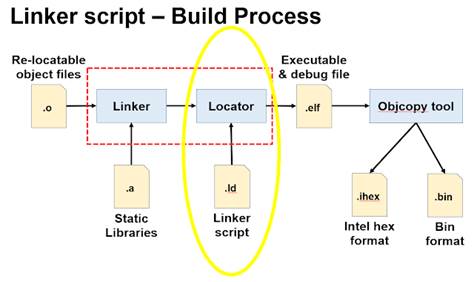
# Build Process

Build Process: Về cơ bản tương tự một chương trình C trên window tuy nhiên có một vài điểm khác biệt:

* Compiler không compile ra file ASM của máy tính nữa, mà ở đây là Cross Compiler.
* Các file Object vẫn được link với thư viện tĩnh, nhưng cần trải qua một bước nữa đó là Locator, bước này dùng 1 file Linker Script để phân bố vùng nhớ cho vi điều khiển.
* File đích tạo ra không còn là .exe mà là các file như .elf (dùng để execute & debug), các file định dạng khác như .hex / .bin để nạp code.

Một [Linker Script File](https://www.laptrinhdientu.com/2021/08/LinkerScriptFile.html) (.ld) là một file text trong quá trình locator nhằm phân vùng địa chỉ trên vi điều khiển, do các vi điều khiển có vùng địa chỉ tuyệt đối trên flash, ram (không giống như máy tính).



Dữ liệu trong Data Section được lưu trên cả Flash và Ram để có đặc tính Non - Volatile của Flash (dữ liệu không bị mất khi reset) và được copy sang RAM sau khi reset, đặc tính Ramdom Access Memory của RAM để thay đổi dữ liệu.

Khi microcontroller được boot thì nó sẽ chạy startup code bên trong flash memory trước khi vào hàm main.

Và startup code này sẽ khởi tạo các biến global => được lưu trữ trong cả flash memory và data section trong RAM.

Có hai loại bộ nhớ Flash và RAM:

* Flash: Non volatile, not ramdom và được sử dụng để lưu trữ code.
* RAM: volatile, ramdom access memory và được sử dung để lưu trữ data.

# Reset Sequence

[Startup file](https://www.laptrinhdientu.com/2021/08/StartupFile.html) được thực hiện trước hàm main, là file khởi tạo ban đầu trước khi call hàm main chạy và chứa vector table.

Startup file:

+ Tìm đến Reset\_Handler() ở trong vector table.

+ Khởi tạo section .data và .bss => copy section data và bss từ Flash lên RAM.

+ Khởi tạo standard library

+ Call đến hàm Main

Khi vi điều khiển reset nó sẽ tìm đến hàm Reset\_Handler() ở trong vector table (trong startup file) sau đó thực hiện một số công việc trong startup file.

[Main Stack Pointer - MSP:](https://www.laptrinhdientu.com/2021/09/Core5.html) chứa địa chỉ của bộ nhớ segment ở thời điểm hiện tại, tức là giá trị thanh ghi này sẽ thay đổi theo địa chỉ mà mình đang thao tác với bộ nhớ Stack.

Khi reset, CPU sẽ tìm đến địa chỉ của Vector Table (0x0000.0000), gán giá trị của địa chỉ này vào MSP để quy định vùng nhớ Stack, tiếp theo bộ xử lý lấy giá trị được đặt trong ô nhớ thứ 2 của Vector Table và gán vào PC để thực hiện hàm Reset\_ISR để thực hiện Startup code. Cuối cùng, nó sẽ gọi đến hàm main.

Vector table: là một vùng nhớ lưu trữ giá trị bắt đầu của Stack, địa chỉ bắt đầu của các hàm Handler.

# Cortex M0+

## Introduce

Bộ xử lý Cortex-M0+ là bộ xử lý ARM Cortex 32-bit cơ bản được thiết kế theo kiến trúc ARMv6-M cho các ứng dụng nhúng. Xây dựng theo kiến trúc von Neumann.

