TAPiT

KHÓA HỌC
LẬP TRÌNH VI ĐIỀU KHIỂN

Giảng viên
NGUYỄN HUỲNH NHẬT THƯƠNG

LỊCH HỌC:
19h30 - 22h00 thứ 2 và thứ 6

ĐỊA ĐIỂM:
Online qua nền tảng Zoom / Google Meet

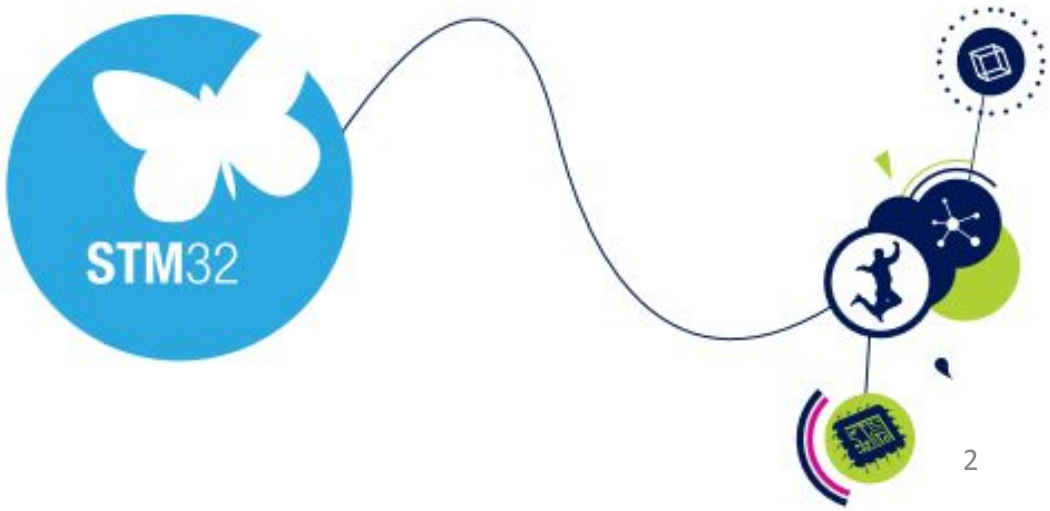
MODULE 4
- CLOCK TREE
- CÁC CHẾ ĐỘ TIẾT KIỆM NĂNG LƯỢNG

0981001119 tapitlrs@gmail.com https://tapit.vn

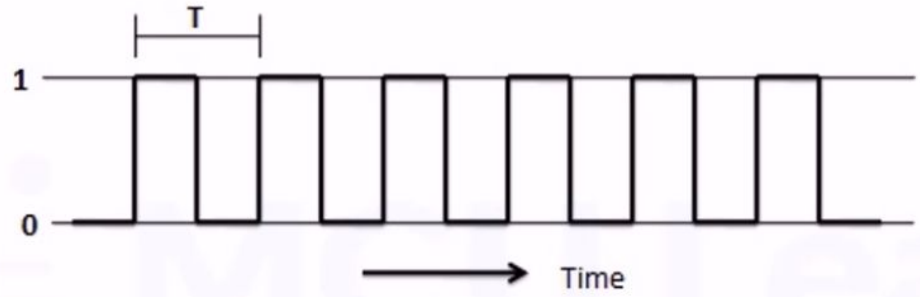
TAPIT
BUILD THE FUTURE WITH US!
Learning - Research - Sharing Community

TAPiT

MODULE 4
▪CLOCK TREE



TÍN HIỆU CLOCK



- Tín hiệu clock được sử dụng để đồng bộ hoạt động của các mạch điện bên trong các IC số hoặc để đồng bộ IC với các mạch khác ở bên ngoài.
- Tín hiệu clock là một tín hiệu dao động định kì
- Tần số, đơn vị Hertz (Hz)
- Tín hiệu clock giống như nhịp tim của thiết bị điện tử

THIẾT KẾ CLOCK

Tần số clock trong vi điều khiển liên quan đến hiệu năng và mức độ tiêu thụ năng lượng

THIẾT KẾ CLOCK CỦA VI ĐIỀU KHIỂN

- Lựa chọn các nguồn cung cấp clock khác nhau
- Tổ chức theo một cấu trúc phân cấp
- Thực hiện được việc bật/tắt cho từng khối
- Cấu hình tốc độ riêng biệt cho từng khối

Hãng ST đã thiết kế và cung cấp cho người dùng khả năng làm việc với toàn bộ clock của vi điều khiển STM32 một cách đơn giản nhất thông qua Giao diện STM32CubeMX và module RCC của thư viện HAL.

NGUỒN CUNG CẤP CLOCK

HIGH SPEED

Nguồn clock chính cung cấp cho vi điều khiển

> Bộ dao động nội RC, được gọi là High Speed Internal – HSI
> Bộ dao động thạch anh bên ngoài, được gọi là High Speed External – HSE

Lựa chọn thạch anh ngoại hay bộ dao động RC?

- Độ chính xác của thạch anh cao hơn so với RC (RC có độ chính xác ~ 1%, thạch anh có độ chính xác ~ phần triệu ppm)
- Một số ngoại vi chỉ có thể chạy được bằng thạch anh ngoài với một tần số nhất định.
- Sử dụng thạch anh ngoài thì sẽ tốn hơn về linh kiện và diện tích board mạch

NGUỒN CUNG CẤP CLOCK

HIGH SPEED

Nguồn clock chính cung cấp cho vi điều khiển

- Bộ dao động nội RC, được gọi là High Speed Internal – **HSI**
- Bộ dao động thạch anh bên ngoài, được gọi là High Speed External – **HSE**

LOW SPEED

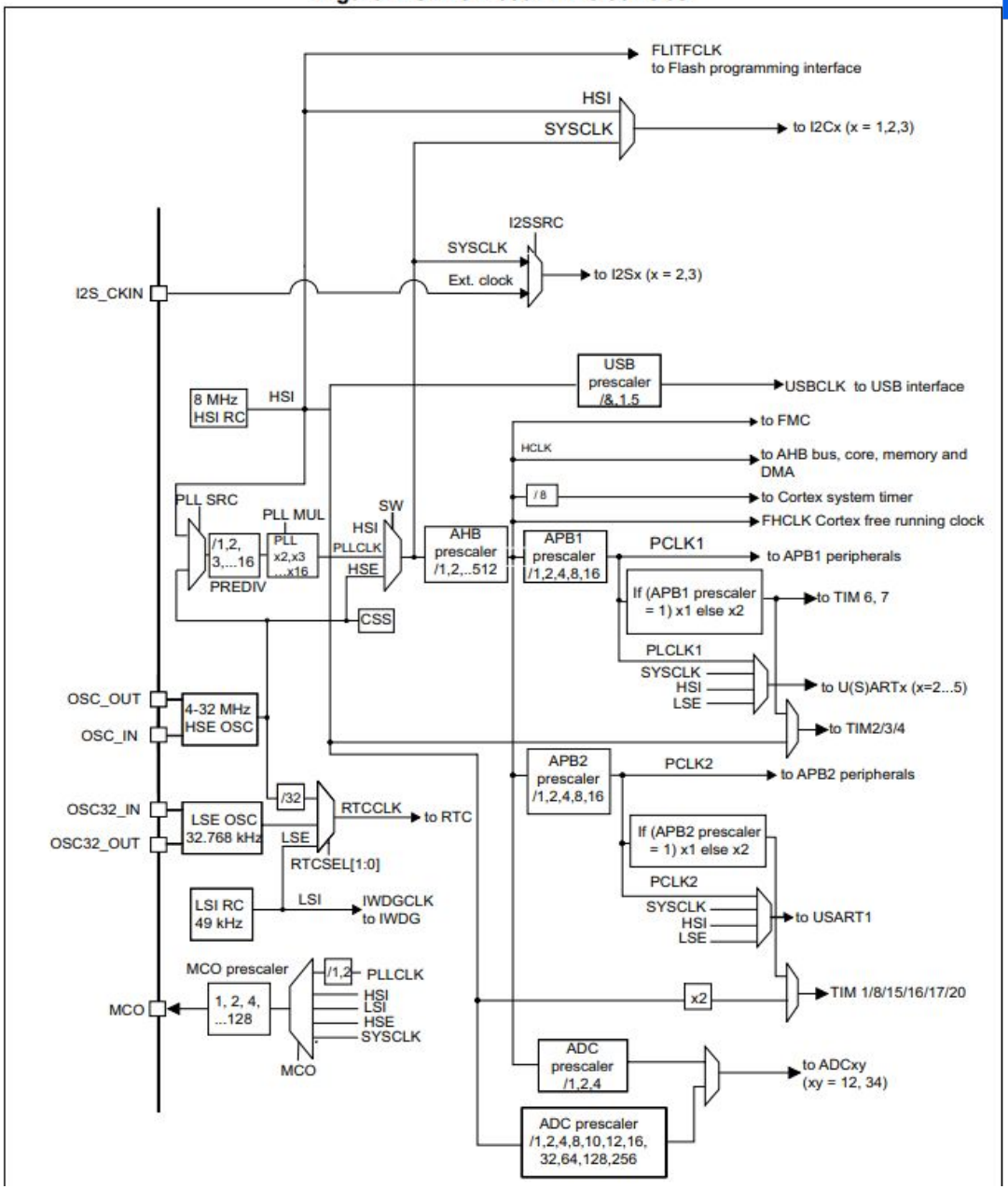
Sử dụng cho RTC và IWDG

- Bộ dao động nội RC tốc độc thấp chuyên dụng, gọi là Low Speed Internal - **LSI**
- Bộ dao động thạch anh ngoài tốc độ thấp được gọi là Low Speed External – **LSE**

