TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMUTE
HCMC University of Technology and Education

TÌM HIỂU SYSTICK VÀ PHƯƠNG PHÁP THAY ĐỔI TỐC ĐỘ TRÊN VI ĐIỀU KHIỂN

GVHD: Huỳnh Hoàng Hà

Embedded System

Nhóm sinh viên thực hiện: 16

TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMUTE
HCMC University of Technology and Education

HỆ THỐNG GIÁM SÁT VÀ CUNG CẤP THỨC ĂN TỰ ĐỘNG

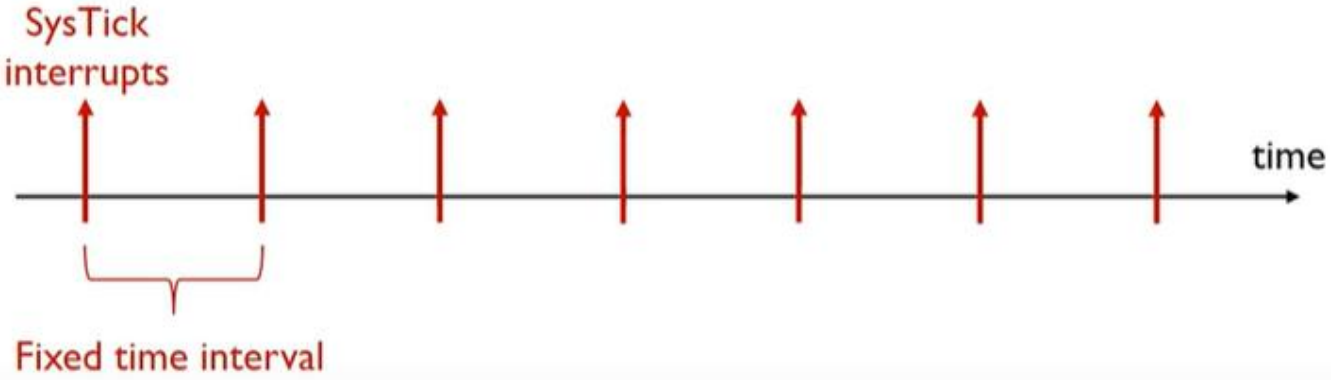
GVHD: Ths.Huỳnh Hoàng Hà

Embedded System

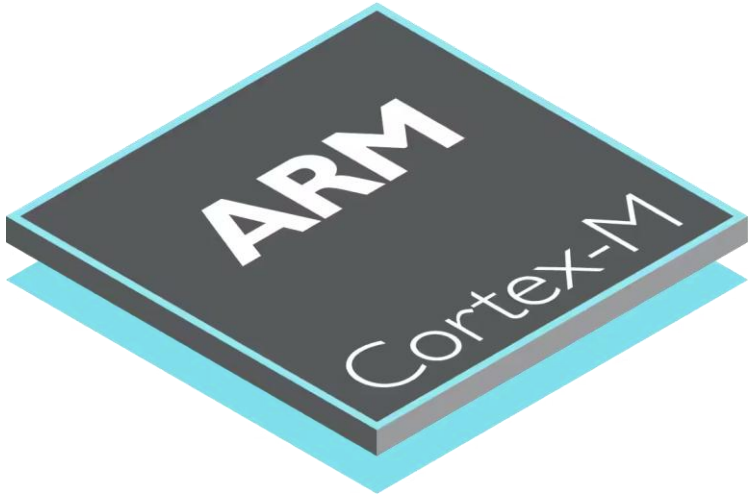
Nhóm sinh viên thực hiện: 16

Tóm tắt nội dung

- **SysTick Timer**: Bộ đếm thời gian 24-bit trong ARM Cortex-M tạo ngắt định kỳ.

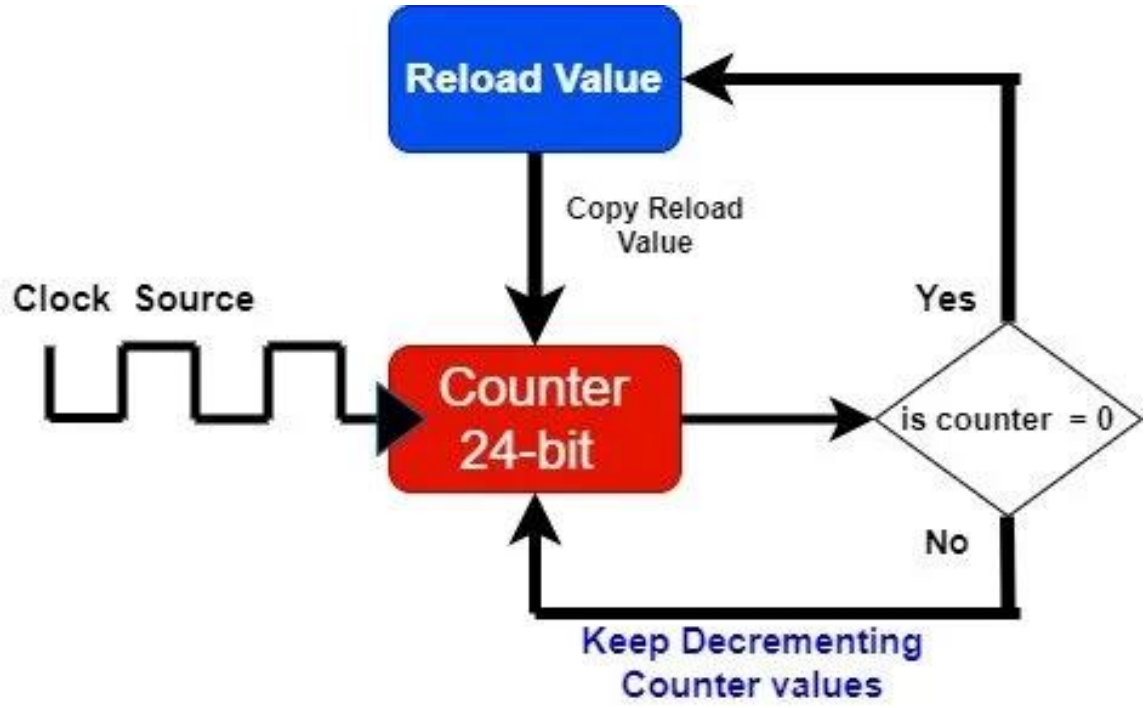


- **Chức năng**:
  - Tạo delay chính xác.
  - Cung cấp ngắt cho RTOS.
  - Thực hiện tác vụ định kỳ.



Tóm tắt nội dung

- **Cơ chế hoạt động**:
  - Chọn nguồn clock (CPU/HCLK hoặc HCLK/8).
  - Thanh ghi chính: **CTRL, LOAD, VAL, CALIB**.
  - Counter về 0 → tạo ngắt → nạp lại giá trị.

