

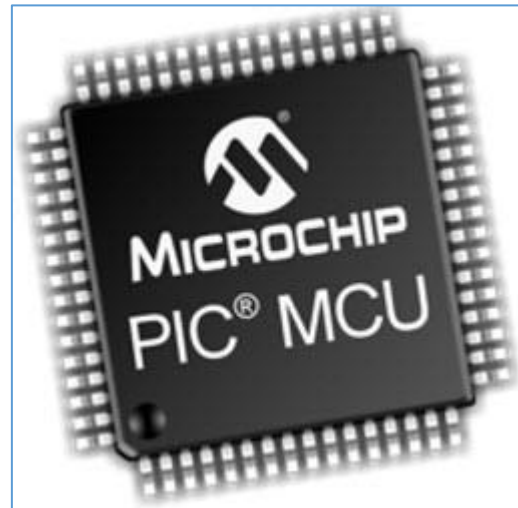
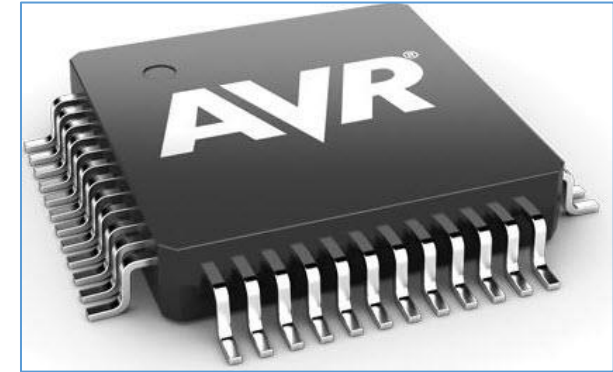
Advanced in Embedded System Programming

Lập trình hệ nhúng nâng cao với
vi điều khiển ARM

Mai Cường Thọ

1. Một số loại vi điều khiển trên thị trường

- Vi điều khiển 8051
- Vi điều khiển AVR
- Vi điều khiển PIC
- Vi điều khiển STM32 (ARM)
- Vi điều khiển MSP
- ...



Main Difference between AVR, ARM, 8051 and PIC Microcontrollers

	8051 ✓	PIC ✓	AVR ✓	ARM ✓
<u>Bus width</u>	8-bit for standard core	8/16/32-bit	8/32-bit	32-bit mostly also available in 64-bit
Communication Protocols	UART, USART,SPI,I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, (special purpose AVR support CAN, USB, Ethernet)	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI (serial audio interface), IrDA
Speed	12 Clock/instruction cycle	4 Clock/instruction cycle	1 clock/ instruction cycle	1 clock/ instruction cycle
Memory	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
ISA	CLSC	Some feature of RISC	RISC	RISC
Memory Architecture	Von Neumann architecture	Harvard architecture	Modified	Modified Harvard architecture
Power Consumption	Average	Low	Low	Low
Families	8051 variants	PIC16,PIC17, PIC18, PIC24, PIC32	Tiny, Atmega, Xmega, special purpose AVR	ARMv4,5,6,7 and series
Community	Vast	Very Good	Very Good	Vast
Manufacturer	NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon, etc.	Microchip Average	Atmel	Apple, Nvidia, Qualcomm, Samsung Electronics, and TI etc.
Cost (as compared to features provide)	Very Low	Average	Average	Low
Other Feature	Known for its Standard	Cheap	Cheap, effective	High speed operation Vast
Popular Microcontrollers	AT89C51, P89v51, etc.	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32, Arduino Community	LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc.

2. Giới thiệu ARM

- ARM (Advanced RISC Machine) là một loại cấu trúc vi xử lý 32 bit và 64 bit kiểu RISC được sử dụng rộng rãi trong các thiết kế nhúng.
- Do có đặc điểm tiết kiệm năng lượng, các bộ CPU ARM chiếm ưu thế trong các sản phẩm điện tử di động, mà với các sản phẩm này việc tiêu tán công suất thấp là một mục tiêu thiết kế quan trọng hàng đầu\
- Ngày nay, hơn 75% CPU nhúng 32-bit là thuộc họ ARM, điều này khiến ARM trở thành cấu trúc 32-bit được sản xuất nhiều nhất trên thế giới.
- Các nhà sản xuất IC đưa ra thị trường hơn 240 dòng vi điều khiển sử dụng lõi ARM .

Các dòng ARM

- Dòng Cortex gồm có 3 phân nhánh chính:
 - dòng A dành cho các ứng dụng cao cấp.
 - dòng R dành cho các ứng dụng thời gian thực.
 - dòng M dành cho các ứng dụng vi điều khiển và chi phí thấp.
- STM32 được thiết kế dựa trên dòng Cortex-M3, dòng CortexM3 được thiết kế đặc biệt để nâng cao hiệu suất hệ thống, kết hợp với tiêu thụ năng lượng thấp
- Cortex-M3 được thiết kế trên nền kiến trúc mới, do đó chi phí sản xuất đủ thấp để cạnh tranh với các dòng vi điều khiển 8 và 16-bit truyền thống
- Cortex-M4