CLOCK TREE

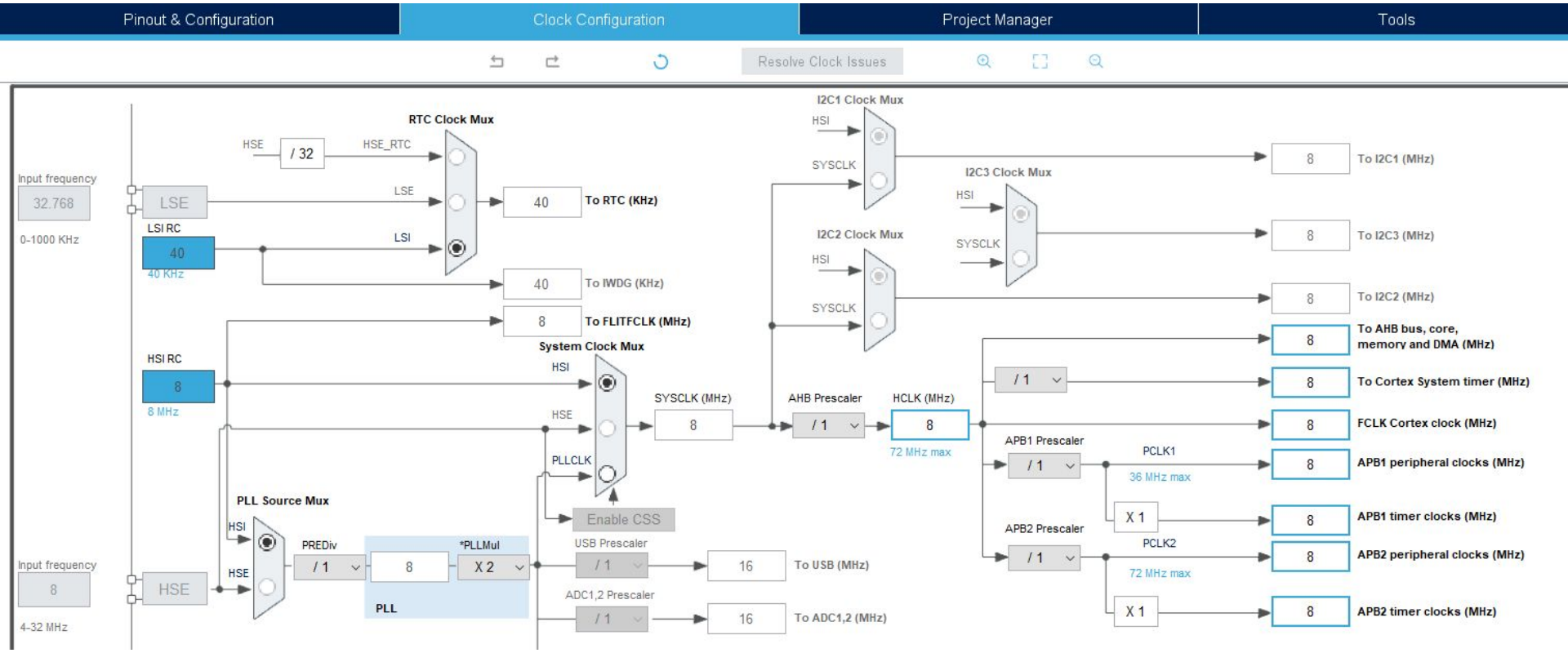
CLOCK TREE

Figure 2. STM32F303xD/E clock tree



Ref Datasheet: https://www.st.com/resource/en/datasheet/stm32f303re.pdf

CubeMX Clock Configuration



CLOCK TREE

Một số thành phần của Clock tree

- Nguồn cung cấp clock cho vi điều khiển HSI, HSE, LSI và LSE
- Các mạch chọn kênh (multiplexer): Lựa chọn đường tín hiệu clock
- Bộ chia tần Prescaler: Giảm tần số
- Vòng lặp khoá pha PLL: Tăng tần số

Việc cấu hình tăng/giảm tần số phục thuộc vào:

- Nhu cầu hiệu suất
- Tốc độ tối đa của ngoại vi
- Năng lượng tiêu thụ của thiết bị

CẤU HÌNH CLOCK TREE

Bước 1: Cấu hình sử dụng nguồn dao động tốc độ cao và nguồn dao động tốc độ thấp

Bước 2: Cấu hình SYSCLCK

Bước 3: Đúng AHB prescaler, APB1 prescaler và APB2 Prescaler

Việc cấu hình và quản lý clock tree của vi điều khiển có thể được thực hiện rất dễ dàng qua giao diện CubeMX.

CẤU HÌNH NGUỒN CLOCK

Khi khởi động:

-> HSI và LSI là nguồn clock mặc định có thể sử dụng

-> HSI cũng là nguồn clock vào mặc định cho SYSCLOCK

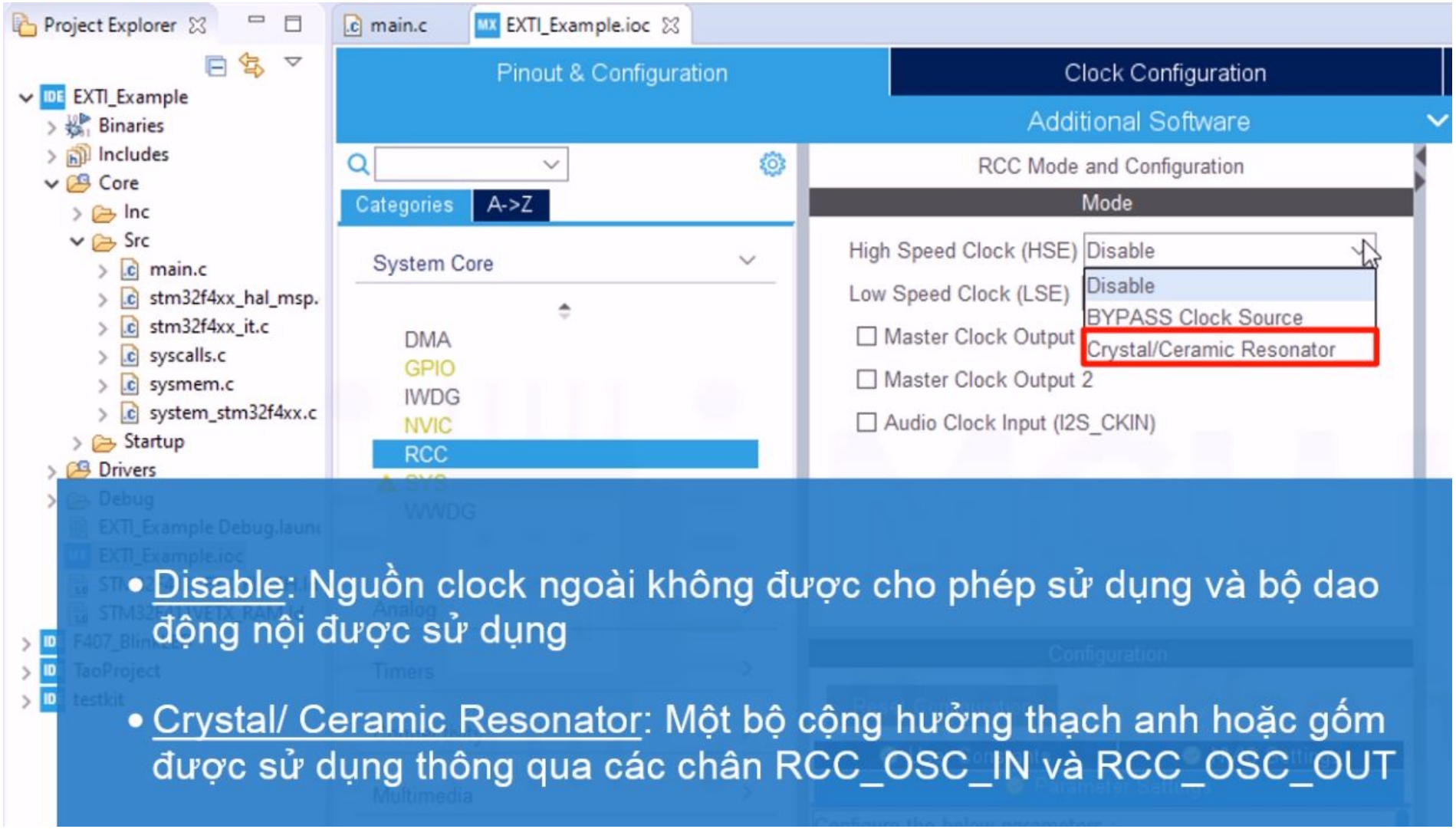
Nếu board mạch thực hành có hỗ trợ thạch anh ngoài cho HSE và LSE thì có thể cấu hình sử dụng các thạch anh này