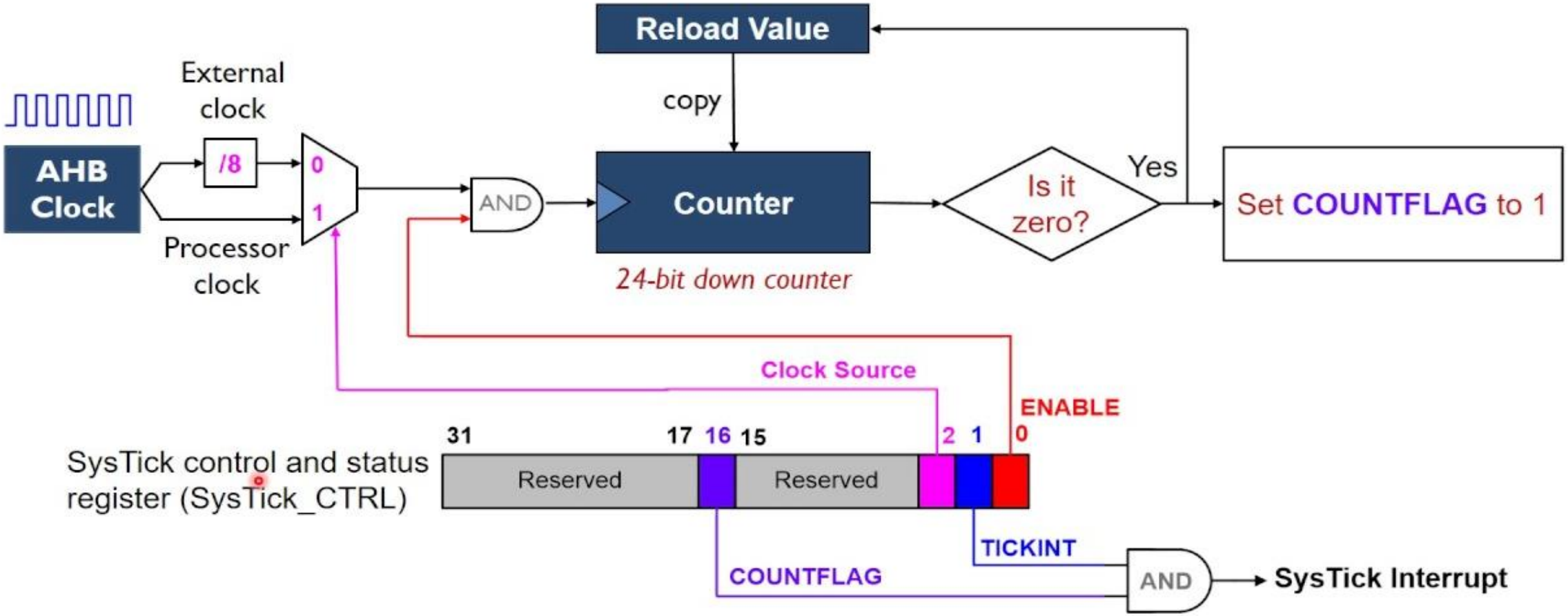


Tóm tắt nội dung

- **Thay đổi tốc độ**:
  - Thay đổi **RELOAD** hoặc nguồn clock.
  - Công thức: RDelay, đo thời gian, lập lịch đa nhiệm, demo code.
  - ELOAD = (Clock / f) – 1.

- **Ứng dụng**: Delay, đo thời gian, lập lịch đa nhiệm, demo code.

1.1. Khái quát về SysTick

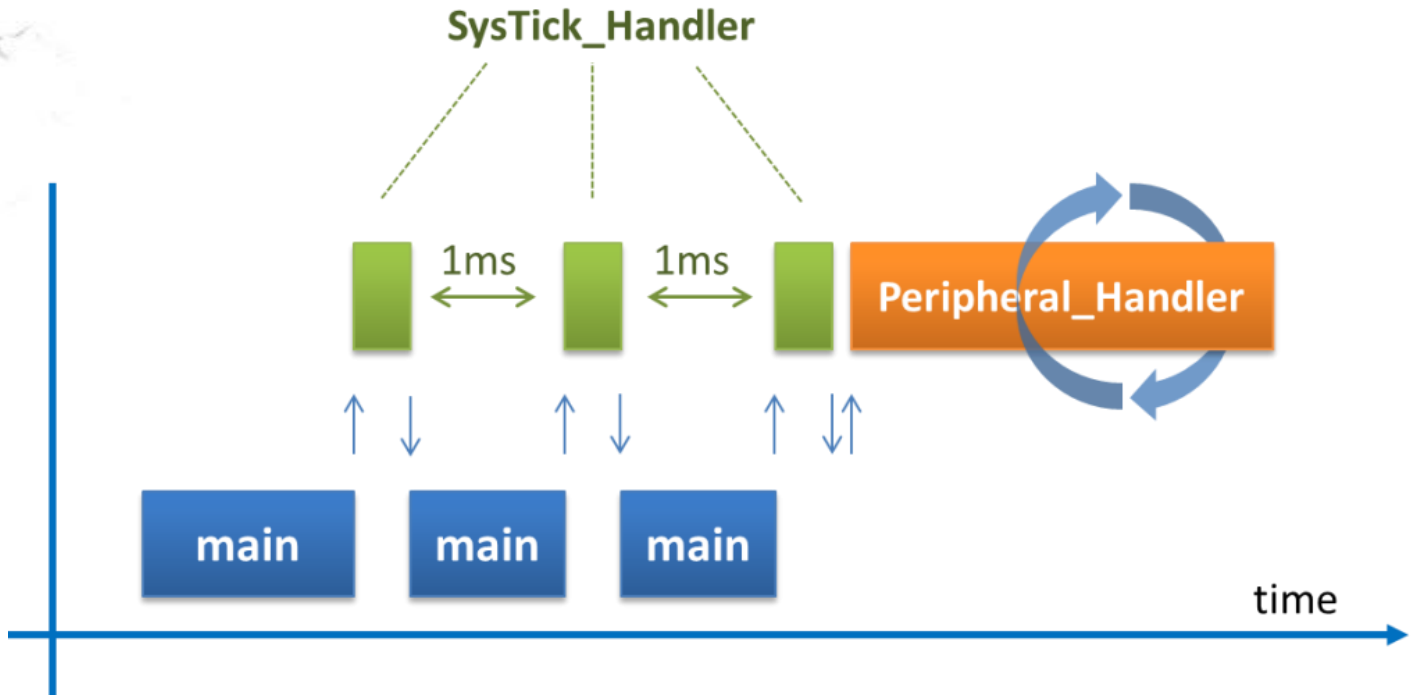


Sơ đồ hoạt động của SysTick

1.1. Khái quát về SysTick

1.1.1. Chức năng

- System Tick Timer thường được được dùng để tạo hàm delay với độ chính xác cao.
- Nhiệm vụ tạo ra ngắt SysTick theo chu kỳ thời gian cố định.