Ảnh có chứa văn bản, biểu đồ, Kế hoạch, sơ đồ

Mô tả được tạo tự động

## Processor mode

Thread mode: Khi khởi động, vi xử lý đều bắt đầu ở Thread mode, ở chế độ này, code của chương trình được thực hiện tuần tự trong hàm main(). Bất cứ khi nào xảy ra Ngắt hay Exception, vi xử lý sẽ chuyển sang chế độ Handler Mode. Ở chế độ này, thanh ghi SP được sử dụng có thể là MSP hoặc PSP (thường ứng dụng trong RTOS).

Handler mode: Bất cứ khi nào xảy ra Ngắt hay Exception, vi xử lý sẽ nhảy vào ISR (Interrupt Service Routine) tương ứng, tức là chuyển sang chế độ Handler Mode. Ở chế độ này, chỉ có MSP được phép sử dụng.

Unprivileged(Chế độ không đặc quyền):

* Quyền truy cập hạn chế vào các thanh ghi hệ thống bằng các instruction MSR và MRS, và không thể sử dụng lệnh CPS để ngắt có che.
* Không thể truy cập các timer SysTick, NVIC hoặc SCB.
* Hạn chế truy cập vào memory hoặc peripherals.

=> Nếu cố gắng truy cập các thanh ghi và vùng nhớ này sẽ gây ra Hardfault

Privileged(Chế độ đặc quyền): Có thể sử dụng tất cả các instruction và có thể truy cập vào bất kỳ các thanh ghi và vùng nhớ.

Trong chế độ Thread mode, thanh ghi CONTROL kiểm soát việc thực thi bằng phần mềm có Privileged hay không.

Trong chế độ Handler, việc thực thi bằng phần mềm luôn được Privileged.

Chỉ phần mềm được Privileged mới có thể ghi vào thanh ghi CONTROL để thay đổi Privilege Level cho việc thực thi phần mềm ở chế độ Thread.

Phần mềm Unprivileged có thể sử dụng instruction SVC để chuyển quyền kiểm soát sang Privileged.

## Stack

Bộ xử lý sử dụng Descending Stack, nghĩa là Stack Pointer chỉ item cuối cùng trong bộ nhớ ngăn xếp. Tức là khi bộ xử lý push một item mới vào stack, Stack Pointer giảm và sau đó ghi item vào vị trí mới.

Bộ xử lý thực hiện hai Stack là: Main Stack và Process Stack.

* Chế độ Thread: Main Stack và Process Stack.
* Chế độ Handler: Main Stack.

## Core Resister

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, hàng

Mô tả được tạo tự động

*R0-R12:* là các thanh ghi 32 bit General Purpose Register cho các hoạt động xử lý, tính toán dữ liệu.

*Stack Pointer - SP:* là thanh ghi R13. Ở chế độ Thread, bit[1] của thanh ghi Control quy định SP dùng để

* 0 = Main Stack Pointer – MSP: Giá trị reset
* 1 = Process Stack Pointer – PSP.

Khi reset, giá trị của MSP được gán với giá trị của địa chỉ 0x00000000.

*Link Register - LR:* là thanh ghi R14, chứa địa chỉ trả về của các lệnh rẽ nhánh như call function. Khi reset, giá trị thanh ghi LR là Unknown.

*Program Counter– PC:* là thanh ghi R15, chứa địa chỉ câu lệnh tiếp theo. Khi reset, giá trị của thanh ghi PC được gán với giá trị của Reset Vector, có địa chỉ 0x00000004. Bit[0] của giá trị này được gán vào bit T EPSR khi reset và phải là 1.

*Program Status Register – PSR:* gồm

* Application Program Status Register (APSR).
* Interrupt Program Status Register (IPSR).
* Execution Program Status Register (EPSR).

Được phân bổ dưới dạng các bitfield loại trữ lẫn nhau trong 32 bit PSR.

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, Phông chữ

Mô tả được tạo tự động

APSR chứa trạng thái hiện tại của các cờ điều kiện từ lệnh thực thi trước đó.