Cấu hình lựa chọn bộ giao động ngoại thạch anh:

Pinout & Configuration □ RCC □ HighSpeed External □ Crystal/Ceramic Resonator

⇒ Kiểm tra tài liệu User Manual và phần cứng của board thực hành

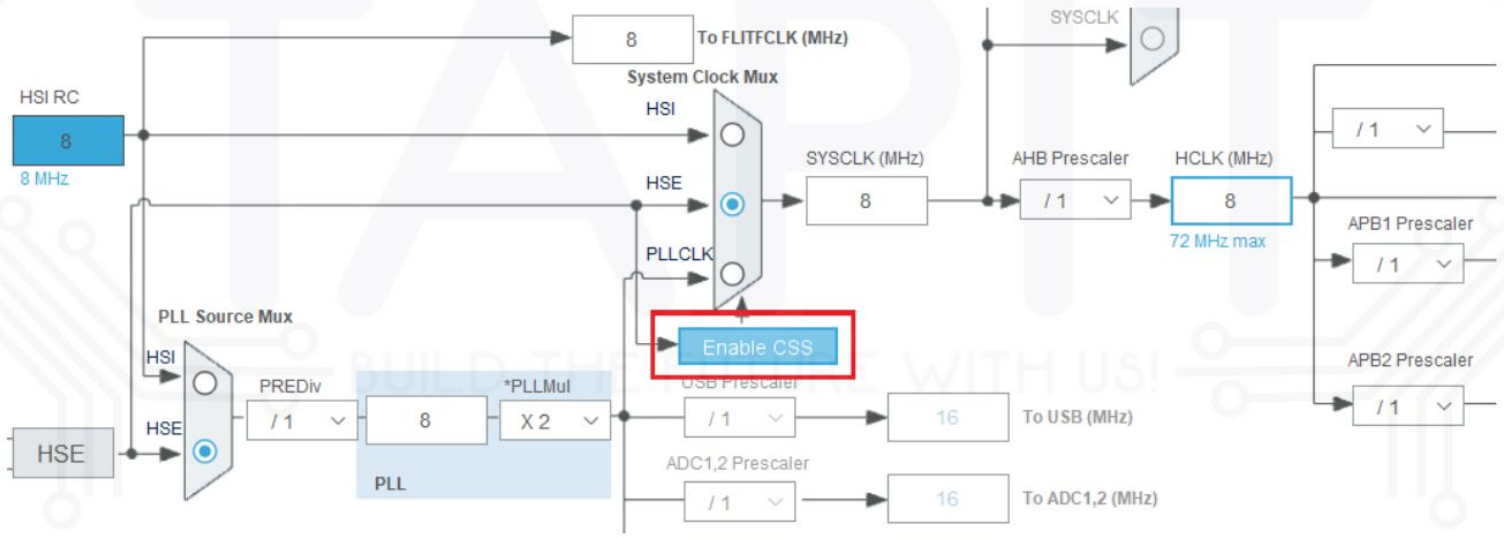
CẤU HÌNH NGUỒN CLOCK



- Disable: Nguồn clock ngoài không được cho phép sử dụng và bộ dao động nội được sử dụng
- Crystal/ Ceramic Resonator: Một bộ cộng hưởng thạch anh hoặc gốm được sử dụng thông qua các chân RCC_OSC_IN và RCC_OSC_OUT

Tìm hiểu Clock Security System (CSS)

trên vi điều khiển STM32 qua ví dụ thực tế



Clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	PLL RDY	PLLON	Res	Res	Res	Res	CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Clock configuration register (RCC_CFGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLNO DIV	MCOPRE[2:1]		MCOF / MCOP RE0	Res	MCO[2:0]			I2SSRC	USBPR E	PLLMUL[3:0]				PLL XTPRE	PLL SRC
rw	rw	rw	r / rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLSR C(1)	Res	PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

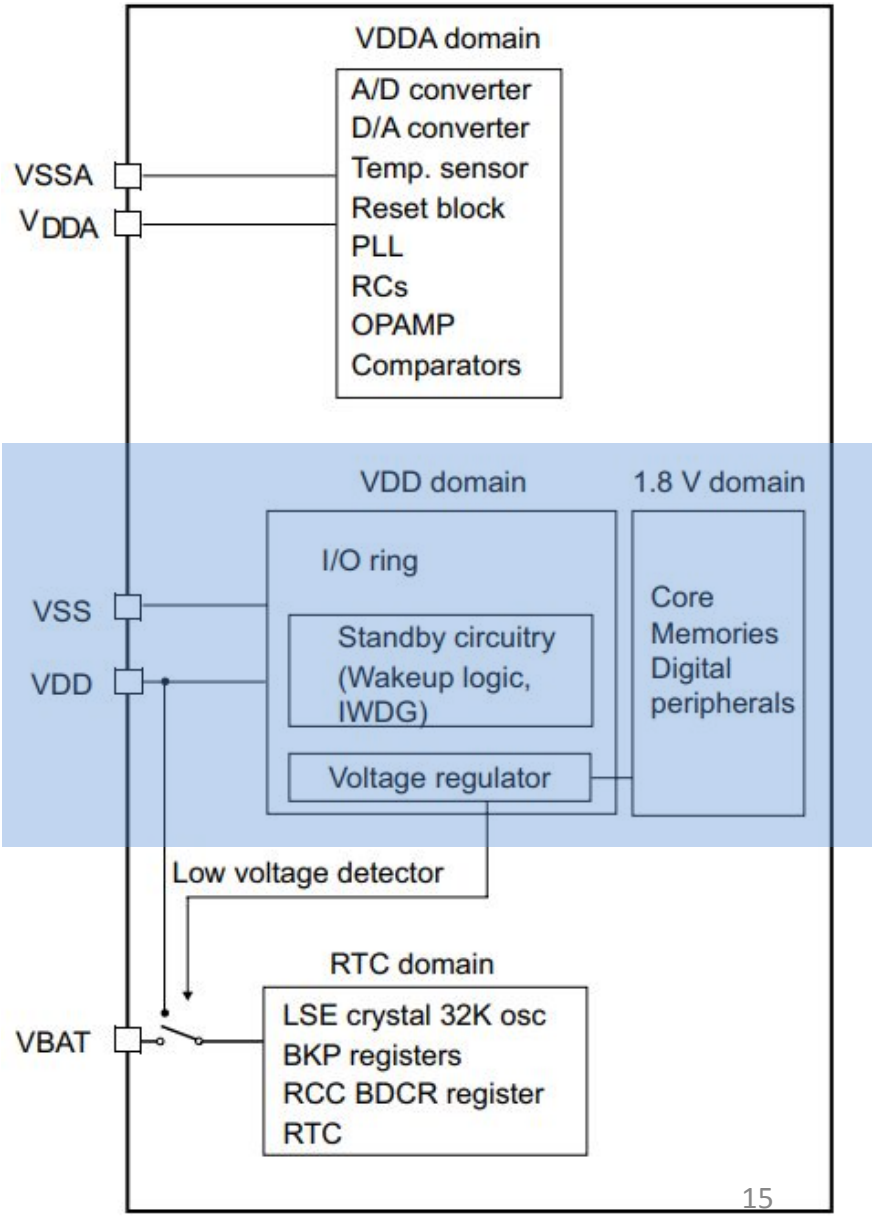
https://tapit.vn/tim-hieu-clock-secuity-system-css-vi-dieu-khien-stm32-qua-vi-du-thuc-te/