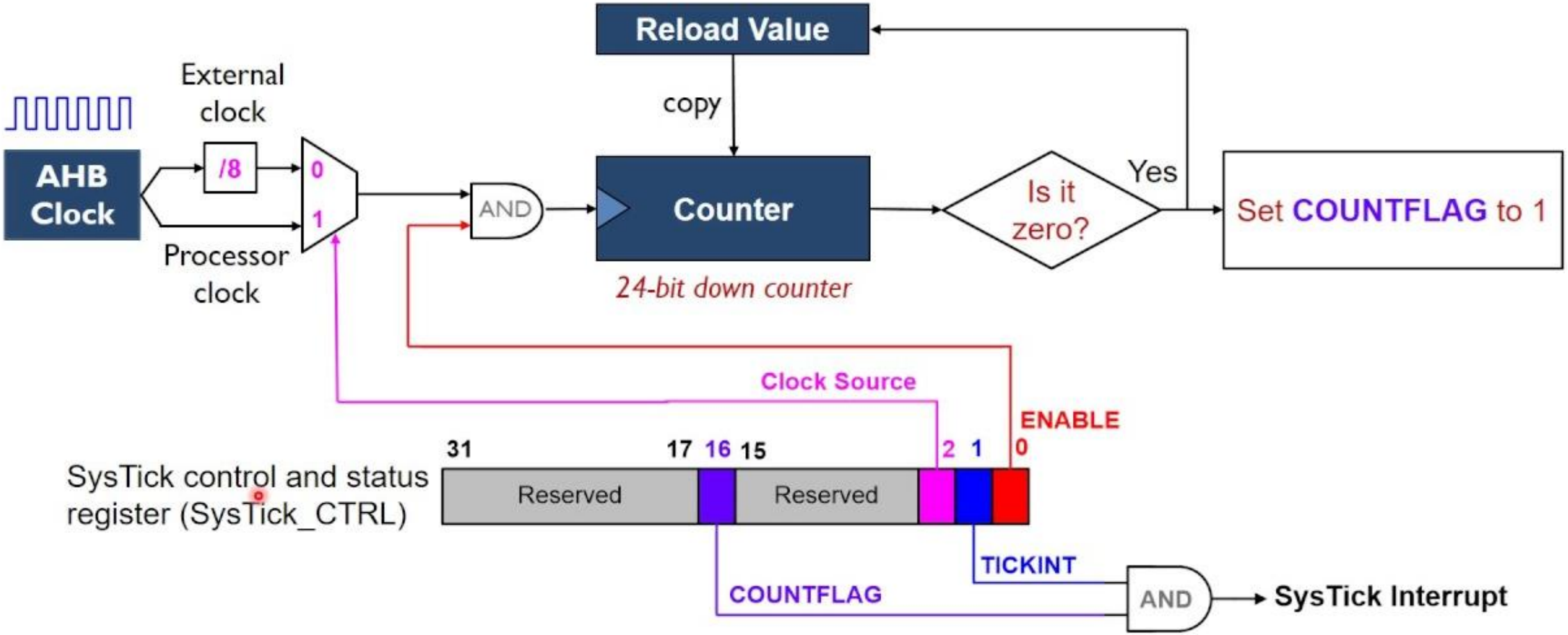


1.1. Khái quát về SysTick

1.1.2. Ứng Dụng

- Cung cấp khoảng ngắt định kỳ cho hệ điều hành thời gian thực RTOS
- Được sử dụng cho các mục đích ngắt có tính chu kỳ
- Tạo một khoảng thời gian trễ (delay)

1.2. Cơ chế hoạt động của SysTick

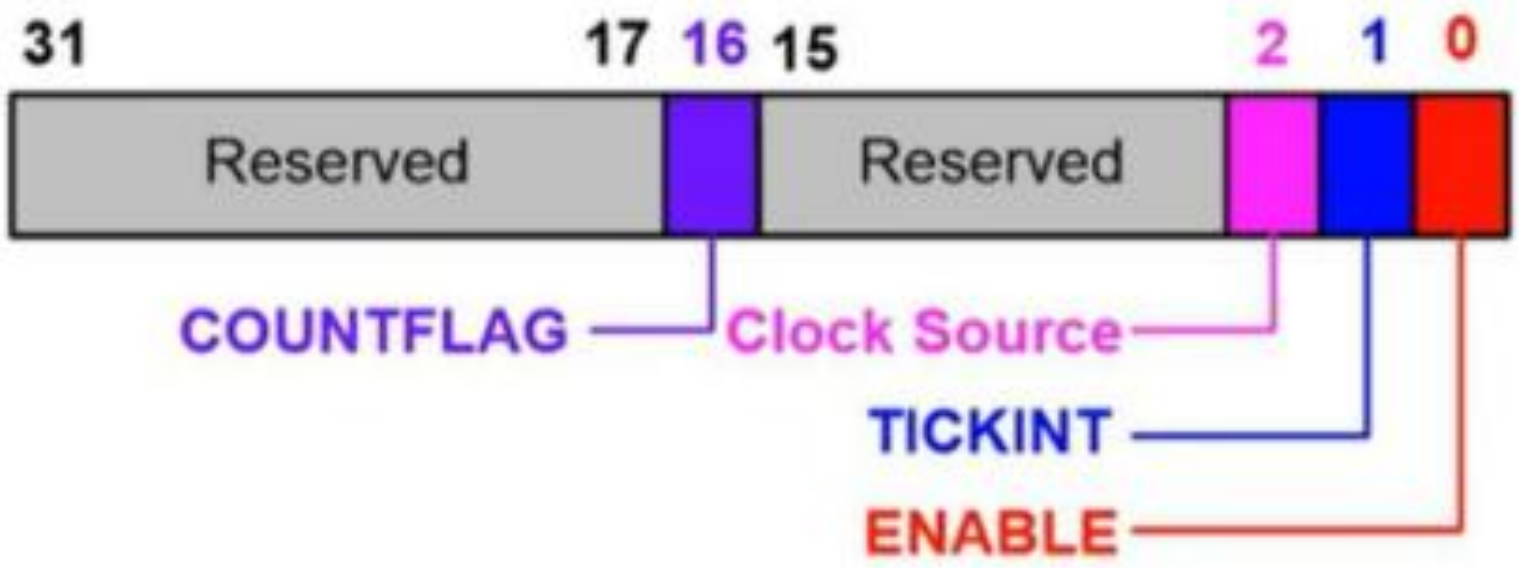


Sơ đồ hoạt động của SysTick

1.2. Cơ chế hoạt động của SysTick

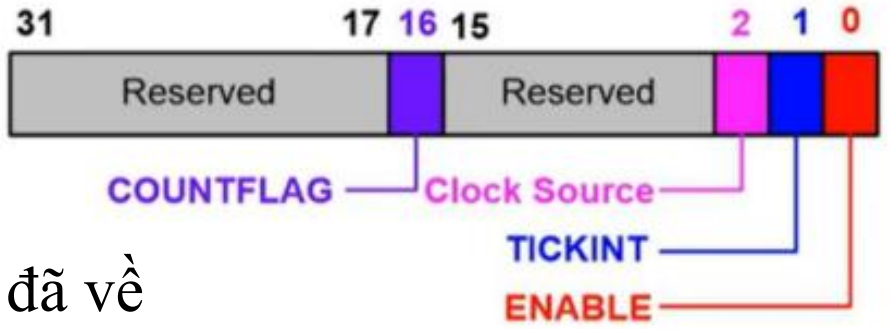
1.2.1. SysTick_CTRL (Control & Status Register)

Một số chức năng của thanh ghi CRTL: Bật/tắt SysTick, chọn nguồn xung clock (HCLK hoặc HCLK/8), kiểm tra cờ tràn (COUNTFLAG), cho phép/ngắt ngắt SysTick.



1.2. Cơ chế hoạt động của SysTick

- Các bit trong thanh ghi:



- Bit 0 – ENABLE (R/W): bật/tắt SysTick.
- Bit 1 – TICKINT (R/W): bật/tắt ngắt SysTick.
- Bit 2 – CLKSOURCE (R/W)
- Bit 16 – COUNTFLAG (RO): kiểm tra counter đã về 0 hay chưa.
- Các bit khác [31:17], [15:3]: Reserved (không dùng).

1.2. Cơ chế hoạt động của SysTick

1.2.2. SysTick_LOAD (Reload Value Register)

- Bit [31:24] – Reserved (Không dùng luôn bằng 0)
- Bit [23:0] – RELOAD : Giá trị nạp vào bộ đếm (CVR) mỗi khi counter đếm về 0

=> Đây chính là chu kỳ lặp của SysTick.

- Giá trị tối đa: 0x00FF.FFFF (16.777.215) : 24 bit
- Bộ đếm giảm dần từ giá trị RELOAD về 0.

1.2. Cơ chế hoạt động của SysTick

1.2.2. SysTick_LOAD (Reload Value Register)

- Ghi giá trị RELOAD = 0 sẽ vô hiệu hóa SysTick, không phụ thuộc vào bit TICKINT.
- Khoảng thời gian giữa hai ngắt SysTick:

$$\text{Interval} = (\text{RELOAD} + 1) \text{ x Chu kỳ Clock}$$

- Nếu có 100 chu kỳ clock giữa hai ngắt SysTick → RELOAD = 99.

$$Ts = 100 \times Tc \rightarrow (\text{RELOAD} + 1) \times Tc = 100 \times Tc \rightarrow \text{RELOAD} = 100 - 1 = 99$$