ARM Cortex Processors: Scalability for Every Market

Cortex-A Processors

Rich OS



Highest performance

Cortex-R Processors



RTOS

Fast response / real-time control

Cortex-M Processors

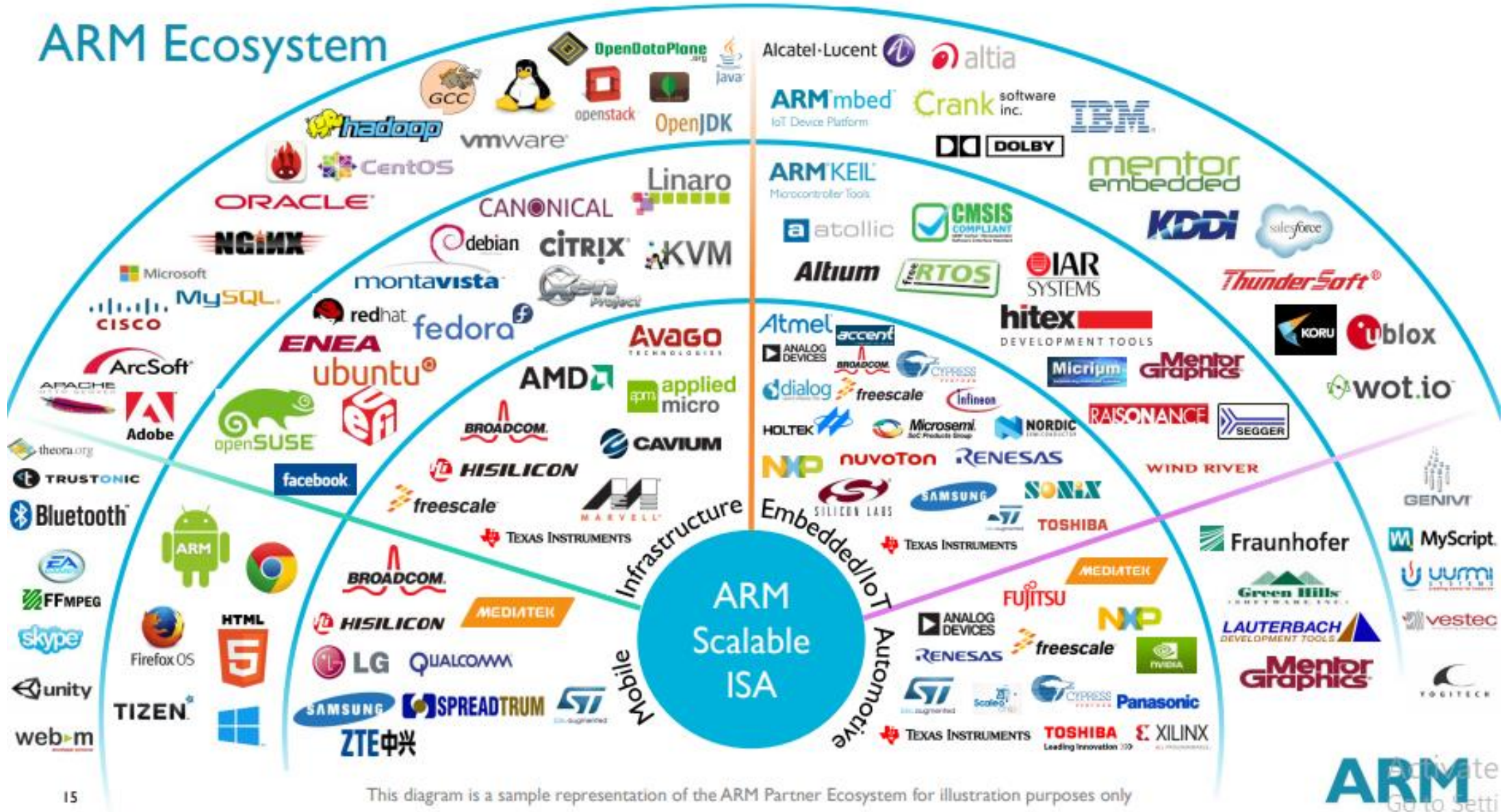
DSP + MCU



Smallest footprint / lowest power



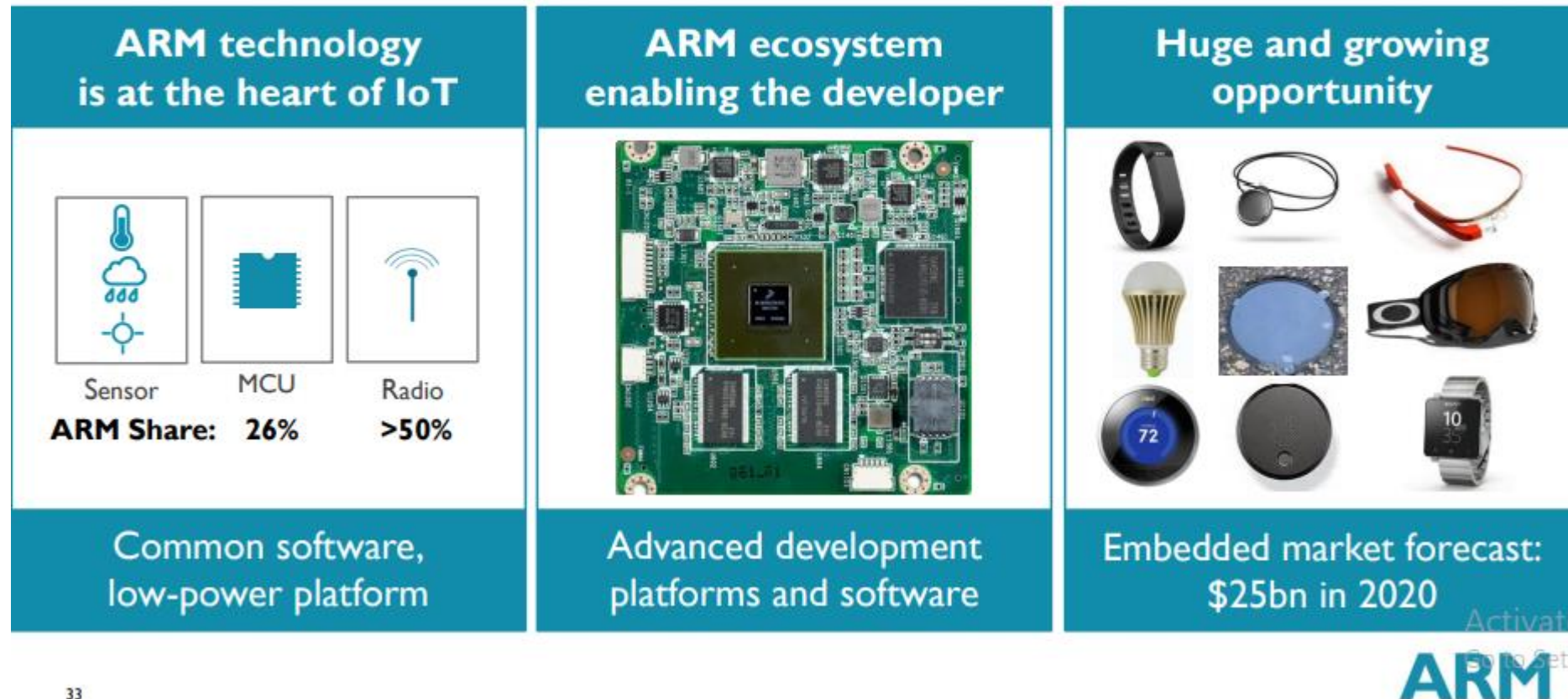
ARM Ecosystem



15
 This diagram is a sample representation of the ARM Partner Ecosystem for illustration purposes only

Nguồn: https://www.arm.com/zh/files/event/1_2015_ARM_Embedded_Seminar_Richard_York.pdf

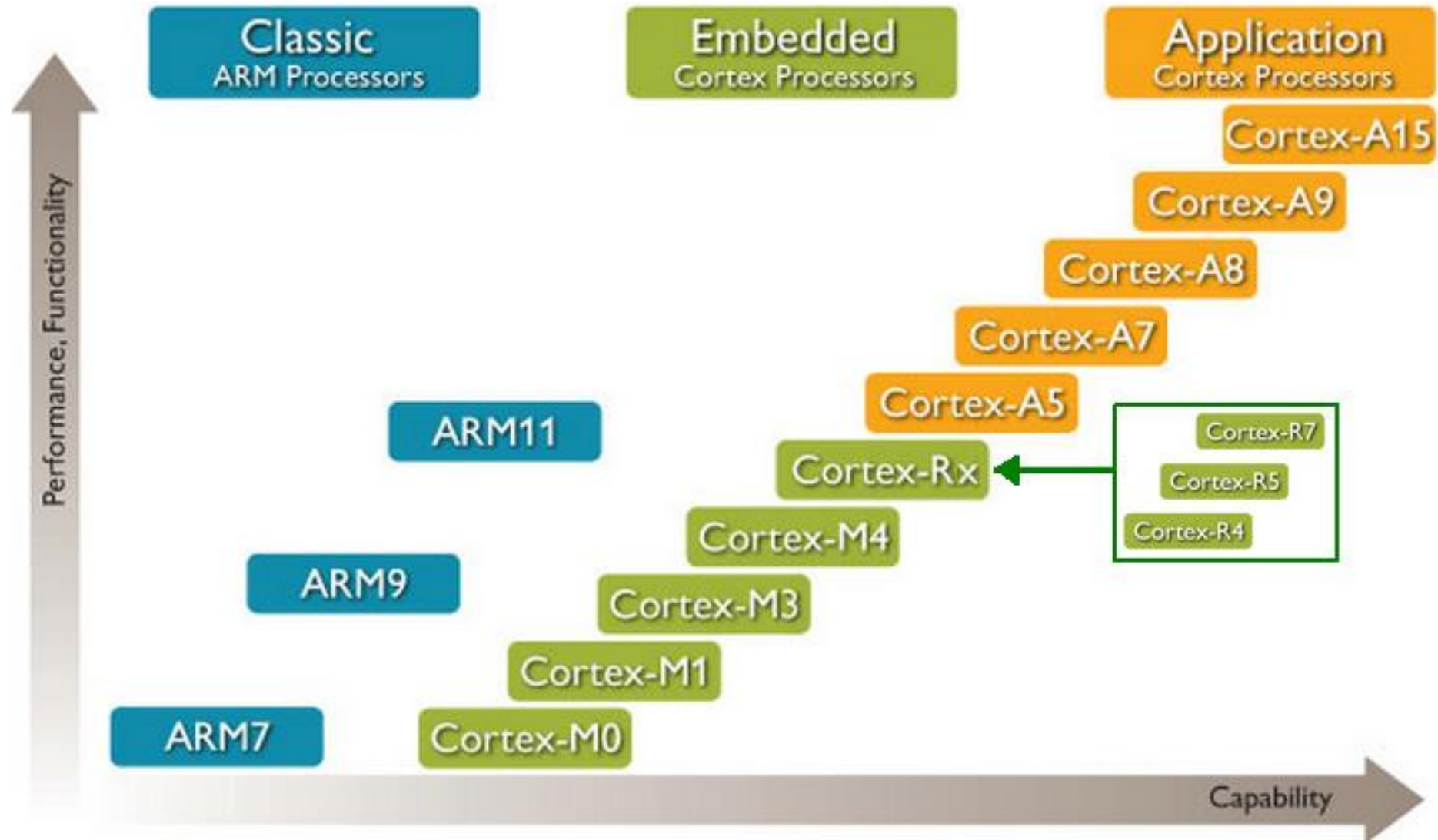
ARM: at the heart of embedded intelligence