NGUỒN CUNG CẤP CHO VI ĐIỀU KHIỂN

TAPiT

❖ VDD/VSS

- Cung cấp nguồn cho VDD domain với điện áp từ 1.8-3.6V
- Điện áp VDD được sử dụng chủ yếu để cung cấp cho các chân I/O
- Khối Voltage Regulator chuyển điện áp VDD thành 1.8V để cung cấp cho 1.8V domain
- Regulator có thể cấu hình được để làm việc ở chế độ tiết kiệm năng lượng



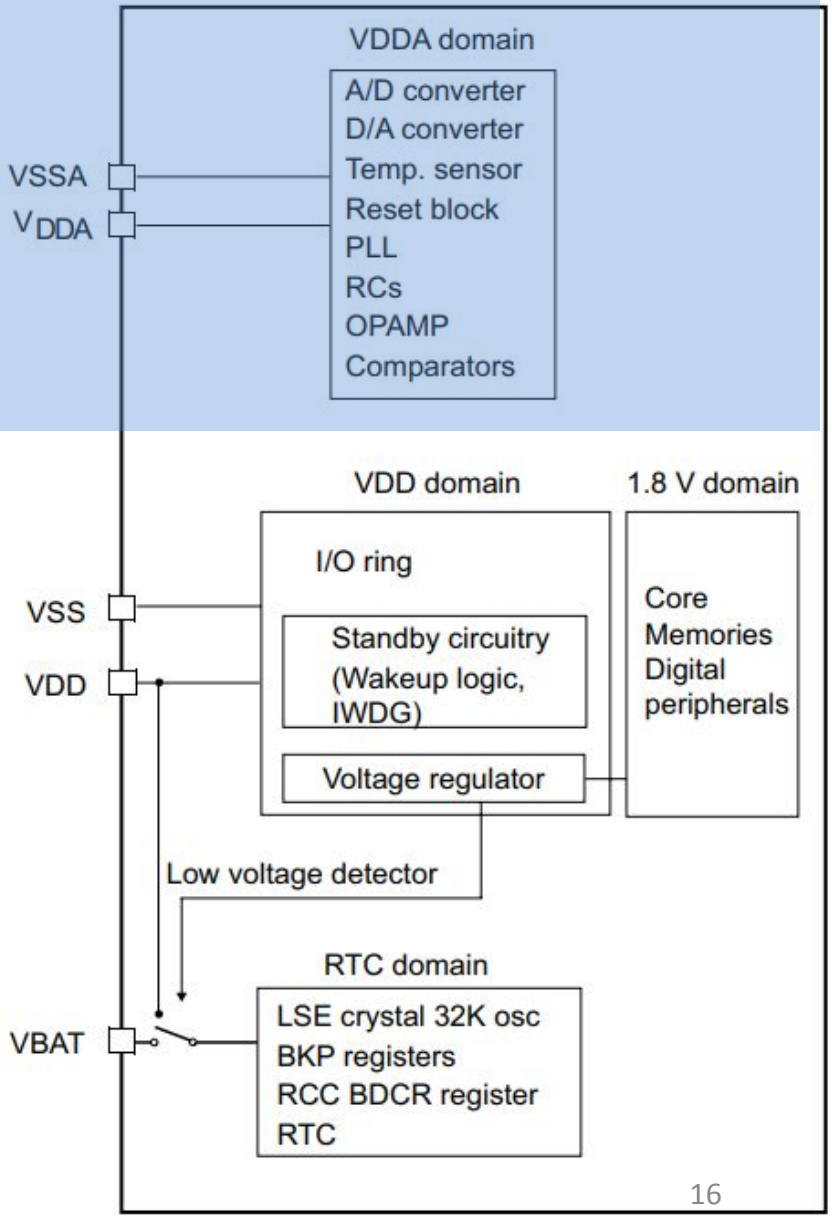
NGUỒN CUNG CẤP CHO VI ĐIỀU KHIỂN

TAPiT

❖ VDDA/VSSA

- Nâng cao độ chính xác hoạt động của khối Analog như chuyển đổi ADC
- Chân cung cấp độc lập VDDA/VSSA để người dùng có thể thiết kế bộ lọc, chống nhiễu riêng
- Lưu ý: Những MCU >64 pins thì mới có VREF+ riêng. ngược lại thì được thiết kế tích hợp vào chân VDDA

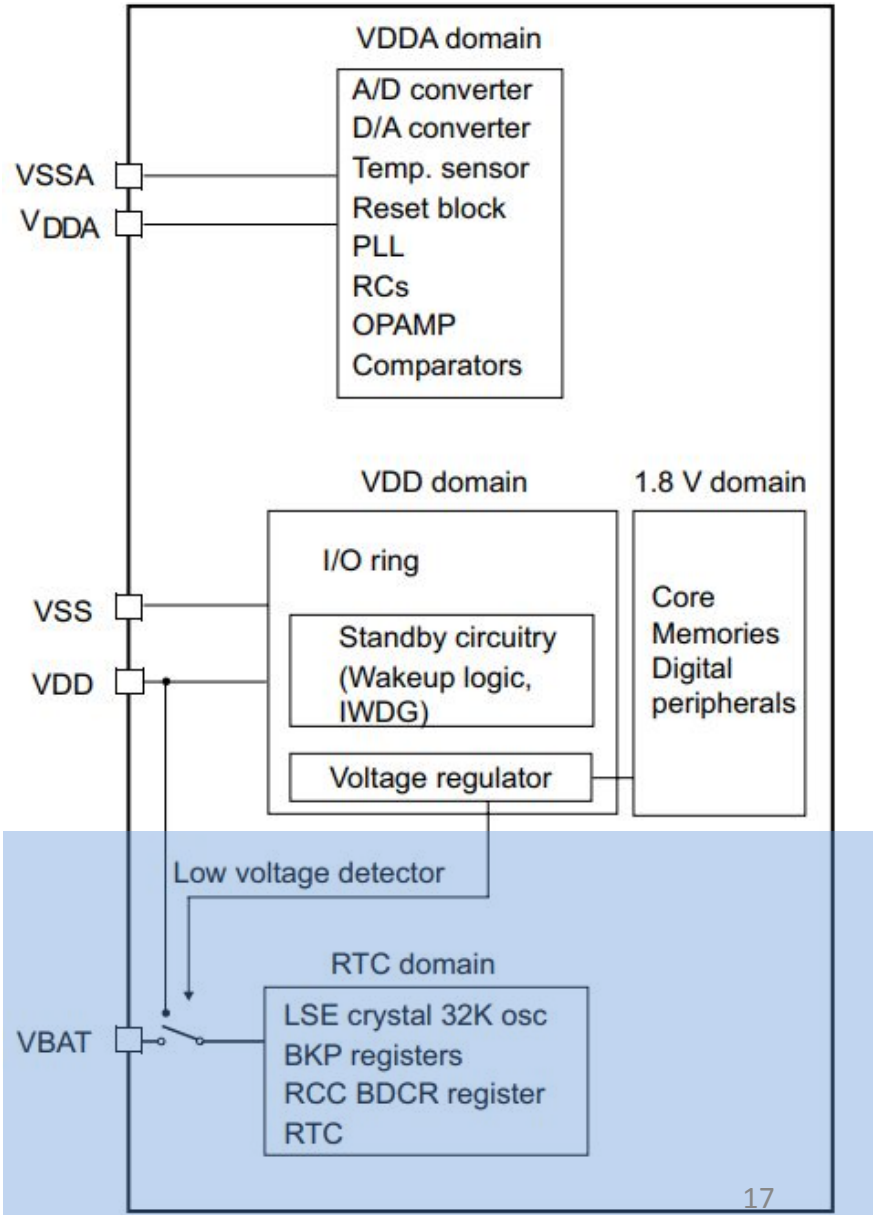
⇒ Ví dụ như VDDA cũng là VREF+, VSSA cũng là VREF-
Thì 10 bit ADC sẽ được sử dụng để lưu trữ các giá trị từ VSSA -> VDDA.
Nhưng nếu có chân VREF+ được thiết kế riêng. Thì 10 bit ADC sẽ được sử dụng để lưu trữ các giá trị từ VSSA (VREF- -> VREF+) >