1.2. Cơ chế hoạt động của SysTick

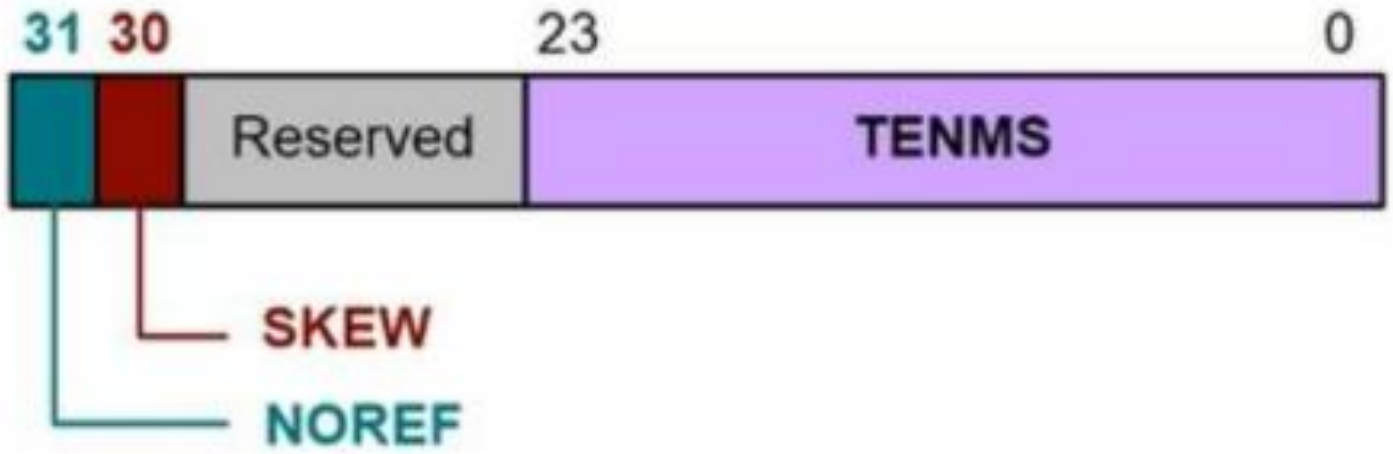
1.2.3. SysTick_VAL (Current Value Register)

- Khi đọc → trả về giá trị hiện tại của bộ đếm.
- Khi bộ đếm chuyển từ 1 xuống 0 → tạo ra một ngắt.
- Khi ghi vào SysTick_VAL → sẽ xóa bộ đếm và đưa COUNTFLAG về 0.

1.2. Cơ chế hoạt động của SysTick

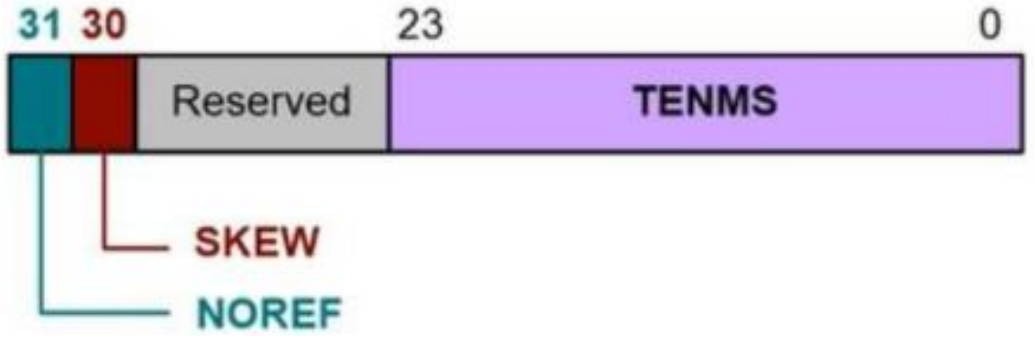
1.2.4. SysTick_CALIB (Calibration Register)

Thanh ghi chỉ đọc (Read-only register). Trong đóTENMS (10 ms): chứa giá trị reload, giá trị này tương ứng với chu kỳ 10 ms. Nhằm cung cấp giá trị hiệu chỉnh theo clock chuẩn 10ms và dùng để tạo thời gian chuẩn theo datasheet.



1.2. Cơ chế hoạt động của SysTick

1.2.4. SysTick_CALIB (Calibration Register)



- Bit 30 – SKEW
- Bit [29:24] – Reserved (không dùng).
- Bit [23:0] – TENMS: Chứa giá trị reload tương ứng với 10ms (tức 100 Hz) theo clock mặc định.
- Bit 31 – NOREF

1.3. Cách thay đổi ngắt thời gian

1.3.1. Công thức chu kỳ ngắt SysTick

$$T_s = (LOAD + 1) \times T_c$$

- **Ý nghĩa:** tính thời gian giữa hai lần ngắt SysTick liên tiếp
- **Trong đó:**

+ LOAD: giá trị nạp lại vào bộ đếm (0 → 0xFFFFFF).

+ T_c: chu kỳ của clock nguồn (nghịch đảo của tần số clock).

- Đây là công thức tổng quát, cho biết nếu ta nạp LOAD bao nhiêu thì khoảng thời gian ngắt sẽ là bao nhiêu.

1.3. Cách thay đổi ngắt thời gian

1.3.2. Công thức tính giá trị RELOAD

RELOAD = (Clock / Tần số mong muốn) − 1

=> Mất đúng (N+1)* chu kỳ clock thì counter mới đếm về 0, sinh ngắt

Muốn thay đổi tốc độ → chỉ cần thay đổi RELOAD hoặc thay đổi nguồn clock.

1.3. Cách thay đổi ngắt thời gian

1.3.2. Công thức tính giá trị RELOAD

Ví dụ:

- Clock = 80 MHz.
- Muốn ngắt mỗi 10 ms → f_mong_muon = 100 Hz.
- RELOAD = (80,000,000 / 100) – 1 = 799,999.

1.4. SysTick VÀ HAL

Một tập hợp các thư viện **chuẩn hóa do ARM phát triển** (ví CMSIS-Core: tuy cập thanh ghi bộ xử lý, Systick)

1.4.1. Phân tích chi tiết biến và dòng liên quan đến SysTick [3]

Biến	Ý nghĩa và chức năng
`__IO uint32_t uwTick`	• __IO là macro CMSIS thể hiện volatile (tham chiếu phần cứng/Interrupt Service Routine khi có thể thay đổi). • uwTick là biến đếm hệ thống (global tick). HAL_IncTick() tăng biến này mỗi lần SysTick IRQ (interrupt Request) xảy ra

1.4. SysTick VÀ HAL

1.4.1. Phân tích chi tiết biến và dòng liên quan đến SysTick [3]

Biến	Ý nghĩa và chức năng
`uint32_t uwTickPrio = (1UL << __NVIC_PRIO_BITS)`	• __NVIC_PRIO_BITS là số bit dùng cho priority trong NVIC (device header). • Sau khi HAL_InitTick() thành công sẽ gán uwTickPrio = TickPriority. • Ý nghĩa: Giá trị ưu tiên ngắt SysTick trong NVIC. Khởi tạo mặc định chưa hợp lệ, sẽ được gán khi gọi HAL_InitTick().

1.4. SysTick VÀ HAL

1.4.1. Phân tích chi tiết biến và dòng liên quan đến SysTick [3]

Biến	Ý nghĩa và chức năng
`HAL_TickFreqTypeDef uwTickFreq = HAL_TICK_FREQ_DEFAULT`	uwTickFreq lưu "số milliseconds được cộng vào uwTick mỗi tick". Giá trị enum (từ HAL header) thường là: • HAL_TICK_FREQ_1KHZ = 1U (1 ms per tick, tức 1 kHz tickbase) • HAL_TICK_FREQ_100HZ = 10U (10 ms per tick) • HAL_TICK_FREQ_10HZ = 100U (100 ms per tick) HAL_TICK_FREQ_DEFAULT = HAL_TICK_FREQ_1KHZ (tức là giá trị thực tế là số ms / interrupt).