* N - Negative flag
* Z – Zero flag
* C – Carry or borrow flag
* V – Overflow flag

IPSR chứa exception number của ISR hiện tại.

* 0 = Thread mode.
* 1 = Reserved.
* 2 = NMI.
* 3 = HardFault.
* 4-10 = Reserved.
* 11 = SVCall.
* 12, 13 = Reserved.
* 14 = PendSV.
* 15 = SysTick, if implemented.
* 16 + n = IRQn.

EPSR chứa bit trạng thái Thumb, dùng để xác định tập lệnh mà bộ xử lý đang sử dụng.

* T = 1: Tập lệnh Thumb ISA
* T = 0: Tập lệnh Arm ISA

+ Thumb : Tập lệnh 16 bit do đó tiết kiệm code size.

+ ARM : Tập lệnh 32 bit do đó thực thi nhanh hơn Thumb.

+ Thumb2 (hỗn hợp 16/32 bit) : là sự trộn lẫn giữa ARM và thumb 16, để có được cả hiệu suất của ARM và mật độ lệnh của Thumb.

+ Sự khác biệt giữa Thumb và Arm:

* Khi xảy ra exception, sử dụng câu lệnh Arm
* Với Thumb, không có lệnh nào để truy cập thanh ghi trạng thái.
* [Với các câu lệnh rẽ nhánh, địa chỉ có LSB được set lên 1 khi ở trạng thái Thumb.](https://www.youtube.com/watch?v=WfPKQPYgoxg&list=PLGZ_RWetU5HKH963lD-LcTye5f2LV4YHZ&index=6)

*Thanh ghi CONTROL:* kiểm soát Stack được sử dụng và optional privilege level để thực thi phần mềm khi processor ở chế độ Thread Mode.

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình, Hình chữ nhật

Mô tả được tạo tự động

* SPSEL: Quy định Stack hiện tại:

0 = MSP là SP hiện tại.

1 = PSP là SP hiện tại.

* NPRIV: Xác định privilege level của chế độ Thread Mode: 0 = Privileged, 1 = Unprivileged. Nếu Unprivileged/Privileged không được cấu hình, bit này được Reserved.

# Memory

Thanh ghi Non – Memory Mapped, là những thanh ghi nằm trong core bộ xử lý và không có bất kỳ địa chỉ nào để truy cập chúng từ chương trình C => Muốn truy cập phải dùng tập lệnh Assembly.

Thanh ghi Memory Mapped – Thanh ghi ánh xạ bộ nhớ: là một phần của memory và là những thanh ghi có địa chỉ. Bằng cách sử dụng địa chỉ này ta có thể đọc và ghi dữ liệu vào thanh ghi này bằng chương trình C.

* Thanh ghi liên quan đến ngoại vi bộ xử lý: NVIC, MPU, SCB, DEBUG …
* Thanh ghi liên quan đến ngoại vi của vi điều khiển: GPIO, RTC, I2C, TIMER, CAN, …

Trong bộ xử lý, vùng nhớ có thể định địa chỉ phụ thuộc vào kích thước của bus địa chỉ.

Địa chỉ bắt đầu của memory thường là bộ nhớ Flash.

Bộ nhớ Flash được chia thành các sector để thao tác với bộ nhớ hiệu quả. Bộ nhớ Flash, ngoài lưu trữ MSP, Vector Table, bộ nhớ Flash sẽ lưu trữ vùng nhớ chương trình ứng dụng, cùng với đó là vùng data (bao gồm read-only data, và vùng nhớ .data).

Bộ nhớ Flash có thể được thao tác ghi trên từng Word (4 bytes), nhưng lại chỉ có thể xóa theo từng Sector. Vì vậy để tiết kiệm số lần ghi/xóa (ảnh hưởng tới tuổi thọ của Flash) cũng như để dễ quản lý, bộ nhớ Flash được chia nhỏ thành các sector, với những sector đầu có kích thước nhỏ, và những sector sau có kích thước lớn dần.

# Interrupt & Exception

Một bộ vi điều khiển cortex M hỗ trợ tới 256 ngắt, trong đó có 15 ngắt System Exceptions là các ngắt đến từ core.