NGUỒN CUNG CẤP CHO VI ĐIỀU KHIỂN

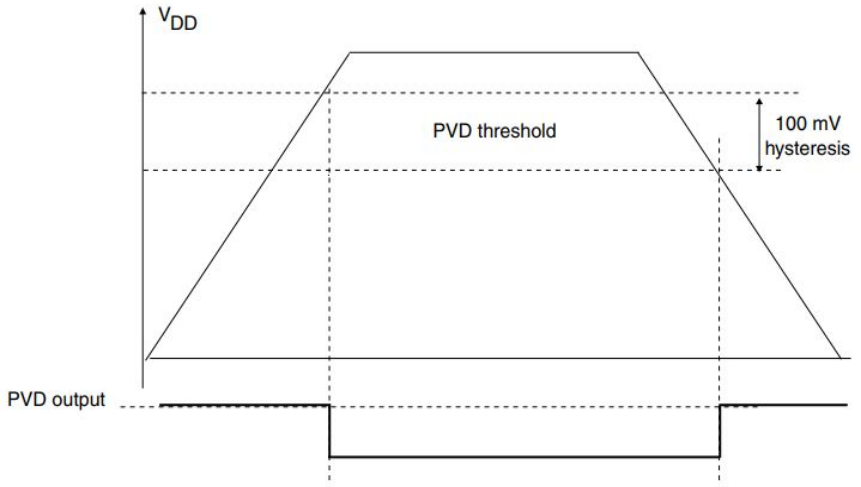
❖ VBAT/VSS

- Duy trì giá trị của các thanh ghi backup và cung cấp nguồn cho khối chức năng RTC
- Khi có nguồn cung cấp VDD thì RTC domain hoạt động nhờ điện áp VDD, khi phát hiện mất nguồn cung cấp VDD thì RTC domain tự động chuyển sang nguồn VBAT



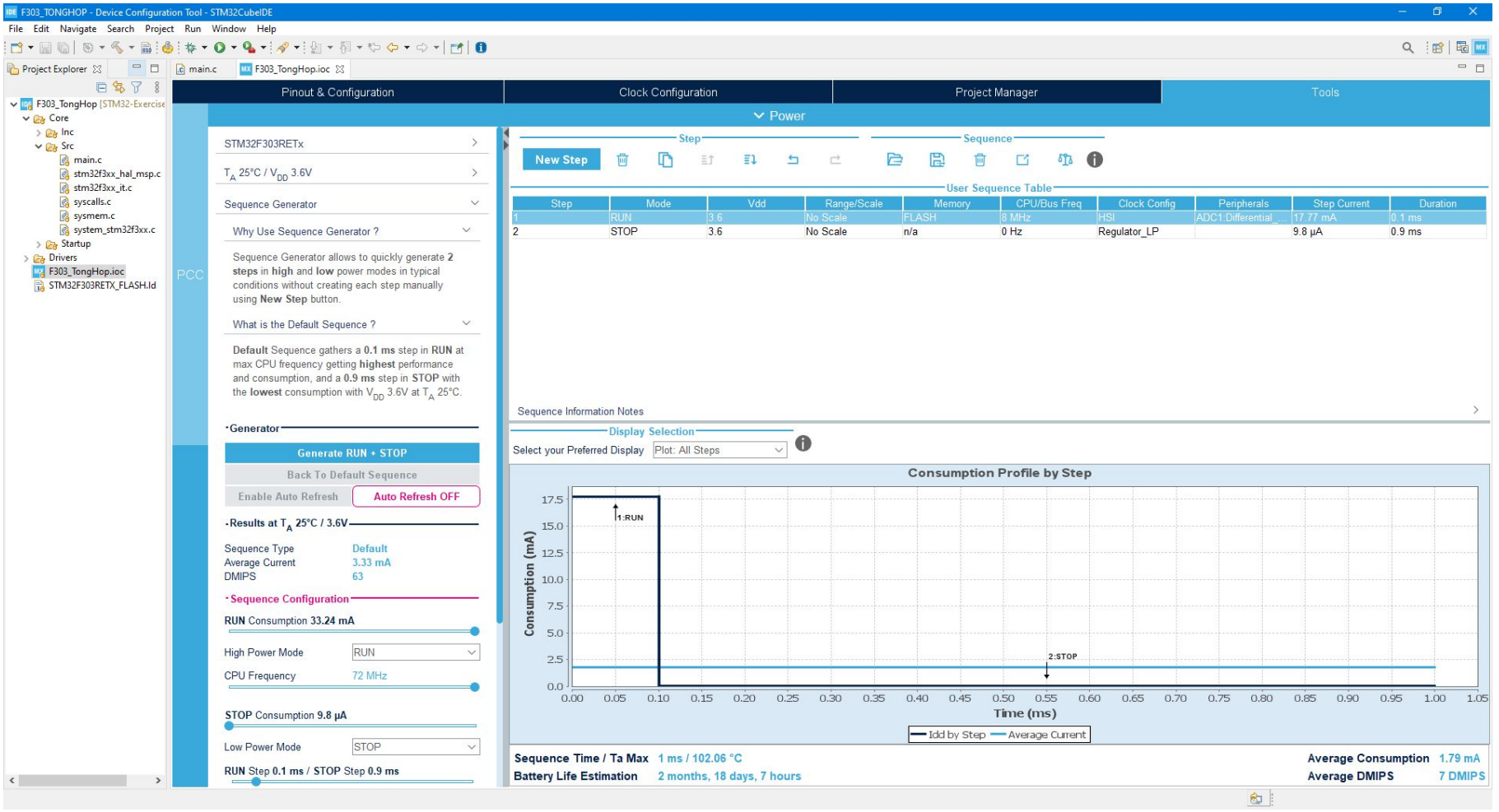
PROGRAMMABLE VOLTAGE DETECTOR (PVD)

- Theo dõi điện áp VDD, so sánh với một ngưỡng PVD
  - ❑ VDD > PVD
  - ❑ VDD < PVD
- Sự kiện phát hiện điện áp vượt ngưỡng được kết nối với EXTI line 16
- Sử dụng Interrupt Handler để đưa ra các thông báo về điện áp hoặc tắt an toàn thiết bị



TÍNH TOÁN ƯỚC LƯỢNG TIÊU THỤ

- STM32CubeIDE -> *.ioc -> Tools

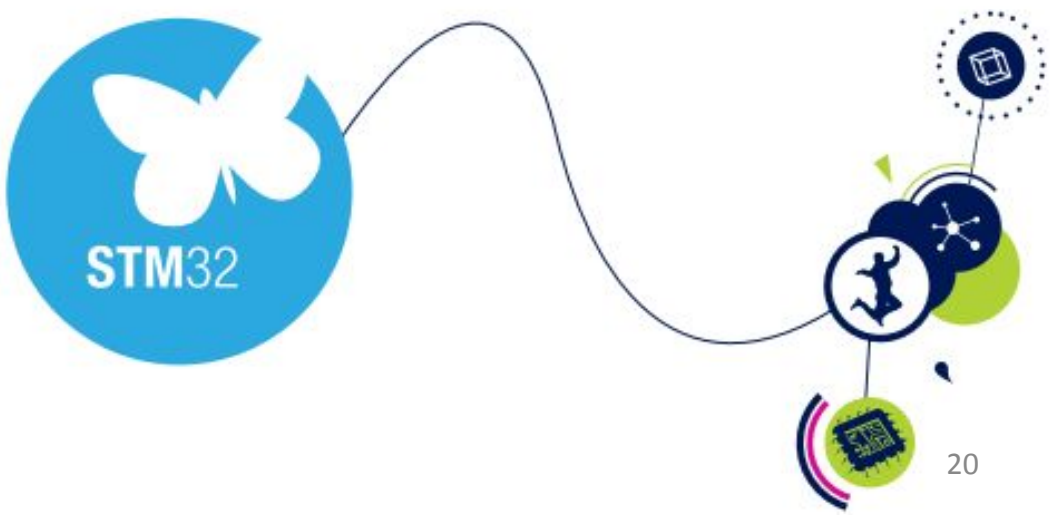


https://www.youtube.com/watch?v=NHgm7dBorRY

TAPIT
BUILD THE FUTURE WITH US!
Learning - Research - Sharing Community

MODULE 4

- CÁC CHẾ ĐỘ TIẾT KIỆM NĂNG LƯỢNG



CÁC VẤN ĐỀ CẦN QUAN TÂM - Mạch

- Các linh kiện điện tử có trong thiết bị (low power)
- Thiết kế của phần mạch nguồn (linear, switching - buck, boost)
- Firmware đảm bảo tối thiểu hóa năng lượng tiêu thụ của toàn bộ thiết bị và của vi điều khiển
- Regulator có thể cấu hình được on/off được.