1.4. SysTick VÀ HAL

NVIC là một phần tích hợp trong CPU giúp quản lý và xử lý ngắt (interrupts)

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
HAL_Init(void) Hàm khởi tạo nền tảng của thư viện HAL trong STM32	**HAL_NVIC_SetPriorityGrouping(NVIC_PRIORITYGROUP_4);** Thiết lập cách phân chia **preemption priority và subpriority** trong NVIC. => Không tác động trực tiếp đến SysTick nhưng ảnh hưởng toàn bộ hệ thống ngắt. **HAL_InitTick(TICK_INT_PRIORITY);** Liên quan trực tiếp đến **HAL_Delay()** và thời gian hệ thống. => HAL không kiểm tra lỗi → nếu thất bại có thể gây **treo chương trình** khi dùng HAL_Delay(). **HAL_MspInit();** Khởi tạo **MSP (MCU Support Package)**. => Người dùng cấu hình ngoại vi: bật xung nhịp, GPIO, DMA, v.v.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
__weak HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)	**1. Cấu hình SysTick:** **HAL_SYSTICK_Config(SystemCoreClock / (1000U / uwTickFreq))** • SystemCoreClock: tần số xung nhịp CPU (Hz). • 1000U / uwTickFreq: mẫu số thể hiện số lần ngắt trên mỗi giây. • Kết quả của phép chia cho biết số chu kỳ clock cần đếm trước khi sinh ra một ngắt SysTick.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
__weak HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)	**Ví dụ minh họa** với SystemCoreClock = 72 MHz: Nếu uwTickFreq = 1 (mặc định, tương ứng 1 ms mỗi tick): 1000/1=1000 ⇒ 72,000,000 / 1000=72,000 SysTick sẽ được cấu hình với LOAD = 71,999 → tạo ra ngắt mỗi 1 ms. Nếu uwTickFreq = 10 (tức 10 ms mỗi tick): 1000/10 = 100 ⇒ 72,000,000 / 100 = 720,000 SysTick sẽ đếm 720,000 chu kỳ LOAD = 719,999 → ngắt xảy ra mỗi 10 ms.

1.4. SysTick VÀ HAL

NVIC là một phần tích hợp trong CPU giúp quản lý và xử lý ngắt (interrupts)

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
__weak HAL_StatusTypeDef HAL_InitTick(uint32_t TickPriority)	**2. Thiết lập độ ưu tiên ngắt** **__NVIC_PRIO_BITS:** số bit dùng cho cấu hình mức ưu tiên (tùy dòng chip, thường là 4 bit). Nếu giá trị TickPriority hợp lệ, hàm **HAL_NVIC_SetPriority()** sẽ được gọi để: Gán mức ưu tiên cho SysTick_IRQn. Cập nhật biến toàn cục uwTickPrio lưu lại giá trị này. Nếu TickPriority vượt giới hạn cho phép, hàm trả về lỗi (HAL_ERROR).

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính

`__weak void HAL_IncTick(void)`
`{`
`  uwTick += uwTickFreq;`
`}`
	Chức năng chính
	Tăng biến toàn cục uwTick mỗi lần SysTick ngắt.
	uwTickFreq xác định độ phân giải:
	=1: tăng 1 ms mỗi tick (mặc định).
	=10: tăng 10 ms mỗi tick.
	Cơ chế gọi hàm
	Thường được gọi trong SysTick Handler (HAL_SYSTICK_IRQHandler hoặc SysTick_Handler).
	Cập nhật thời gian hệ thống mỗi khi có SysTick IRQ.
	Ý nghĩa từ khóa __weak
	Cho phép override hàm trong main.c.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính

`__weak uint32_t HAL_GetTick(void) {`
`  return uwTick;`
`}`
	Hàm trả về giá trị hiện tại của biến toàn cục **uwTick**.
	
	Vì uwTick được tăng đều trong hàm HAL_IncTick() (mặc định gọi trong SysTick Handler)
	
	Giá trị trả về của HAL_GetTick() chính là thời gian hệ thống (system time) tính theo đơn vị millisecond.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính

__weak void HAL_Delay(uint32_t Delay)
uint32_t tickstart = HAL_GetTick();
uint32_t wait = Delay;
 	
if (wait < HAL_MAX_DELAY)
{
  wait += (uint32_t)(uwTickFreq);
}
 	
while ((HAL_GetTick() - tickstart) < wait)
{
	HAL_Delay() cung cấp một phương pháp đơn giản để chờ theo mili-giây, dựa trên uwTick.
	
	Khi gọi hàm, chương trình sẽ dừng lại trong vòng while cho đến khi số mili-giây trôi qua đạt yêu cầu.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
__weak void HAL_SuspendTick(void) CLEAR_BIT(SysTick->CTRL, SysTick_CTRL_TICKINT_Msk);	SysTick->CTRL là registry điều khiển SysTick (CMSIS define). SysTick_CTRL_TICKINT_Msk là mask bit TICKINT (bit bật/tắt interrupt). **HAL_SuspendTick()** tắt bit ngắt → SysTick counter có thể vẫn chạy nhưng không sinh ngắt, do đó HAL_IncTick() không được gọi và uwTick không tăng → HAL_Delay() treo nếu được gọi.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính
__weak void HAL_ResumeTick(void) SET_BIT(SysTick->CTRL, SysTick_CTRL_TICKINT_Msk);	HAL_ResumeTick() bật lại interrupt. Lưu ý: Tạm dừng cả counter (không chỉ interrupt) thì cần thao tác thêm với SysTick->CTRL (ENABLE bit), nhưng HAL chỉ bật/tắt TICKINT.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính

HAL_StatusTypeDef HAL_SetTickFreq(HAL_TickFreqTypeDef Freq)	**1.Kiểm tra tham số**
	assert_param(IS_TICKFREQ(Freq));
	Đảm bảo giá trị truyền vào Freq hợp lệ (ví dụ HAL_TICK_FREQ_10HZ).
	
	2. Lưu tạm giá trị cũ
	backup prevTickFreq = uwTickFreq;
	uwTickFreq = Freq;
	Lưu lại giá trị tick frequency hiện tại (prevTickFreq) để khôi phục nếu cấu hình mới thất bại.
	Gán uwTickFreq bằng giá trị mới Freq.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Các lệnh chính

HAL_StatusTypeDef HAL_SetTickFreq(HAL_TickFreqTypeDef Freq)	3. **Tái cấu hình SysTick**
	status = HAL_InitTick(uwTickPrio);
	Gọi lại HAL_InitTick() để tính toán lại giá trị reload của SysTick dựa trên uwTickFreq mới.
	Nếu cấu hình thành công → status = HAL_OK.
	Nếu thất bại → status = HAL_ERROR.
	4. Khôi phục khi lỗi
	if (status != HAL_OK) { uwTickFreq = prevTickFreq; }
	Nếu không cấu hình được SysTick với tần số mới, hàm khôi phục uwTickFreq bằng giá trị cũ.

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Chức năng
HAL_IncTick(void)	Tăng biến uwTick(biến thời gian hệ thống (system tick counter) mỗi 1 ms. Được gọi trong SysTick_Handler().
HAL_GetTick(void)	Lấy giá trị tick hiện tại (đơn vị: ms). Dùng để đo thời gian hoặc timeout.
HAL_Delay(uint32_t Delay)	Tạo delay theo ms bằng cách chờ HAL_GetTick() tăng. Blocking

1.4. SysTick VÀ HAL

1.4.2. Các hàm HAL liên quan đến Systick

Hàm	Chức năng
HAL_SYSTICK_Config(uint32_t TicksNumb)	Cấu hình SysTick theo số tick (thường gọi trong HAL_Init).
HAL_SYSTICK_IRQHandler(void)	Hàm xử lý ngắt SysTick (được gọi trong SysTick_Handler).
HAL_SYSTICK_Callback(void)	Callback weak, override để viết code chạy mỗi 1 ms.

1.4. SysTick VÀ HAL

1.4.3. Các thành phần phần cứng (SysTick registers)

Hàm	Chức năng
SysTick_Config(uint32_t ticks)	Hàm CMSIS: Cấu hình reload value, bật counter và ngắt. Thường dùng trong bare-metal (không HAL).
SysTick_Handler(void)	Hàm xử lý ngắt SysTick (gốc). Trong HAL, gọi HAL_IncTick() và HAL_SYSTICK_IRQHandler().
SysTick->CTRL	Thanh ghi điều khiển (Enable, TickInt, Clock source, COUNTFLAG).