33

Nguồn: https://www.arm.com/zh/files/event/1_2015_ARM_Embedded_Seminar_Richard_York.pdf

Cortex Processor roadmap



Architecture Versions and Thumb® ISA

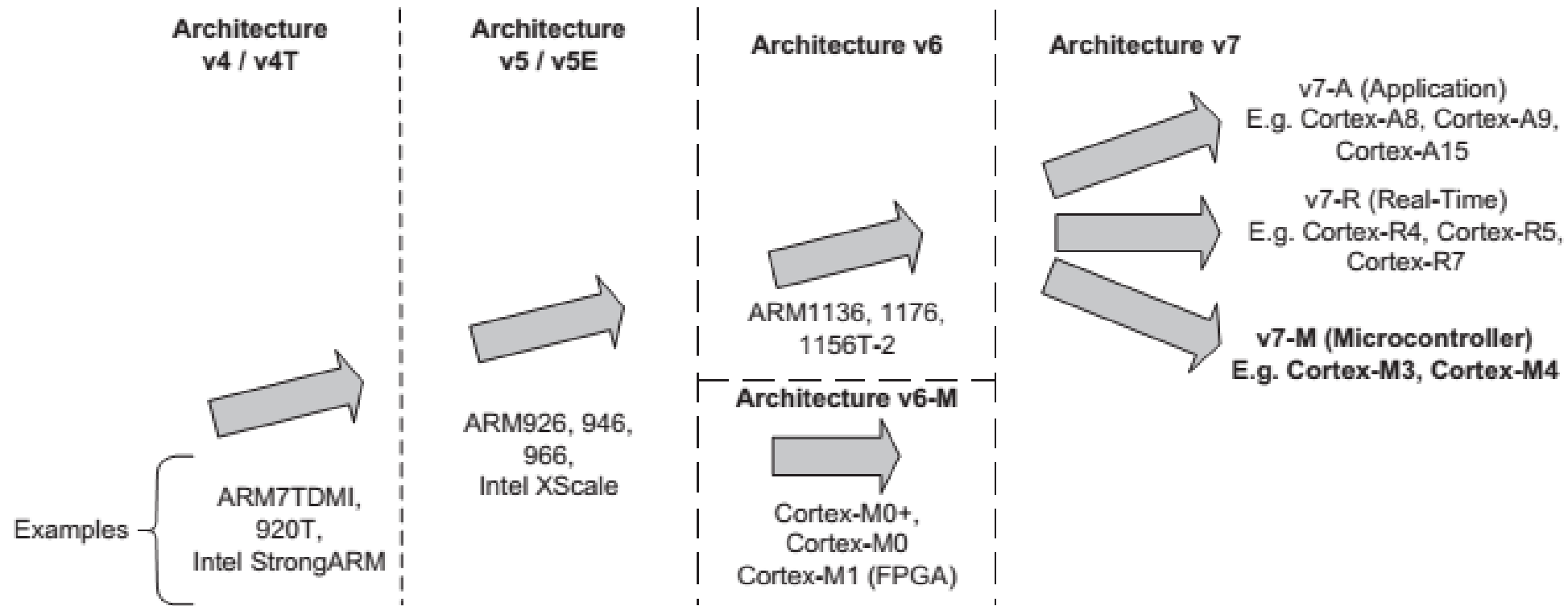


FIGURE 1.7

The evolution of ARM processor architecture

Instruction set

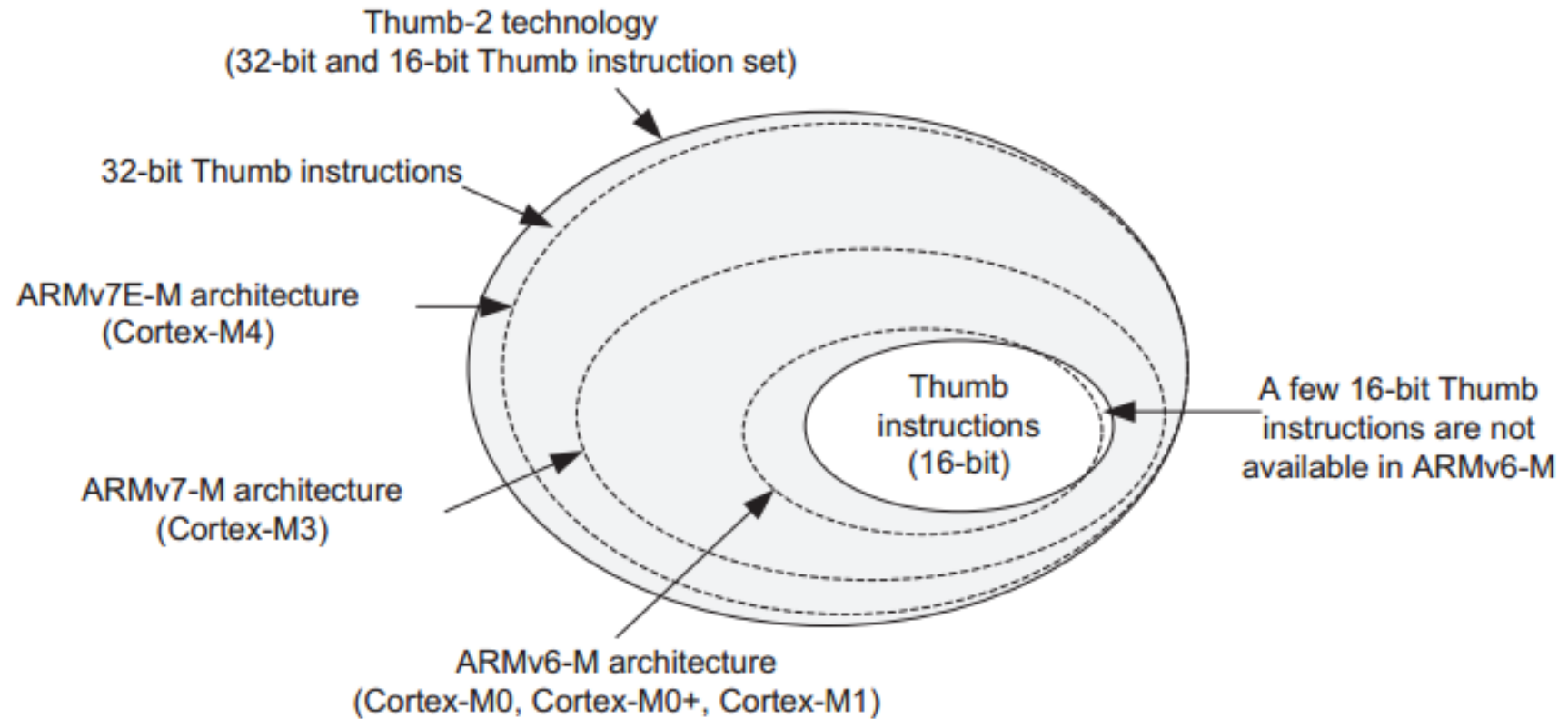
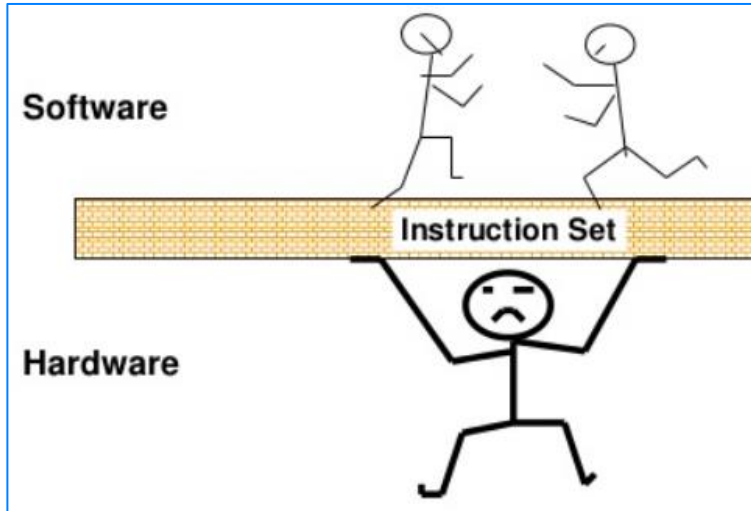
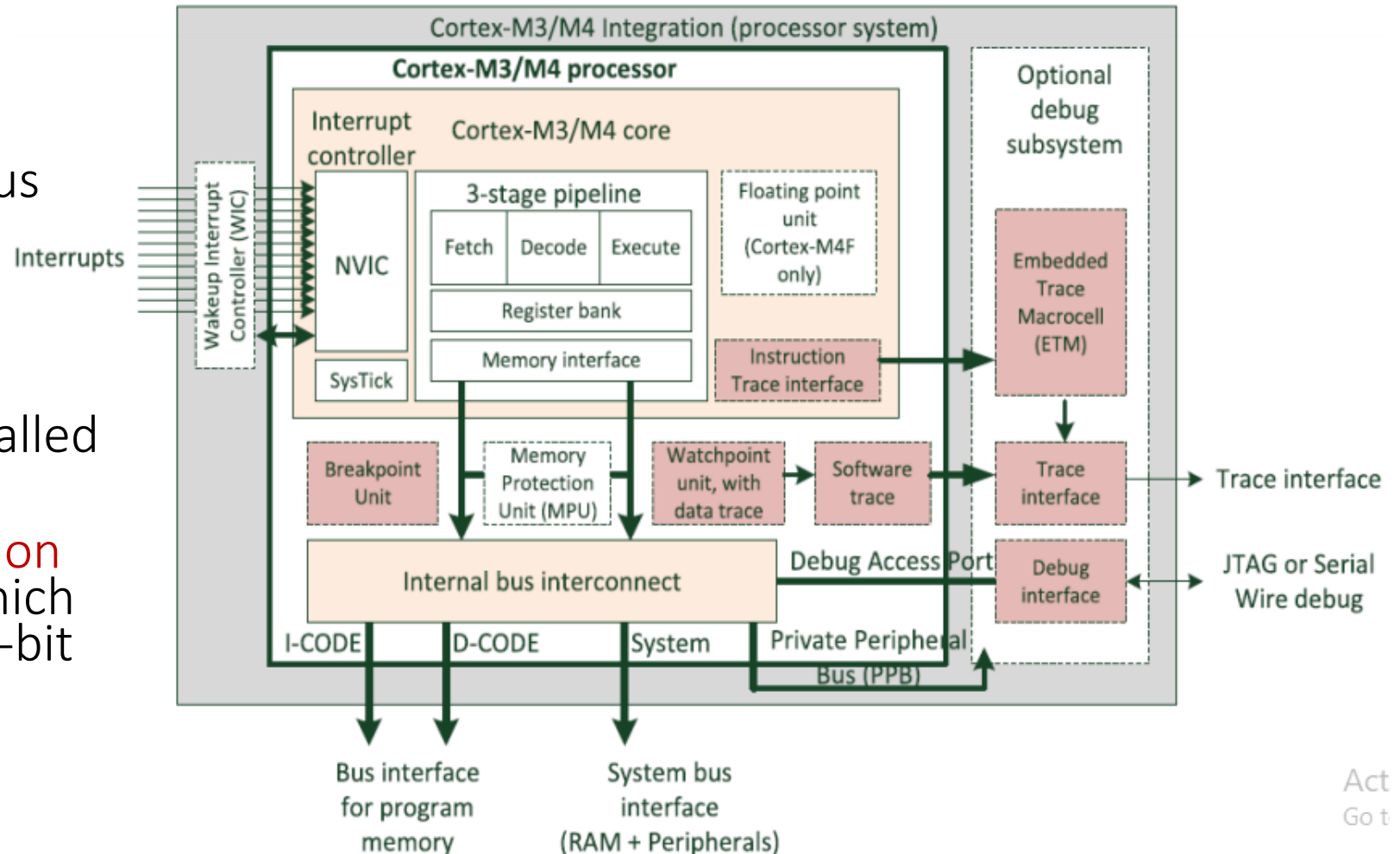


FIGURE 1.9

The relationship between the Thumb instruction set and the instruction set implemented in the Cortex-M processors

3. Kiến trúc ARM Cortex M3, M4

- 32-bit architecture.
- Internal registers in the register bank,
- the data path, and the bus interfaces are all 32 bits wide.
- The Instruction Set Architecture (ISA) in the Cortex-M processors is called the **Thumb ISA**
- Thumb ISA and is **based on Thumb-2 Technology** which supports a mixture of 16-bit and 32-bit instructions.