CÁC TÁC NHÂN ẢNH HƯỞNG ĐẾN NĂNG LƯỢNG TIÊU THỤ CỦA VI ĐIỀU KHIỂN

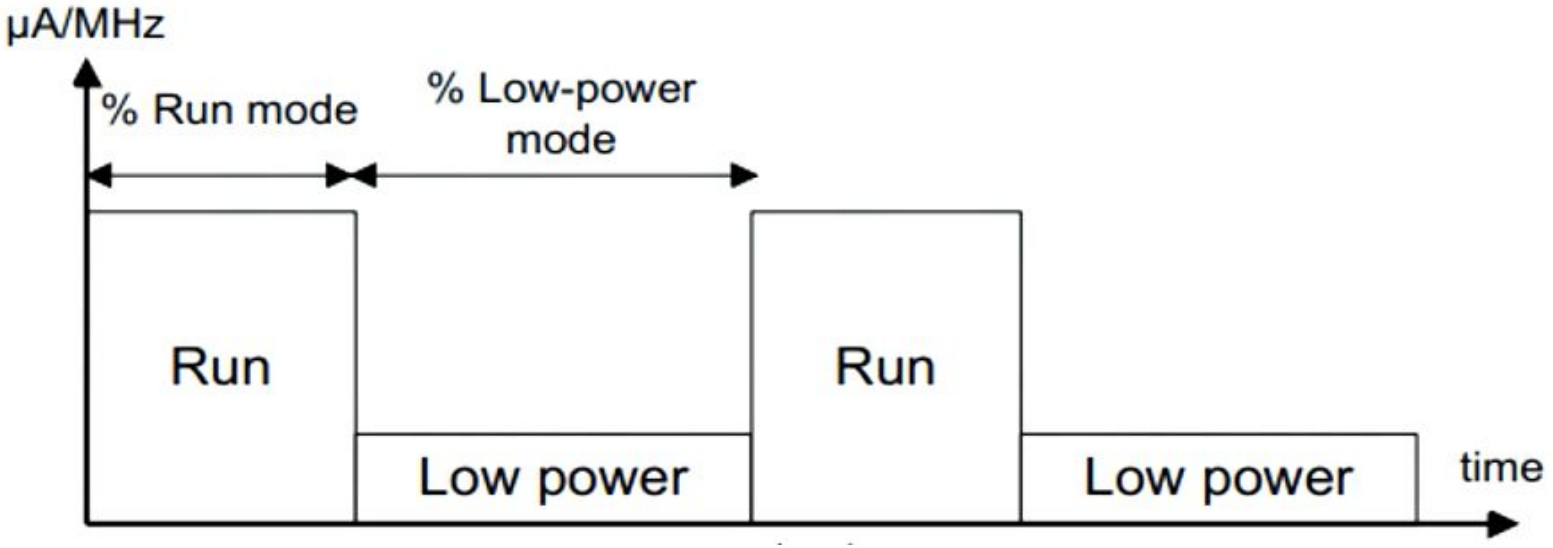
- Tốc độ clock (Cân nhắc, tính toán được giữa tốc độ và hiệu năng)
- Sự phức tạp của một vi điều khiển: Vi điều khiển càng hỗ trợ nhiều ngoại vi và nhiều tính năng thì càng cần nhiều năng lượng, Thiết kế của vi điều khiển
- Sử dụng các chế độ tiết kiệm năng lượng của vi điều khiển

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA VĐK

- RUN mode
- SLEEP mode
- STOP mode
- STANDBY mode

Những vấn đề cần quan tâm:

- Những hạn chế về tính năng (Effect)
- Cách đưa vi điều khiển vào chế độ (Entry)
- Cách thoát ra và thời gian đánh thức (Wakeup)



SLEEP MODE

Effect	Entry	Wakeup
• CPU sẽ bị dừng lại • Tất cả các I/O pin giữ nguyên trạng thái như lúc ở chế độ RUN • Tất cả các ngoại vi đều thức và có thể sinh yêu cầu ngắt • Toàn bộ dữ liệu của các thanh ghi, và RAM không bị ảnh hưởng	• Sử dụng câu lệnh assembly WFI hoặc WFE • Thư viện HAL: HAL_PWR_EnterSLEEPMode(,)	• Nguồn đánh thức: Any Interrupt • Vi xử lý tiếp tục thực hiện chương trình tại vị trí đã sleep • Thời gian chuyển từ chế độ SLEEP sang chế độ RUN là ngay lập tức

❖ Sleep on exit

- Chương trình ứng dụng chỉ hoạt động bên trong Interrupt Handler.
- Vi điều khiển vào chế độ Sleep và thức dậy khi có 1 tín hiệu Interrupt xảy ra, thực thi xong Interrupt Handler thì vi điều khiển vào lại chế độ Sleep

Interrupt driven
https://tapit.vn/cac-ky-thuat-thiet-ke-luong-chuong-trinh-cho-thong-nhung/

STOP MODE

TAPiT

Effect	Entry	Wakeup
• CPU sẽ bị dừng lại • Tất cả các I/O pin giữ nguyên trạng thái như lúc ở chế độ RUN • Tất cả các clock trong miền 1.8v đều bị ngừng • Bộ giao động HSI, HSE, PLL bị disable. • Toàn bộ dữ liệu của các thanh ghi, và RAM không bị ảnh hưởng • Khối voltage regulator có thể cấu hình ở chế độ normal hoặc low power	• Thư viện HAL: HAL_PWR_EnterSTOPMode(,)	• Nguồn đánh thức: Any EXTI line • Vi xử lý tiếp tục thực hiện chương trình tại vị trí đã sleep stop • Nguồn clock của vi điều khiển được đưa về mặc định là HSI • Thời gian chuyển từ chế độ STOP sang chế độ RUN phụ thuộc vào phụ thuộc vào thời gian khởi động HSI, Flash, Votage Regulator