1.4. SysTick VÀ HAL

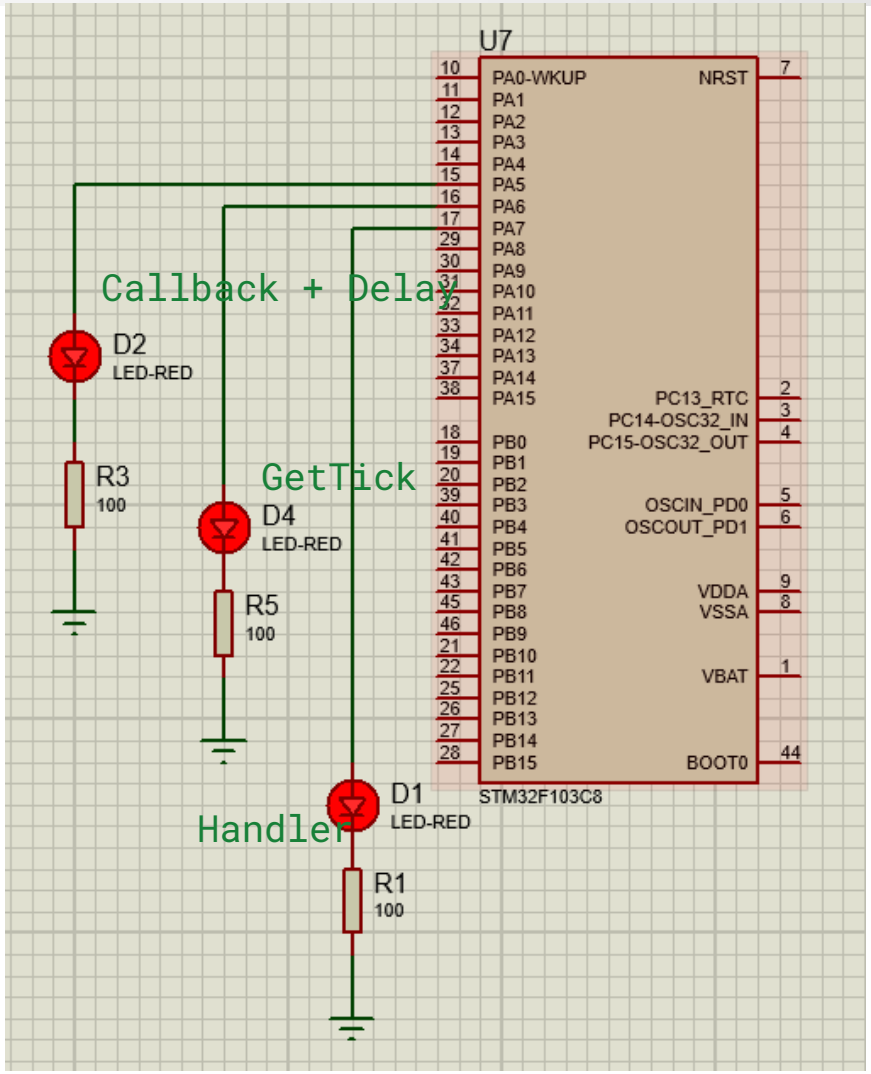
1.4.3. Các thành phần phần cứng (SysTick registers)

Hàm	Chức năng
SysTick->LOAD	Giá trị nạp lại (reload value) – khi SysTick đếm từ đây về 0 sẽ tạo ngắt.
SysTick->VAL	Giá trị hiện tại của bộ đếm – có thể reset về 0 để bắt đầu lại chu kỳ.
SysTick->CALIB	Giá trị hiệu chỉnh chuẩn (chuẩn 1ms nếu dùng HSI hoặc HSE). Chỉ có ý nghĩa nếu dùng trong chế độ auto-calibration (ít dùng trong HAL).

1.5. Code demo

1.5.1. SysTick và HAL

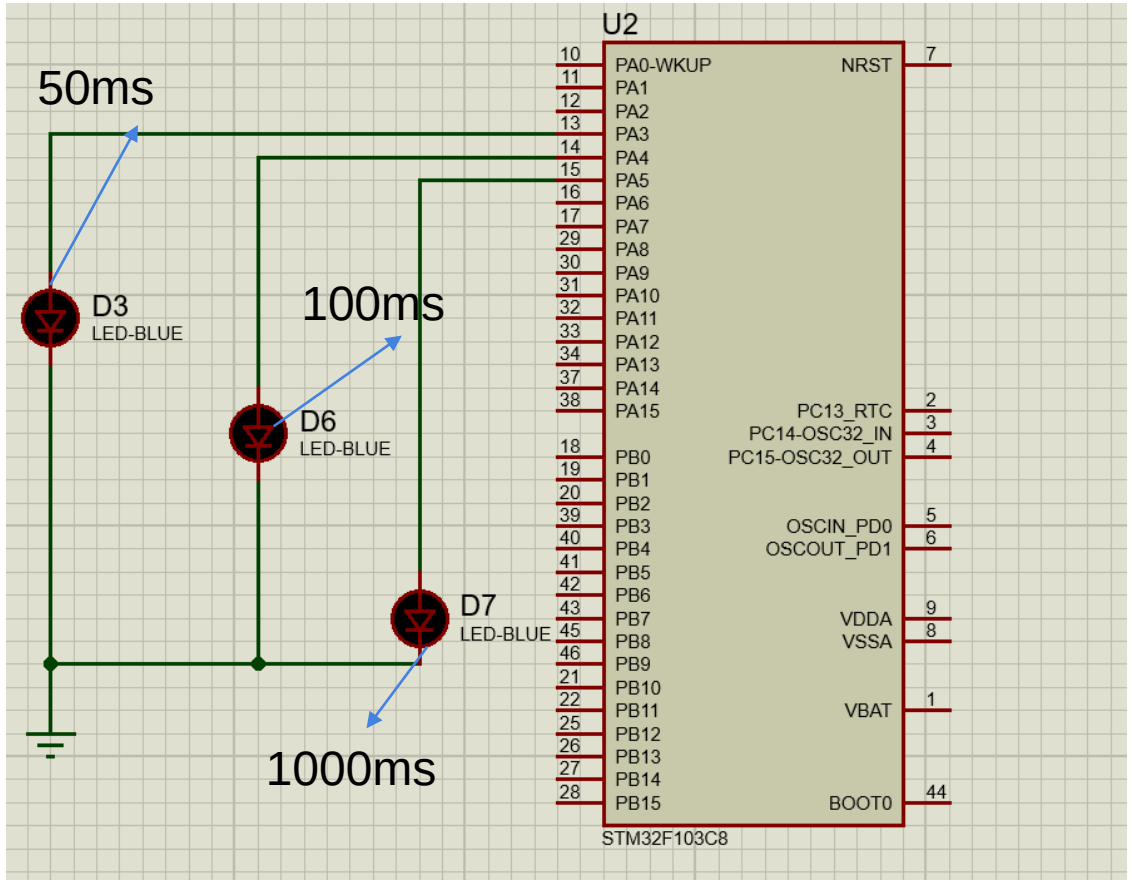
- Mô phỏng:
- Bài tập SYSTICK chớp-tắt led cho STM32
- (dòng STM32F103C08) để so sánh tốc độ
- Soure code: Data_Science/Systick.zip at main · giunzz/Data_Science