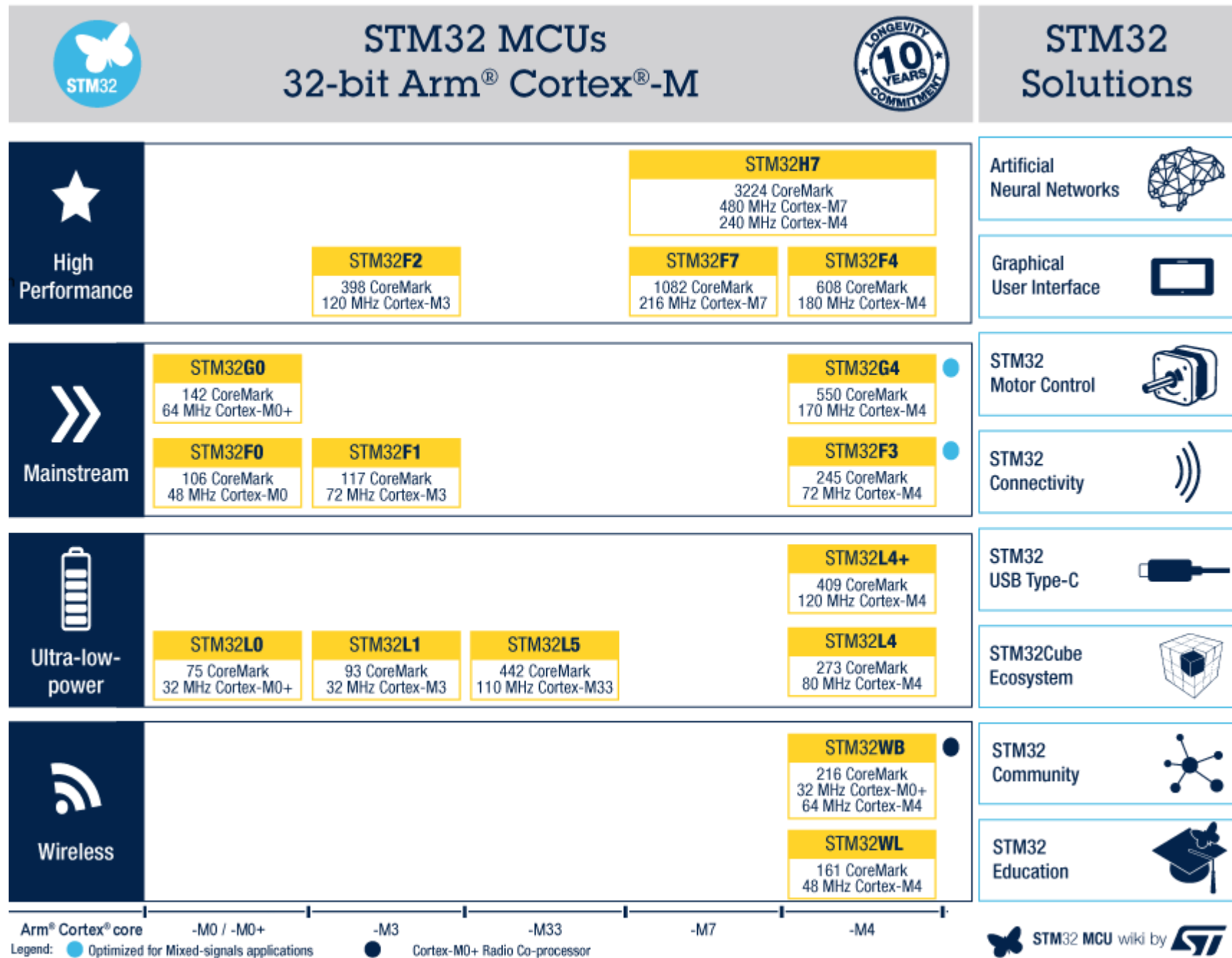


4. Giới thiệu STM32

- Tập đoàn **ST Microelectronic** cho ra mắt dòng **STM32**, vi điều khiển đầu tiên dựa trên nền **lõi ARM Cortex-M3** thế hệ mới do hãng ARM thiết kế, lõi ARM Cortex-M3 là sự cải tiến của lõi ARM7 truyền thống
- Dòng ARM CortexTM-M là thế hệ mới, thiết lập các tiêu chuẩn mới về hiệu suất, chi phí, ứng dụng cho các thiết bị cần tiêu thụ năng lượng thấp, và đáp ứng yêu cầu thời gian thực khắt khe.



4. Giới thiệu STM32 ..



4. Giới thiệu STM32 ..



STM32F1 MCU Series 32-bit Arm® Cortex®-M3 (DSP + FPU) – Up to 72 MHz



- -40 to 105°C range
- USART, SPI, I²C
- 16- and 32-bit timers
- Temperature sensor
- Up to 3x12-bit ADC
- Dual 12-bit ADC
- Low voltage 2.0 to 3.6V (5V tolerant I/Os)

Product line	FCPU (MHz)	Flash (Kbytes)	RAM (Kbytes)	USB 2.0 FS	USB 2.0 FS	FSMC	CAN 2.0B	3-phase MC Timer	PS	SDIO	Ethernet IEEE1588	HDMI CEC
STM32F100 Value line	24	16 to 512	4 to 32			•		•				•
STM32F101	36	16 to 1M	4 to 80			•						
STM32F102	48	16 to 128	4 to 16	•								
STM32F103	72	16 to 1M	4 to 96	•		•	•	•	•	•		
STM32F105 STM32F107	72	64 to 256	64		•	•	•	•	•		•	

4. Giới thiệu STM32 ..





STM32F4 MCU Series 32-bit Arm® Cortex®-M4 – Up to 180 MHz



<ul style="list-style-type: none"> • ART Accelerator™ • SDIO • USART, SPI, I²C • I²S + audio PLL • 16 and 32-bit timers • 12-bit ADC (0.41 µs) • True Random Number Generator • Batch Acquisition Mode • Low voltage 1.7 to 3.6 V • Temperature: -40 °C to 125 °C 	Product lines	F _{CPU} (MHz)	Flash (Kbytes)	RAM (KB)	Ethernet I/F IEEE 1588	2x CAN	Camera I/F	SDRAM I/F	Dual Quad-SPI	SAI	SPDIF RX	Chrom-ART Graphic Accelerator™	TFT LCD Controller	MIPI DSI
	Advanced lines													
	STM32F469 ²	180	512 K to 2056 K	384	•	•	•	•	•	•		•	•	•
	STM32F429 ²	180	512 K to 2056 K	256	•	•	•	•		•		•	•	
	STM32F427 ²	180	1024 K to 2056 K	256	•	•	•	•		•		•		
	Foundation lines													
	STM32F446	180	256 K to 512 K	128		•	•	•	•	•	•			
	STM32F407 ²	168	512 K to 1024 K	192	•	•	•							
	STM32F405 ²	168	512 K to 1024 K	192		•								
	Product lines	F _{CPU} (MHz)	Flash (Kbytes)	RAM (KB)	RUN current (µA/MHz)	STOP current (µA)	Small package (mm)	FSMC (NOR/PSRAM/LCD support)	QSPI	DFSDM	DAC	TRNG	DMA Batch Acquisition Mode	USB 2.0 OTG FS

4. Giới thiệu STM32 ..

<div>  <div> STM32F4 MCU Series 32-bit Arm® Cortex®-M4 – Up to 180 MHz </div>  </div>														
<ul style="list-style-type: none"> • Batch Acquisition Mode • Low voltage 1.7 to 3.6 V • Temperature: • -40 °C to 125 °C 	Product lines	F _{CPU} (MHz)	Flash (Kbytes)	RAM (KB)	RUN current (µA/MHz)	STOP current (µA)	Small package (mm)	FSMC (NOR/PSRAM/LCD support)	QSPI	DFSDM	DAC	TRNG	DMA Batch Acquisition Mode	USB 2.0 OTG FS
	Access lines													
	STM32F401	84	128 K to 512 K	up to 96	Down to 128	Down to 10	Down to 3x3							•
	STM32F410	100	64 K to 128 K	32	Down to 89	Down to 6	Down to 2.553x 2.579				•	•	BAM	-
	STM32F411	100	256 K to 512 K	128	Down to 100	Down to 12	Down to 3.034x 3.22						BAM	•
	STM32F412	100	512 K to 1024 K	256	Down to 112	Down to 18	Down to 3.653x 3.651	•	•	•		•	BAM	• +LPM ¹
	STM32F413 ²	100	1024 K to 1536 K	320	Down to 115	Down to 18	Down to 3.951x 4.039	•	•	•	•	•	BAM	• +LPM ¹