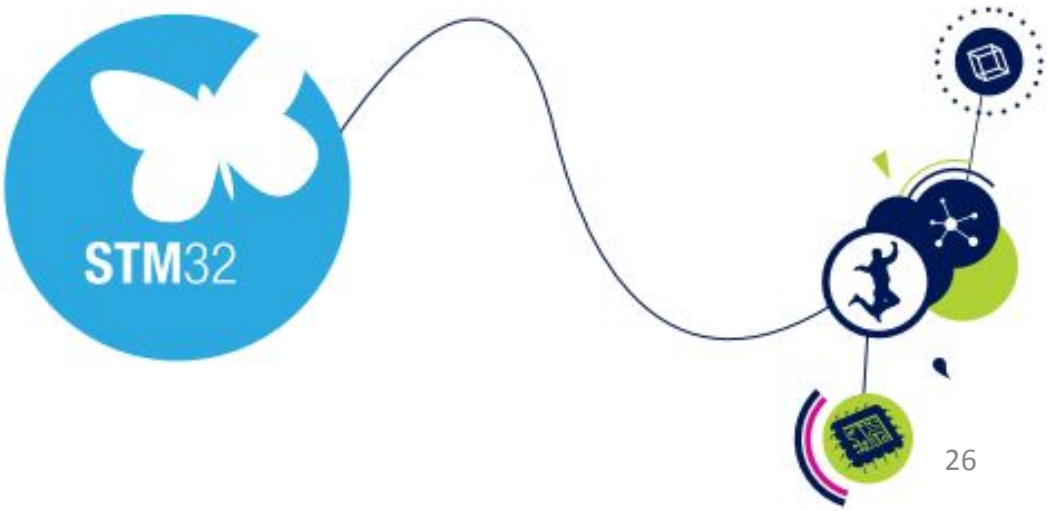
STANDBY MODE

TAPiT

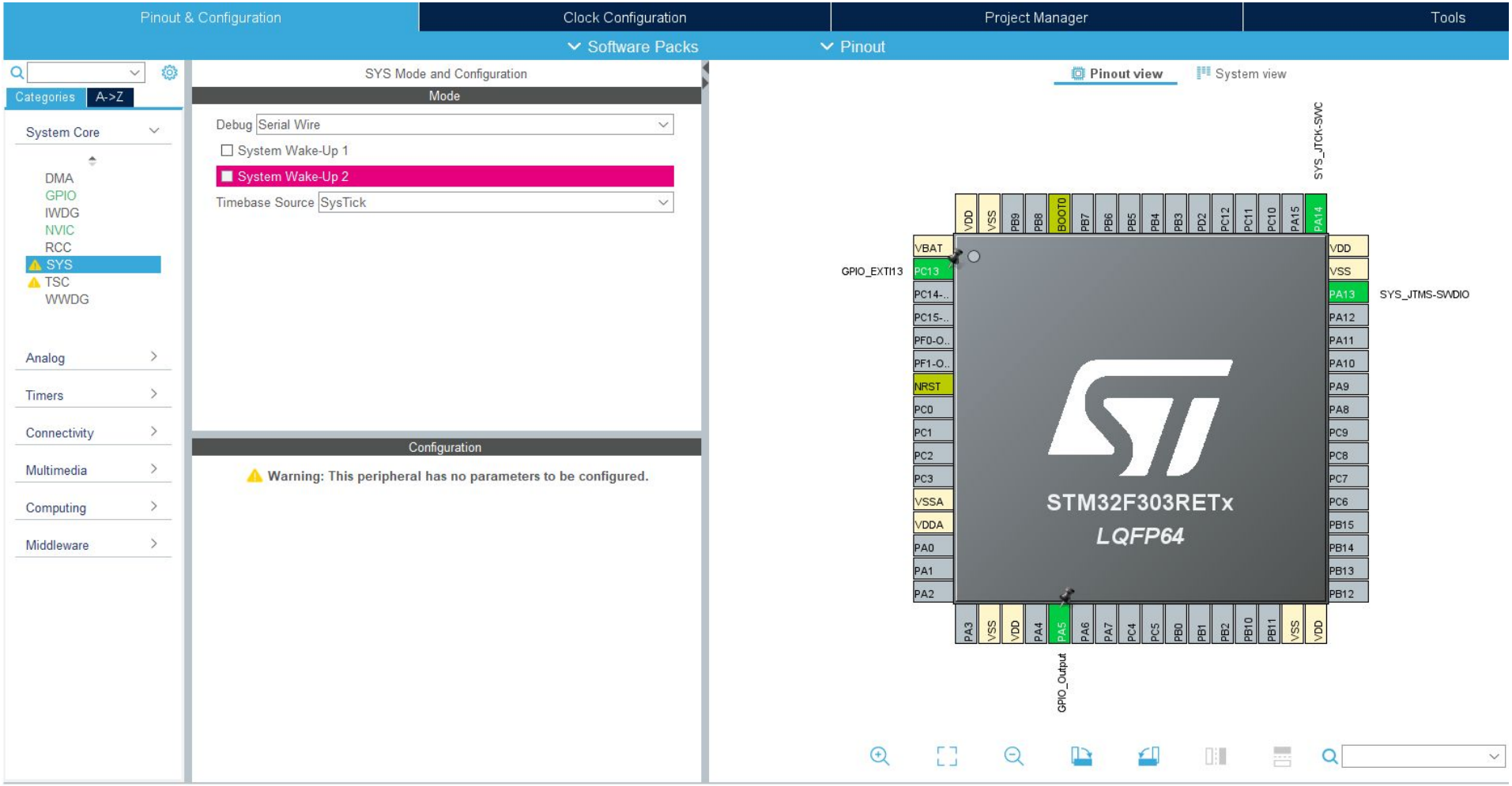
Effect	Entry	Wakeup
• CPU sẽ bị dừng lại • Voltage Regulator sẽ bị tắt • Tất cả các clock trong miền 1.8v đều bị ngừng • Bộ giao động HSI, HSE, PLL bị disable. • Toàn bộ dữ liệu của các thanh ghi, và RAM sẽ bị mất, ngoại trừ một số thanh ghi ở vùng Backup Register • ~~Khối voltage regulator có thể cấu~~ hình ở chế độ normal hoặc low power	• Thư viện HAL: HAL_PWR_EnterSTANDBYMode(,)	• Nguồn đánh thức: - Reset ngoài từ chân NRST pin - IWDG reset - Sườn lên (Rising Edge) nếu sử dụng WKUPx pins • Vi điều khiển sẽ chạy lại từ đầu • Nguồn clock được đưa về mặc định HSI

MODULE 4

- THỰC HÀNH SLEEP MODE



GIAO DIỆN CẤU HÌNH

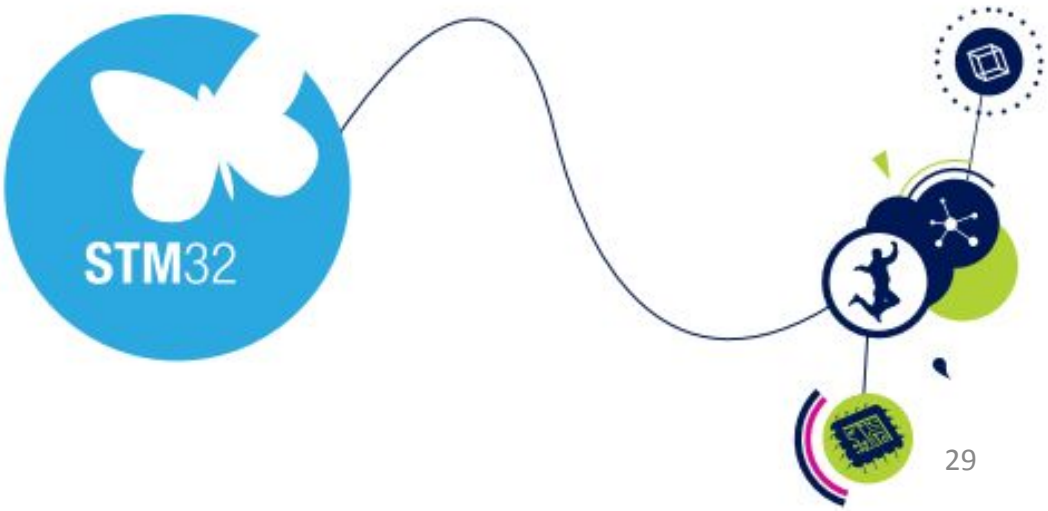


GIAO DIỆN C/C++

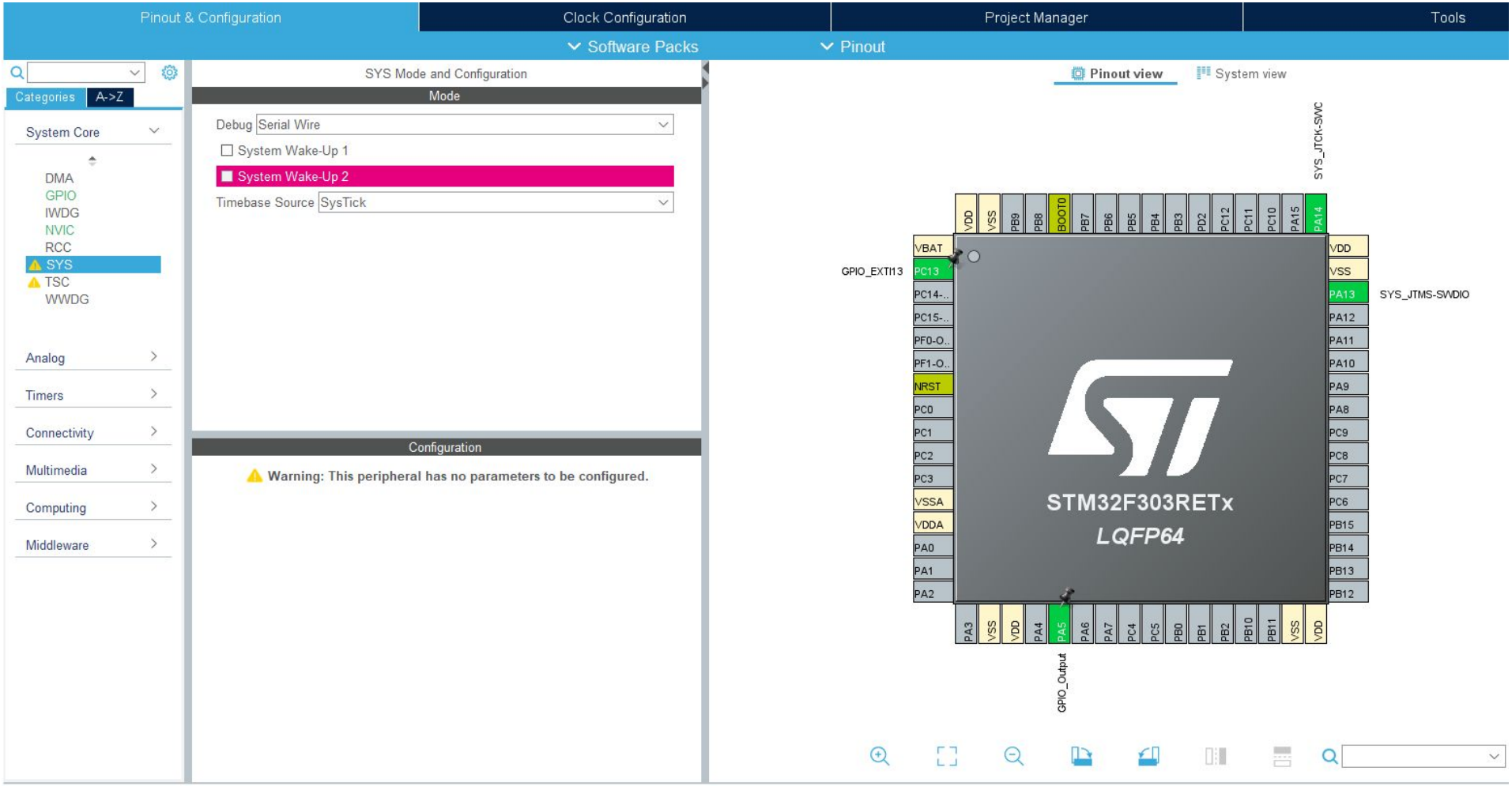
```
/* USER CODE BEGIN WHILE */
 while (1)
 {
        for(uint8_t i = 0; i < 10; i++)
        {
              HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
              HAL_Delay(200);
        }
        HAL_SuspendTick();
        HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);
        HAL_ResumeTick();
         /* USER CODE END WHILE */

 }
```