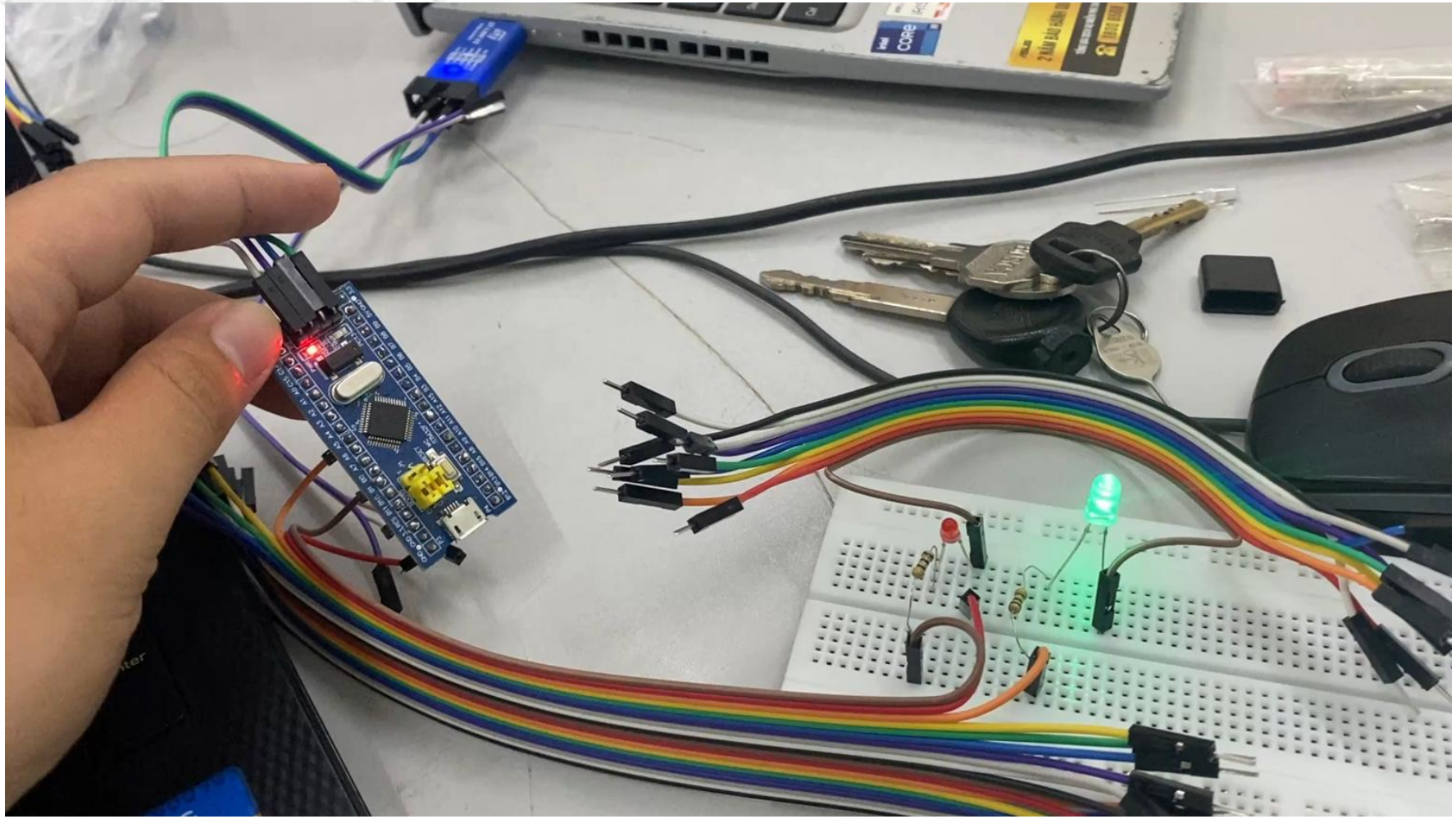
1.5. Code demo

1.5.2. SysTick viết hàm sử dụng thanh ghi

- Mô phỏng:
- Bài tập SYSTICK chớp-tắt led cho STM32
- (dòng STM32F103C08) để so sánh tốc độ
- Soure code: giunzz/STM32_GK: STM32
- Video thực nghiệm:



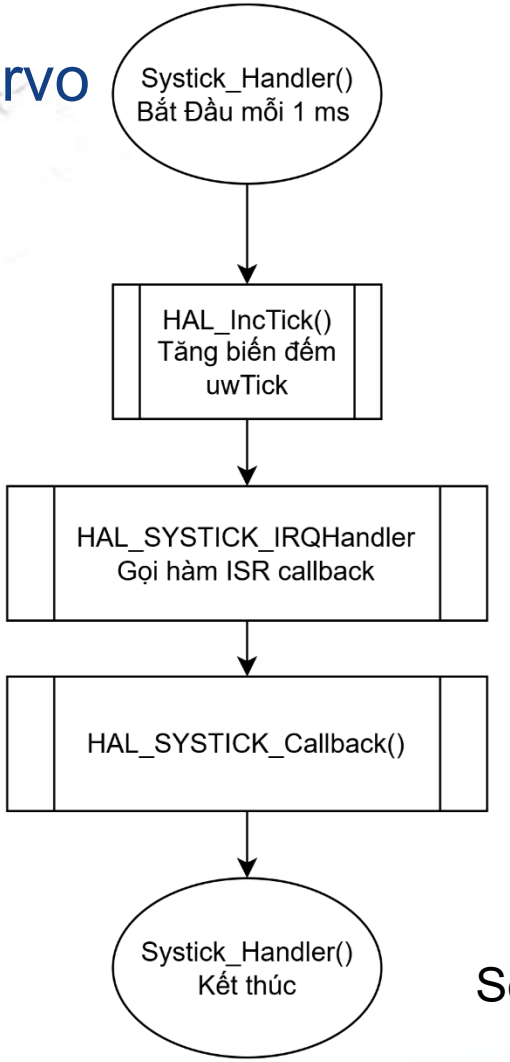
1.5. Video thực nghiệm



1.5. Code demo

1.5.3. Sử dụng API điều khiển servo

Ví dụ: quá trình điều khiển động cơ cho cá ăn. Mỗi khi có lệnh cho ăn thì sẽ thực hiện đếm lên 1000 tick tương ứng với 1s



Sơ đồ ngắt Systick sử dụng API.

1.4. Code demo

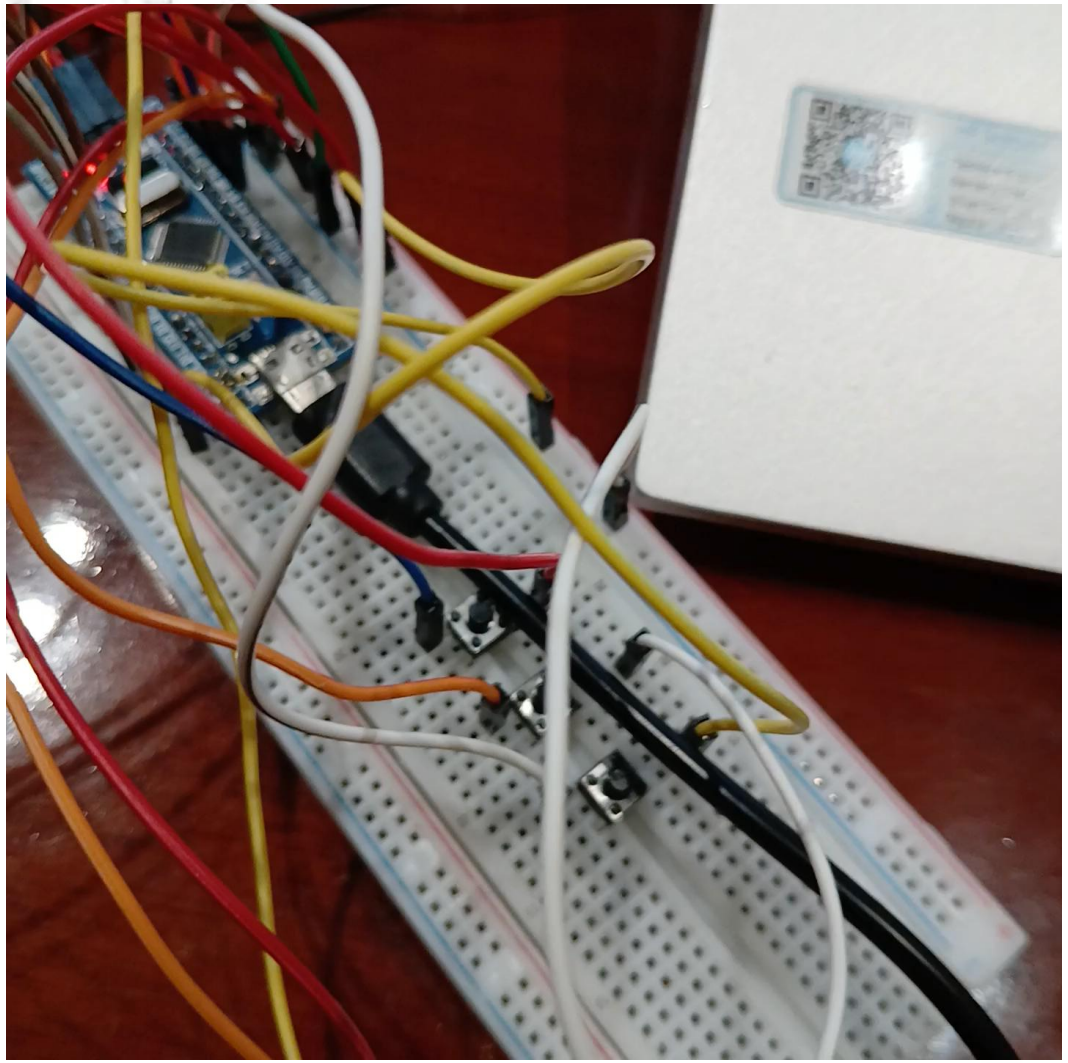
1.5.3. Sử dụng API

```
void SysTick_Handler(void)
{
 /*
 USER CODE BEGIN SysTick_IRQn 0 */
 /* USER CODE END SysTick_IRQn 0 *
 HAL_IncTick();
 /* USER CODE BEGIN SysTick_IRQn 1 */
 HAL_SYSTICK_IRQHandler();
/* USER CODE END SysTick_IRQn 1 */
}
```

```
void HAL_SYSTICK_IRQHandler(void)
{
 HAL_SYSTICK_Callback();
}

void HAL_SYSTICK_Callback(void)
{
   // Ví dụ: quay động cơ trong 1s
   static uint32_t counter = 0;
   if(counter >= 1000) {  // 1000 * 1ms = 1s
    feed.IsFeed = 3;
       counter = 0;
   }
   if (feed.IsFeed == 2) {
       counter++;
   }
}
```