Notes:

1. Link Power Management

2. The same devices are also found with embedded HW AES encryption (128-/256-bit)

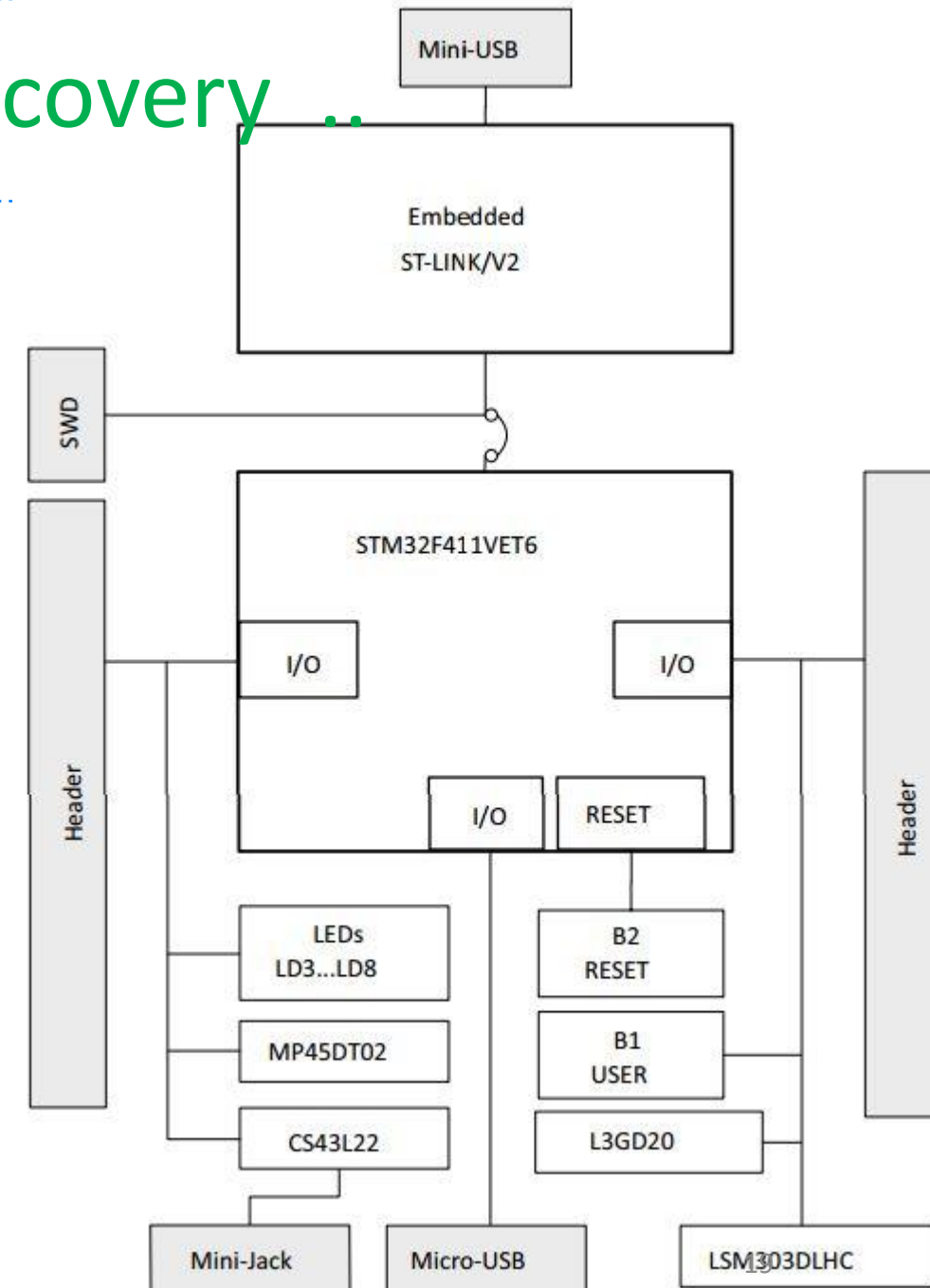
5. Kit phát triển STM32F411 discovery

- Sử dụng MCU
STM32F411VET6 32-bit lõi
ARM Cortex-M4F.
- 1 MB Flash, 192 KB RAM
và có 100 chân (LQFP100
package)
- Tích hợp sẵn mạch nạp
STLINK/V2 trên board
- Nguồn sử dụng từ USB hoặc
nguồn ngoài 5V



5. Kit phát triển STM32F411 discovery ...

- External application power supply: 3 V and 5 V
- **L3GD20**: ST MEMS **motion sensor** 3-axis digital output gyroscope – *Cảm biến chuyển động*
- **LSM303DLHC**: ST MEMS system-in-package featuring a 3D digital linear **acceleration sensor** and a 3D digital **magnetic sensor** – *Cảm biến gia tốc*
- **MP45DT02**: ST MEMS **audio sensor**, omnidirectional digital **microphone** - *Cảm biến âm thanh*
- **CS43L22**, audio DAC with integrated class D speaker drive – *Cửa xuất âm thanh*
- Extension header for LQFP100 I/Os for a quick connection to the prototyping board and an easy probing
- Comprehensive free software including a variety of examples, part of STM32CubeF4 package or STSW-STM32136 for legacy Standard Libraries usage