MODULE 8

- THỰC HÀNH STOP MODE



GIAO DIỆN CẤU HÌNH

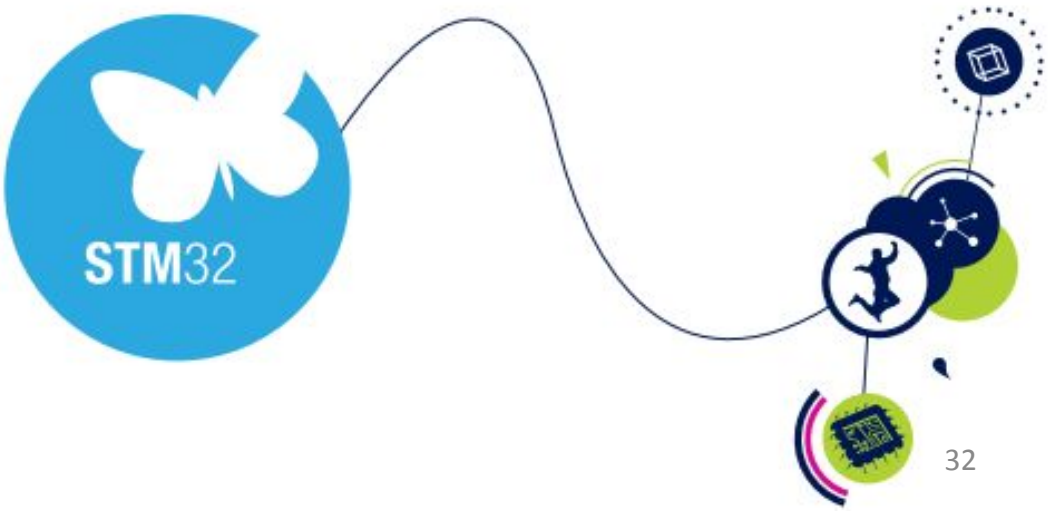


GIAO DIỆN C/C++

```
/* USER CODE BEGIN WHILE */
  while (1)
  {
        for(uint8_t i = 0; i < 5; i++)
        {
              HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
              HAL_Delay(1000);
        }
        HAL_SuspendTick();
        HAL_PWR_EnterSTOPMode(PWR_MAINREGULATOR_ON, PWR_STOPENTRY_WFI);
        HAL_ResumeTick();
         /* USER CODE END WHILE */
  }
```

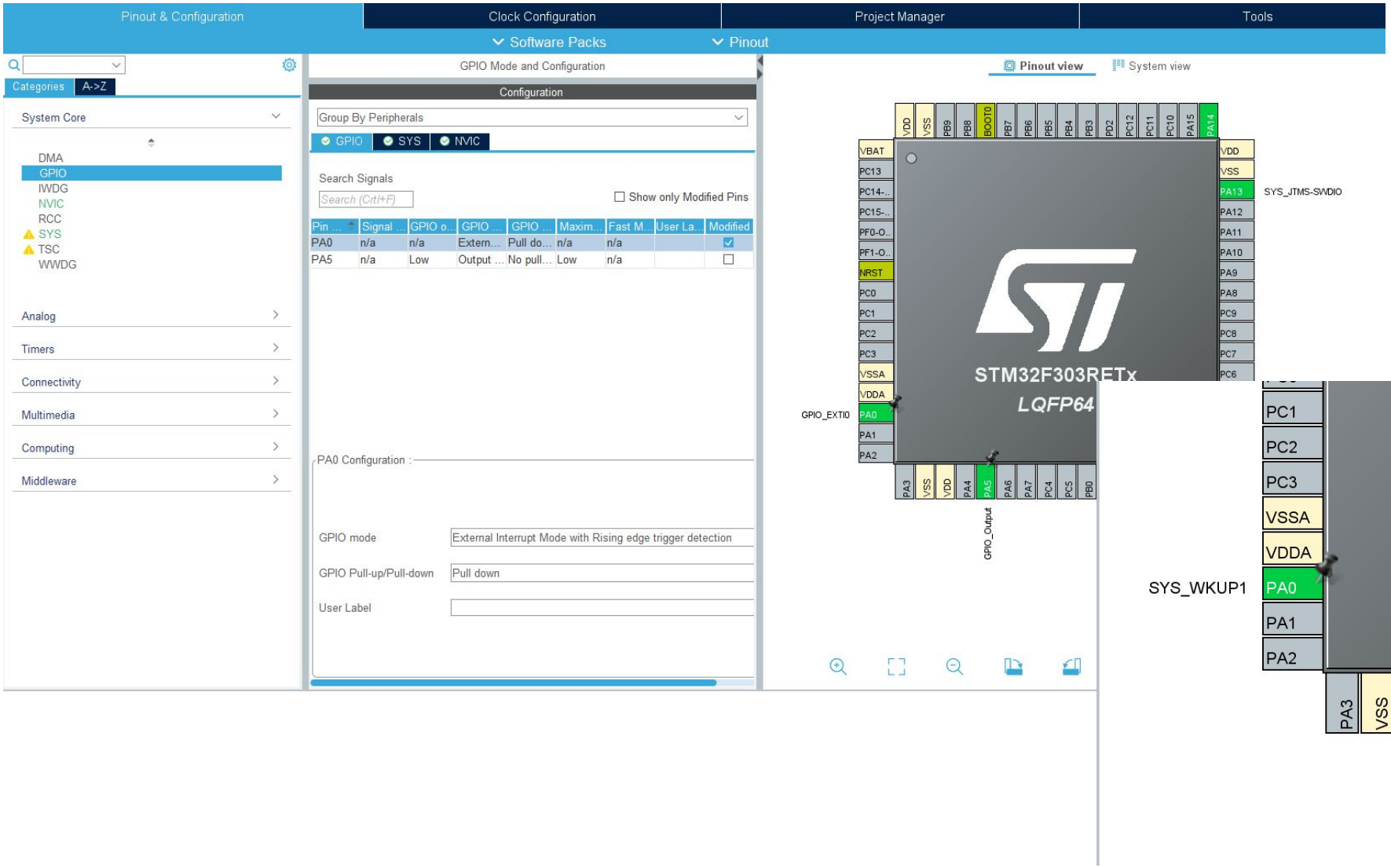
MODULE 8

- THỰC HÀNH STANDBY MODE



Lưu ý:
Mức Logic1 trên chân WKUP sẽ làm đánh thức vi điều khiển khỏi chế độ standby chứ không phải chỉ là Rising Edge.
Khi cấu hình phần cứng, không được dùng trở kéo lên tại chân WKUP (Ví dụ với USER BUTTON NUCLEO F303, có trở kéo lên sẽ làm VDK bị đánh thức liên tục).
— Có thể cấu hình chân WKUP là 1 chân EXTI hoặc 1 chân SYS_WKUPx

GIAO DIỆN CẤU HÌNH



GIAO DIỆN C/C++

```
/* USER CODE BEGIN 2 */
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
 HAL_Delay(3000);
 /* USER CODE END 2 */

/* USER CODE BEGIN WHILE */
 while (1)
 {
      for(uint8_t i = 0; i < 5; i++)
      {
           HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
           HAL_Delay(1000);
      }
      HAL_PWR_DisableWakeUpPin(PWR_WAKEUP_PIN1);
      __HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);
      HAL_PWR_EnableWakeUpPin(PWR_WAKEUP_PIN1);
      HAL_PWR_EnterSTANDBYMode();
  /* USER CODE END WHILE */
}
```

Lưu ý:
Mức Logic1 trên chân WKUP sẽ làm đánh thức vi điều khiển khỏi chế độ standby chứ không phải chỉ là Rising Edge.
Khi cấu hình phần cứng, không được dùng trở kéo lên tại chân WKUP (Ví dụ với USER BUTTON NUCLEO F303, có trở kéo lên sẽ làm VDK bị đánh thức liên tục).
— Có thể cấu hình chân WKUP là 1 chân EXTI hoặc 1 chân SYS_WKUPx

Instructor

Eng. Nguyen Huynh Nhat Thuong