CMSIS sử dụng giá trị âm cho các System Exceptions. Các ngắt còn lại là các ngắt ngoại vi (Interrupt Request - ISR) được bắt đầu bằng 0.

=> Dễ dàng phân loại System Exceptions và Peripheral Interrupt.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Khi một ngắt được phục vụ, Interrupt Number hoặc Current Exceptions được ghi vào thanh ghi Program Status Register (PSR).

Interrupt Number trong PSR khác với Interrupt Number trong hàm CMSIS:

Interrupt Number in PSR = 16 + Interrupt Number for CMSIS.

Kích hoạt System Exception khác với kích hoạt Peripheral Interrupt.

* Một số System Exception như reset và hard fault luôn được kích hoạt. Các System Exception có thể được bật hoặc tắt bởi module tương ứng của chúng chẳng hạn như Systick.
* Bật/tắt tất cả các Peripheral Interrupt được thực hiện bằng cách sửa đổi hại bộ thanh ghi: thanh ghi ISER và thanh ghi ICER.

Thanh ghi ISER được sử dụng để kích hoạt các Peripheral Interrupt. Ghi 1 vào một bit trong ISER để kích hoạt ngắt tương ứng. Tuy nhiên, ghi 0 vào một bit trong thanh ghi không vô hiệu hoá ngắt tương ứng.

Thanh ghi ICER được sử dụng để vô hiệu hoá các Peripheral Interrupt. Ghi 1 vào một bit trong ICER sẽ vô hiệu hoá ngắt tương ứng. Ghi 0 vào một bit trong ICER không có tác động.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Mô tả được tạo tự động

## Exception States

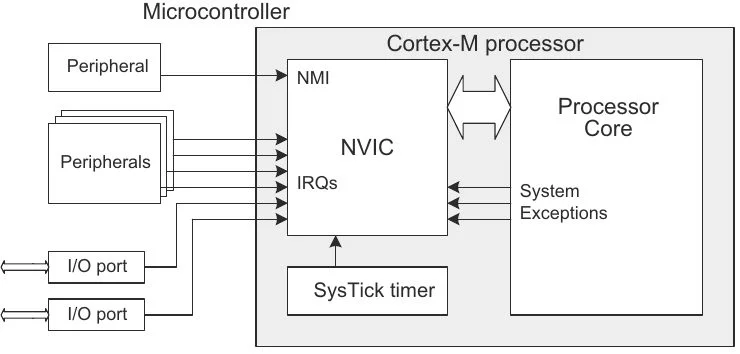
Pending: Exception đang chờ bộ xử lý phục vụ, yêu cầu ngắt từ thiết bị ngoại vi hoặc từ phần mềm có thể thay đổi trạng thái của ngắt tương ứng để chờ xử lý.

Active: Exception đang được bộ xử lý phục vụ nhưng chưa hoàn thành.

Inactive: Exception không active và không pending.

## NVIC

NVIC - Nested Vector Interrupt Controller, là bộ điều khiển vector ngắt lồng nhau, nó là một ngoại vi của Core Cortex Mx.



Khi NVIC nhận được yêu cầu ngắt, nó buộc vi xử lý nhảy tới và thực thi một đoạn mã đặc biệt, được gọi là trình xử lý ngắt - ISR.

Các điểm vào của tất cả các chương trình ngắt, được lưu trữ trong một bảng đặc biệt gọi là Vector Table.

Vector Table được lưu trước tại một khu vực xác định trong bộ nhớ và được ánh xạ tới địa chỉ thấp nhất của bộ nhớ flash. Tuy nhiên, có thể ánh xạ lại nó tới bộ nhớ SRAM.

Vector Table chứa một mảng các địa chỉ bộ nhớ hay một mảng con trỏ hàm. Mỗi loại ngắt được gán một số gọi là Number Interrupt và được sử dụng để lập chỉ mục vào bảng Vector Table. Mỗi chỉ mục này dài 4 byte và chứa địa chỉ bắt đầu của một trình xử lý ngắt - ISR.