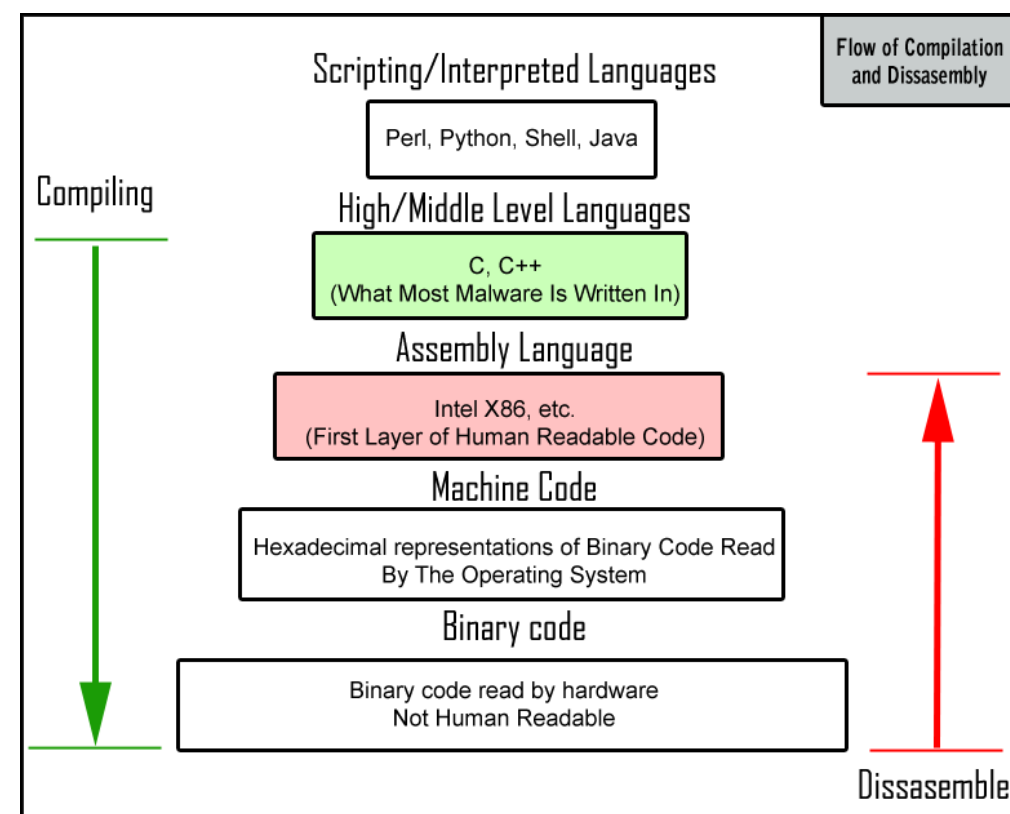
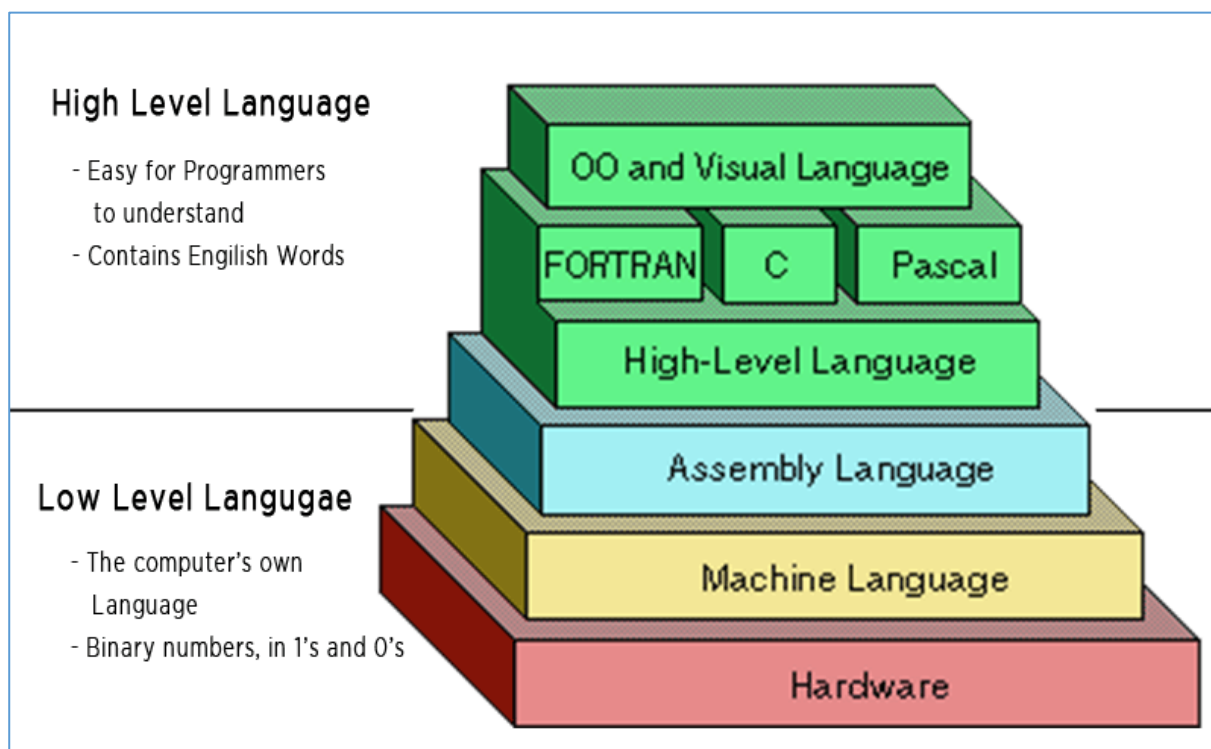


6. Lập trình HTN

- *Lập trình ứng dụng trên hệ nhúng phụ thuộc vào*
 - *nền tảng (platform) phần cứng,*
 - *phần mềm của hệ nhúng đó.*
- **Hệ nhúng không có hệ điều hành:**
 - Thường sử dụng các vi điều khiển hiệu năng tương đối thấp (8051, ATmega, PIC, ARM7, ...)
 - **Lập trình bằng C, ASM**
 - Môi trường, công cụ lập trình tùy theo từng dòng vi điều khiển (CodeVision, AVR Studio, Keil C...)
 - Phù hợp các ứng dụng điều khiển vào/ra cơ bản, các giao tiếp ngoại vi cơ bản.

6. Lập trình HTN ...

- Sử dụng Ngôn ngữ lập trình bậc thấp: Lập trình hợp ngữ
- Sử dụng các ngôn ngữ lập trình bậc cao: **C**, C++, Java, Python



6.1. Lập trình hợp ngữ cho ARM-Cortex M

- Lập trình mức gần mã máy nhất
- Sử dụng ký hiệu gợi nhớ
- Thao tác trực tiếp lên các thanh ghi lập trình được của CPU
- Ưu điểm: Vì ngôn ngữ Assembler rất gần gũi với ngôn ngữ máy nên chương trình
 - Chạy nhanh.
 - Tiết kiệm bộ nhớ.
- Nhược điểm:
 - Khó viết bởi vì yêu cầu người lập trình rất am hiểu về phần cứng.
 - Khó tìm sai: bao gồm sai về cú pháp (syntax) và sai về thuật toán (Algorithm).
 - Không chuyển chương trình Assembler cho các máy tính có cấu trúc khác nhau.

6.1. Lập trình hợp ngữ cho ARM-Cortex M ..

Ví dụ

- Ngôn ngữ C

$x = (a + b) - c$

- ARM Assembly

ADR r4, a

; lấy địa chỉ của **biến a**, cất vào thanh ghi **r4**

LDR r0, [r4]

; lấy giá trị của **a**, ở ngăn nhớ có địa chỉ **r4**, cất vào thanh ghi **r0**

ADR r4, b

; lấy địa chỉ của **b**, cất vào thanh ghi **r4** (sử dụng lại)

LDR r1, [r4]

; lấy giá trị của **a**, cất vào thanh ghi **r1**

ADD r3, r0, r1

; tính tổng a+b, cất vào thanh ghi **r3**

ADR r4, c

; lấy địa chỉ của **c**, cất vào thanh ghi **r4**

LDR r2, [r4]

; lấy giá trị của **c**, cất vào thanh ghi **r1**

SUB r3, r3, r2

; r3=r3-r2, hoàn thiện tính **x**

ADR r4, x

; lấy địa chỉ của **x**, cất vào thanh ghi **r4**

STR r3, [r4]

; lưu giá trị trong **r3**, vào ngăn nhớ có địa chỉ **r4** (store value of x)

7. The Cortex microcontroller software interface standard (CMSIS)

- Chuẩn giao diện phần mềm cho vi điều khiển Cortex (hay thư viện CMSIS)

