6 chỉ có 1 Bank duy nhất).
  + \*Các hàm Lock thường được gọi sau khi thực hiện xóa/ghi dữ liệu vào Flash để đưa vùng nhớ vào trạng thái Lock.

Ví dụ trong hàm xóa Flash, Flash được Unlock bởi hàm FLASH\_Unlock();, sau khi xóa xong, hàm FLASH\_Lock(); đưa vùng nhớ trở về trạng thái Lock.

void Flash\_Erase(uint32\_t addresspage){

FLASH\_Unlock();

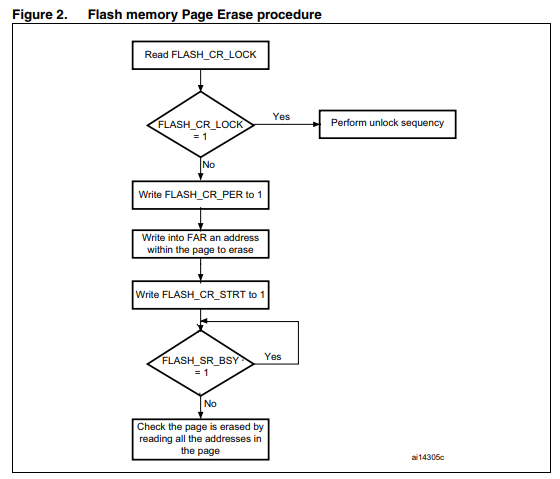
//Erase the Flash memory.

FLASH\_Lock();

}

* **Các hàm xóa Flash**: Flash chỉ có thể được xóa theo từng Page (ở STM32F103 là 1Kb mỗi Page) hoặc xóa theo cả Bank (STM32F103 chỉ có 1 Bank):
  + FLASH\_Status FLASH\_EraseAllBank1Pages(void): Xóa tất cả các Page trong Bank 1 của Flash.
  + FLASH\_Status FLASH\_EraseAllBank2Pages(void): Xóa tất cả các Page trong Bank 2 của Flash.
  + FLASH\_Status FLASH\_EraseAllPages(void): Xóa toàn bộ Flash.
  + FLASH\_Status FLASH\_ErasePage(uint32\_t Page\_Address): Xóa 1 page cụ thể trong Flash, cụ thể là Page bắt đầu bằng địa chỉ Page\_Address.

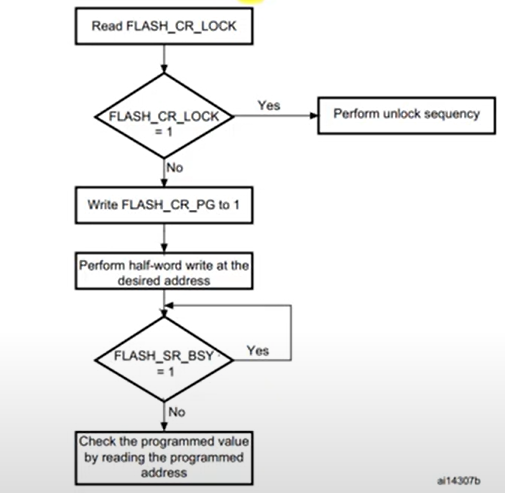
Sơ đồ quá trình xóa Flash mô tả như hình sau:



* Đầu tiên, kiểm tra cờ LOCK của Flash, nếu Cờ này đang được bật, Flash đang ở chế độ Lock và cần phải được Unlock trước khi sử dụng.
* Sau khi FLash đã Unlock, cờ CR\_PER được set lên 1.
* Địa chỉ của Page cần xóa được ghi vào FAR.
* Set bit CR\_STRT lên 1 để bắt đầu quá tình xóa.
* Kiểm tra cờ BSY đợi haonf tất quá trình xóa.

Sau khi được xóa, vùng nhớ Flash sẵn sàng cho việc ghi dữ liệu.

Sơ đồ quá trình ghi dữ liệu:



* Tương tự quá trình xóa, đầu tiên Cờ LOCK được kiểm tra.
* Sau khi xác nhận đã Unlock, CỜ CR\_PG được set lên 1.
* Quá trình ghi dữ liệu vào địa chỉ tương ứng sẽ được thực thi.
* Kiểm tra cờ BSY để đợi quá trình ghi hoàn tất.
* **FlagStatus FLASH\_GetFlagStatus(uint32\_t FLASH\_FLAG)**: hàm này trả về trạng thái của Flag. Ở bài này ta sẽ dùng hàm này để kiểm tra cờ FLASH\_FLAG\_BSY. Cờ này báo hiệu rằng Flash đang bận (Xóa/Ghi) nếu được set lên 1.