1.5. Video thực nghiệm

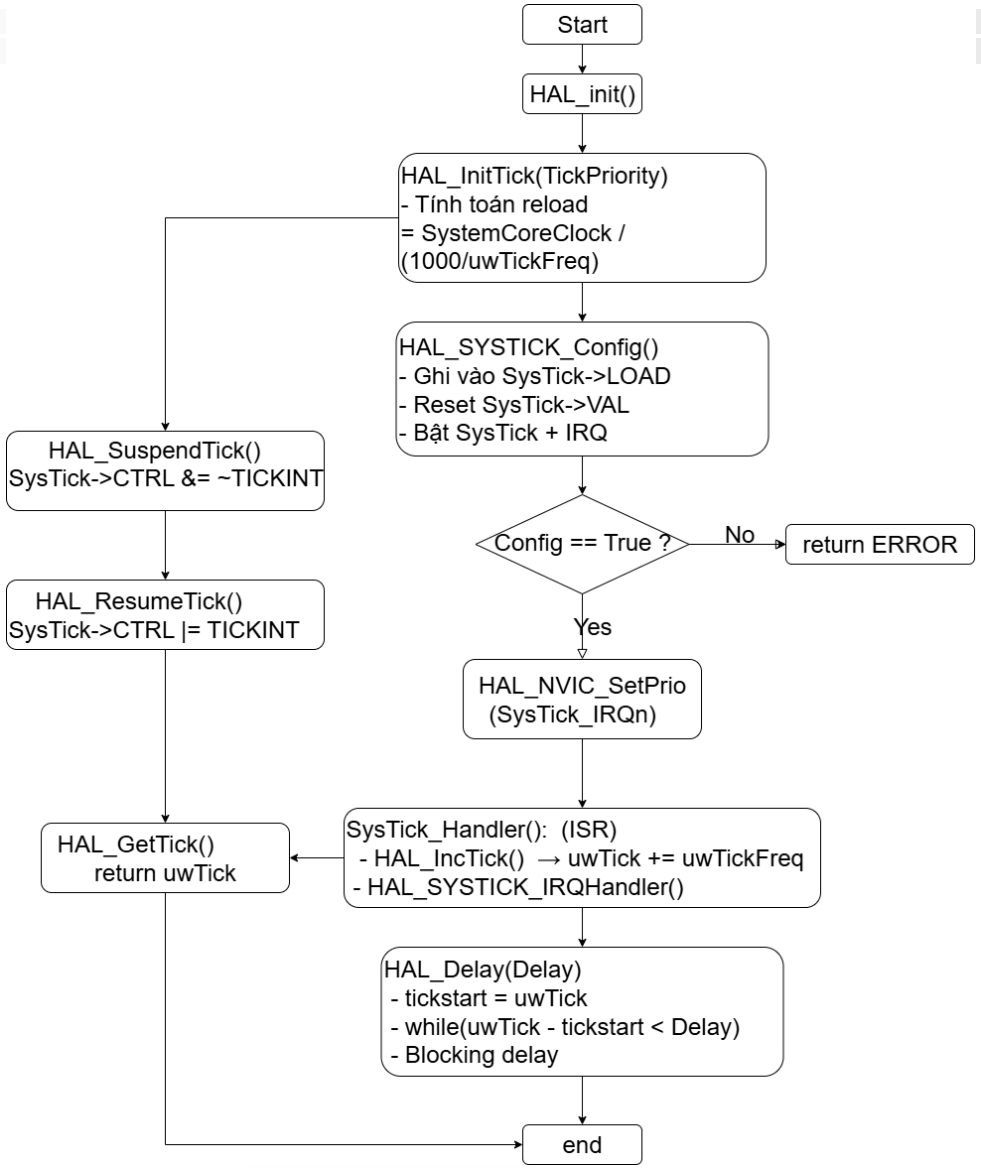


TỔNG KẾT

- Phần lý thuyet: báo cáo đã hệ thống hoá kiến thức về System Tick Timer (SysTick) trên ARM Cortex-M: từ vai trò của bộ đếm 24-bit, nguồn xung lựa chọn, các thanh ghi điều khiển/trạng thái và cơ chế tạo ngắt định kỳ, đến những mẫu lập trình tiêu biểu (khởi tạo, delay bằng ngắt/polling).

- Dựa trên cơ sở lý thuyết, ứng dụng minh hoạ cách phối hợp SysTick làm Bài tập SYSTICK chớp-tắt led cho STM32 (dòng STM32F103C08) để so sánh tốc độ thay đổi nhằm triển khai cơ chế tạo nhịp bằng polling SysTick để tạo bước thời gian 1 ms ở cấu hình chuẩn HSI 8 MHz. Mục tiêu tạo trễ 1 ms bằng polling SysTick (LOAD=8000−1, đếm theo HCLK, chờ COUNTFLAG, không dùng ngắt). Vì vậy khi giữ LOAD nhưng tăng HCLK (PLL 72 MHz) → chu kỳ giảm; chuyển CLKSOURCE sang HCLK/8 → chu kỳ tăng; giữ HCLK nhưng đổi LOAD → điều chỉnh chu kỳ có kiểm soát.

TỔNG KẾT



- Hình trên mô tả cơ chế hoạt động của SysTick trong HAL STM32.

TÀI LIỆU THAM KHẢO

- [1] Embedded Systems and Deep Learning. (n.d.). Lecture 12: System Timer (SysTick) [Video]. YouTube. https://www.youtube.com/watch?v=aLCUDv_fgoU
- [2] M. (2010). Cortex-M3 devices generic user guide (Rev. r2p1). ARM Ltd. https://developer.arm.com/documentation/dui0552/a
- [3] STMicroelectronics, “Description of STM32L0 HAL and low-layer drivers,” UM1749, ST, 2017. [Online]. Available: https://www.st.com/resource/en/user_manual/um1749-description-of-stm32l0-hal-and-low-layer-drivers-stmicroelectronics.pdf. [Accessed: 29-Sep-2025].