Ví dụ: Khi ngắt x được kích hoạt, NVIC sử dụng Number Interrupt x, làm giá trị chỉ mục, để tra cứu bảng vector ngắt, tìm địa chỉ bộ nhớ của trình xử lý ngắt - ISR tương ứng cho ngắt x và buộc vi xử lý nhảy tới và thực hiện ngắt này.

## External Interrupt Controller

External Interrupt Controller giám sát sự thay đổi của tín hiệu điện áp => Yêu cầu ngắt => Gửi đến NVIC

Hỗ trợ 16 External Interrupt kết hợp với GPIO. Mỗi ngắt này được liên kết với một chân GPIO cụ thể.

Bộ điều khiển có một bộ ghép kênh - MUX cho mỗi External Interrupt này.

*Enable ngoại vi, Enable NVIC, Enable Core => interrupt mới được thực hiện.*

Một bộ xử lý có nhiều hơn 16 chân GPIO => Làm thế nào để bộ điều khiển mask các chân GPIO tới External Interrupt

=> Với STM32, chỉ các chân có số pin x mới có thể là nguồn của External Interrupt x.

Ví dụ: Nguồn của External Interrupt 3 có thể là PA3, PB3, PC3, PD3, ...

Thanh ghi External Interrupt Configuration Resister(SYSCFG\_EXTICR) chỉ định chân nào được chọn làm nguồn External Interrupt.

Không có nhiều chân có thể được sử dụng đồng thời làm nguồn External Interrupt.

Ví dụ: Nếu chân PA3 có External Interrupt thì không thể sử dụng chân PB3 làm nguồn cho External Interrupt 3.

## Priority

Mức độ ưu tiên - Priority được coi là mức độ khẩn cấp của ngắt, tức là quy định ngắt nào cần được thực hiện trước.

Giá trị mức độ ưu tiên – priority value là thước đo mức độ khẩn cấp của ngắt, còn gọi là mức độ ưu tiên – priority levels.

Với Cortex M, giá trị ưu tiên này càng nhỏ thì mức độ ưu tiên càng lớn. Chẳng hạn, các Exception System đa số có mức ưu tiên nhỏ hơn 0 và được cố định (trong đó Reset Handler có priority value nhỏ nhất, tương đương với mức độ ưu tiên lớn nhất).

Nếu hai ngắt xảy ra cùng lúc thì NVIC sẽ tiếp nhận thực thi ngắt có mức độ ưu tiên cao hơn (priority value thấp hơn), và đưa ngắt còn lại vào trạng thái chờ - Pending.

Pre-Empt Priority: Khi bộ xử lý đang chạy một trình xử lý ngắt và một ngắt khác xảy ra, thì các giá trị ưu tiên trước – preempt priority sẽ được so sánh và ngắt với mức độ preempt priority cao hơn (giá trị nhỏ hơn) sẽ được cho phép chạy.

Sub Priority: Giá trị này chỉ được sử dụng khi hai ngắt có cùng giá trị pre-empty priority xảy ra cùng thời điểm. Trong trường hợp này, ngắt có mức ưu tiên phụ cao hơn sẽ được xử lý trước.

# Systick Timer

Systick Timer là một timer, là một timer độc lập nằm trong lõi Cortex.

SysTick Timer là một Timer đếm ngược 24 bit. Xảy ra ngắt SysTick Interrupt khi thanh ghi counter value về 0, và tự nạp lại giá trị được lưu trong reload value register.

SysTick Timer bào gồm:

* Một bộ đếm Counter để đếm xuống.
* Một thanh ghi chứa giá trị nạp lại.
* Một cờ báo tràn Timer.
* Một bộ phận để cấu hình xung clock đầu vào.