Khi đó, có thể viết hàm để xóa 1 Page Flash như sau:

void Flash\_Erase(uint32\_t addresspage){

FLASH\_Unlock();

FLASH\_ErasePage(addresspage);

while(FLASH\_GetFlagStatus(FLASH\_FLAG\_BSY) == 1);

FLASH\_Lock();

}

Với tham số uint32\_t addresspage là địa chỉ bắt đầu của Page cần xóa. Ví dụ muốn xóa page 0, tham số truyền vào sẽ là 0x08000000 (vùng nhớ bắt đầu page 0), page số 124, tham số truyền vào sẽ là 0x08000000 + 123\*1024 (vì mỗi page có kích thước 1Kb-1024 bytes).

Sau khi xóa xong, vùng nhớ tương ứng ở Flash sẽ sẵn sàng cho việc ghi data.

* **Các hàm ghi data vào Flash:** có thể ghi data vào flash với kích thước mỗi 2 hoặc 4 byte tương ứng với các hàm sau:
  + FLASH\_Status FLASH\_ProgramHalfWord(uint32\_t Address, uint16\_t Data): Ghi dữ liệu vào vùng nhớ Address với kích thước mỗi 2 byte (Halfword).
  + FLASH\_Status FLASH\_ProgramWord(uint32\_t Address, uint32\_t Data): Ghi dữ liệu vào vùng nhớ Address với kích thước mỗi 4 byte (Word).

Khi đó, hàm ghi data có thể triển khai như sau:

Hàm **WriteInt** ghi 1 giá trị kiểu int vào Flash, tương ứng với 2 byte data.

void Flash\_WriteInt(uint32\_t address, uint16\_t value){

FLASH\_Unlock();

FLASH\_ProgramHalfWord(address, value);

while(FLASH\_GetFlagStatus(FLASH\_FLAG\_BSY) == 1);

FLASH\_Lock();

}

Tương tự, hàm **WriteFloat** ghi 1 giá trị float vào Flash, tương ứng 4 byte data.

void Flash\_WriteFloat(uint32\_t address, float value){

FLASH\_Unlock();

uint8\_t data[4];

\*(float\*)data = value;

FLASH\_ProgramWord(address, \*(uint32\_t\*)data);

while(FLASH\_GetFlagStatus(FLASH\_FLAG\_BSY) == 1);

FLASH\_Lock();

}

Ở đây, giá trị float được chia thành 4 byte (uint8\_t data[4]), Sau đó được truyền đi bằng hàm FLASH\_ProgramWord.

**Flash\_WriteNumByte(uint32\_t address, uint8\_t \*data, int num):** Ghi vào Flash 1 chuỗi num byte có địa chỉ từ data vào địa chỉ address trong Flash.

void Flash\_WriteNumByte(uint32\_t address, uint8\_t \*data, int num){

FLASH\_Unlock();

uint16\_t \*ptr = (uint16\_t\*)data;

for(int i=0; i<((num+1)/2); i++){

FLASH\_ProgramHalfWord(address+2\*i, \*ptr);

ptr++;

}

FLASH\_Lock();

}

Để đọc data từ Flash, chỉ cần trả về nội dung tại địa chỉ cần đọc:

* Hàm **ReadInt** đọc về 1 giá trị int, tương ứng với 2 byte:

uint16\_t Flash\_ReadInt(uint32\_t address){

FLASH\_Unlock();

return \*(uint16\_t \*) (address);

FLASH\_Lock();

}

* Hàm **ReadFloat** đọc về 4 byte dưới dạng 1 giá trị float:

float Flash\_ReadFloat(uint32\_t address){

FLASH\_Unlock();

uint32\_t data;

data = \*(\_\_IO uint32\_t\*)(address);

return \*(float\*)(&data);

FLASH\_Lock();

}

* Hàm **ReadNumByte** đọc về 1 chuỗi các giá trị với kích thước 1 byte:

void Flash\_WriteNumByte(uint32\_t address, uint8\_t \*data, int num){

FLASH\_Unlock();

uint16\_t \*ptr = (uint16\_t\*)data;

for(int i=0; i<((num+1)/2); i++){

FLASH\_ProgramHalfWord(address+2\*i, \*ptr);

ptr++;

}

FLASH\_Lock();

}

Khi đó, quá trình Ghi bắt đầu với việc gọi hàm **Flash\_Erase** với tham số là vùng nhớ cụ thể cần xóa. Sau đó có thể tùy chọn ghi dữ liệu bằng các hàm **Write.** Việc đọc thì đơn giản hơn, chỉ cần lấy ra giá trị bằng các hàm **Read**.

Việc nắm rõ thao tác trên Flash giúp ta hiểu được cách ghi/đọc dữ liệu bộ nhớ chương trình.