Ảnh có chứa văn bản, biểu đồ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

System Timer hay SysTick được điều khiển bởi 4 thanh ghi bao gồm:

* Thanh ghi SysTick Control and Status Register(SYST\_CSR)
* Thanh ghi SysTick Reload Value (SYST\_RVR)
* Thanh ghi SysTick Current Value (SYST\_CVR)
* Thanh ghi SysTick Calibration Value(SYST\_CALIB)

Trong thanh ghi SysTick Control and Status Register(SYST\_CSR): Chỉ có 4 bit được sử dụng bao gồm 1 bit status và 3 bit control. Bit status là COUNTFLAG, Bit Control bao gồm: bit select the systick timer clock CLKSOURCE, bit cho phép ngắt TICKINT, bit cho phép counter.

CLKSOURCE = 1 => Chọn processor clock

CLKSOURCE = 0 => Chọn external clock

Khi thanh ghi Counter xuống 0 => COUNTFLAG đặt thành 1.

Khoảng thời gian giữa hai lần ngắt SysTick là:

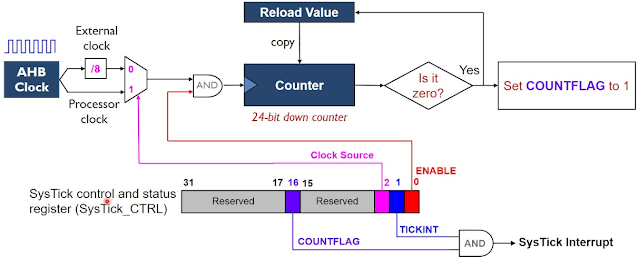
Interval = (RELOAD + 1) \* Source\_Clock\_Period.

Ví dụ: Nếu giữa hai lần ngắt SysTick là 100 chu kỳ Clock thì reload value sẽ là 99.

Ghi giá trị vào thanh ghi Current Value, xoá thanh ghi này và COUNTFLAG xuống 0 => System Timer reload từ thanh ghi reload value trên clock tiếp theo.

Thanh ghi Counter có một giá trị ngẫu nhiên khi reset => Phải luôn xóa về 0 trước khi enable.

Thanh ghi SysTick Calibration Value(SYST\_CALIB) là thanh ghi only – read.

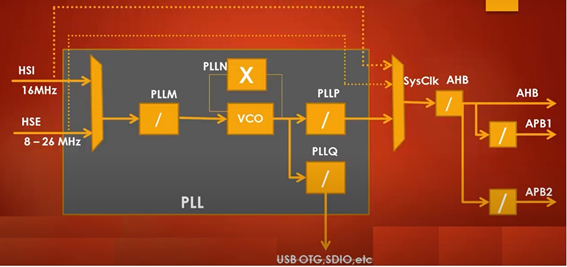


# Clock

Dao động càng nhanh thì khả năng tính toán và xử lý càng tốt, tuy nhiên, nó sẽ tốn điện hơn => với các dự án đòi hỏi việc tiết kiệm năng lượng, đòi hỏi phải cố gắng giảm tần số hoạt động của con chip xuống tối thiểu.

Các vi điều khiển 32 bit cần phải config clock trước khi sử dụng các ngoại vi. Mặc định, tất cả clock của các thiết bị ngoại vi sẽ bị vô hiệu hóa để tiết kiệm năng lượng.

Đối với VĐK STM32, clock được cấu hình qua thanh ghi RCC.



The system clock (SYSCLK):

* HSI – High Speed Internal: Lấy từ bộ giao động RC nội, mặc định là 16 MHz.
* HSE – High Speed External: Lấy từ dao động thạch anh
* PLL – Phase Lock Loop: bộ nhân tần

2 nguồn clock:

* 40kHz low speed internal RC ( LSI RC) : thực hiện RTC hoặc watchdog để wakeup từ chế độ stop/ standby.
* 32.768kHz low speed external crystal (LSE crystal): real-time clock (RTCCLK).

Các ngoại vi được cấp xung clock thông qua ba đường bus sau:

* AHB (Advanced High Speed Buses ): Đây là Bus kết nối hệ thống.
* APB1, APB2 (Advanced Peripheral Buses 1,2): Đây là các Bus kết nối với thiết bị ngoại vi và kết nối với hệ thống thông qua AHB.