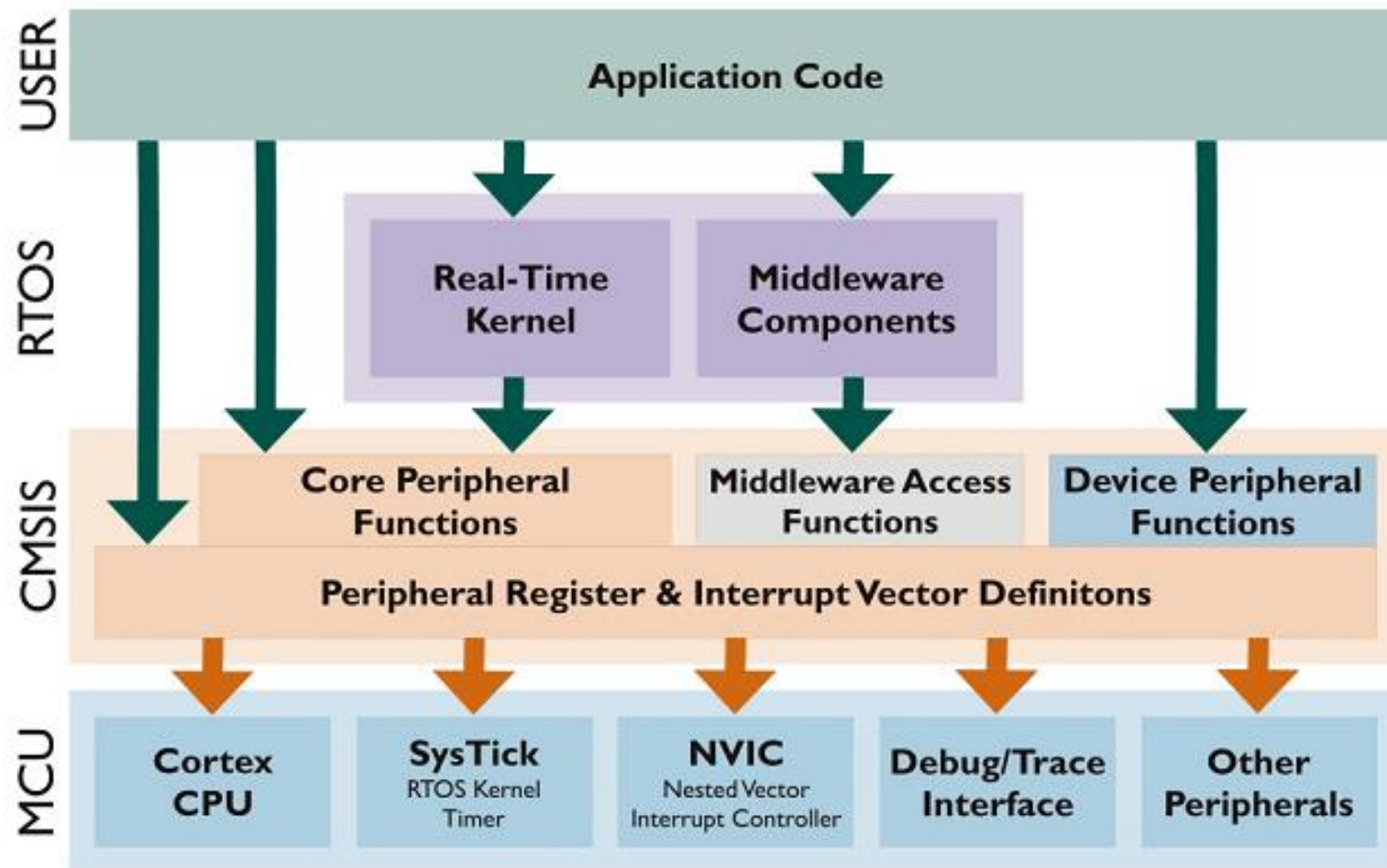


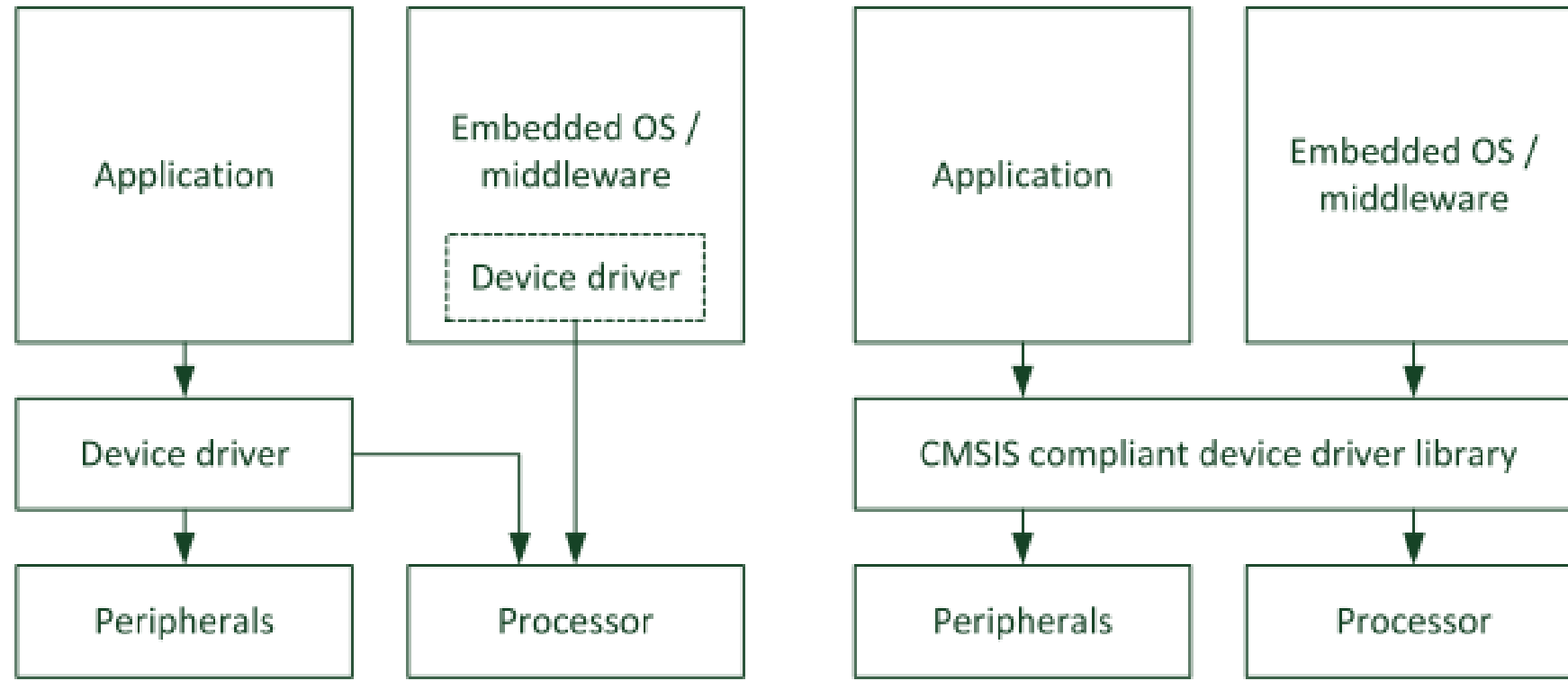
Figure 2 CMSIS Structure functional flow

7.1. CMSIS - why?

- CMSIS được phát triển bởi ARM
 - to allow microcontroller and software vendors to use **a consistent software infrastructure to develop software solutions for Cortex -M** microcontrollers
 - Currently the Cortex-M microcontroller market comprises
 - More than 15 microcontroller vendors shipping Cortex-M microcontroller products
 - More than 10 toolchain vendors
 - More than 30 embedded operating systems
 - Additional Cortex-M middleware software providers for codecs, communication protocol stacks, etc.

7.2 CMSIS - the aims?

- **Enhanced software reusability**
 - makes it easier to reuse software code in different Cortex-M projects, reducing time to market and verification efforts.
- **Enhanced software compatibility**
 - by having a consistent software infrastructure (e.g., API for processor core access functions, system initialization method, common style for defining peripherals), software from various sources can work together, reducing the risk in integration.
- **Easy to learn**
 - the CMSIS allows easy access to processor core features from the C language. In addition, once you learn to use one Cortex-M microcontroller product, starting to use another Cortex-M product is much easier because of the consistency in software setup.
- **Toolchain independent**
 - CMSIS-compliant device drivers can be used with various compilation tools, providing much greater freedom.
- **Openness**
 - the source code for CMSIS core files can be downloaded and accessed by everyone, and everyone can develop software products with CMSIS.



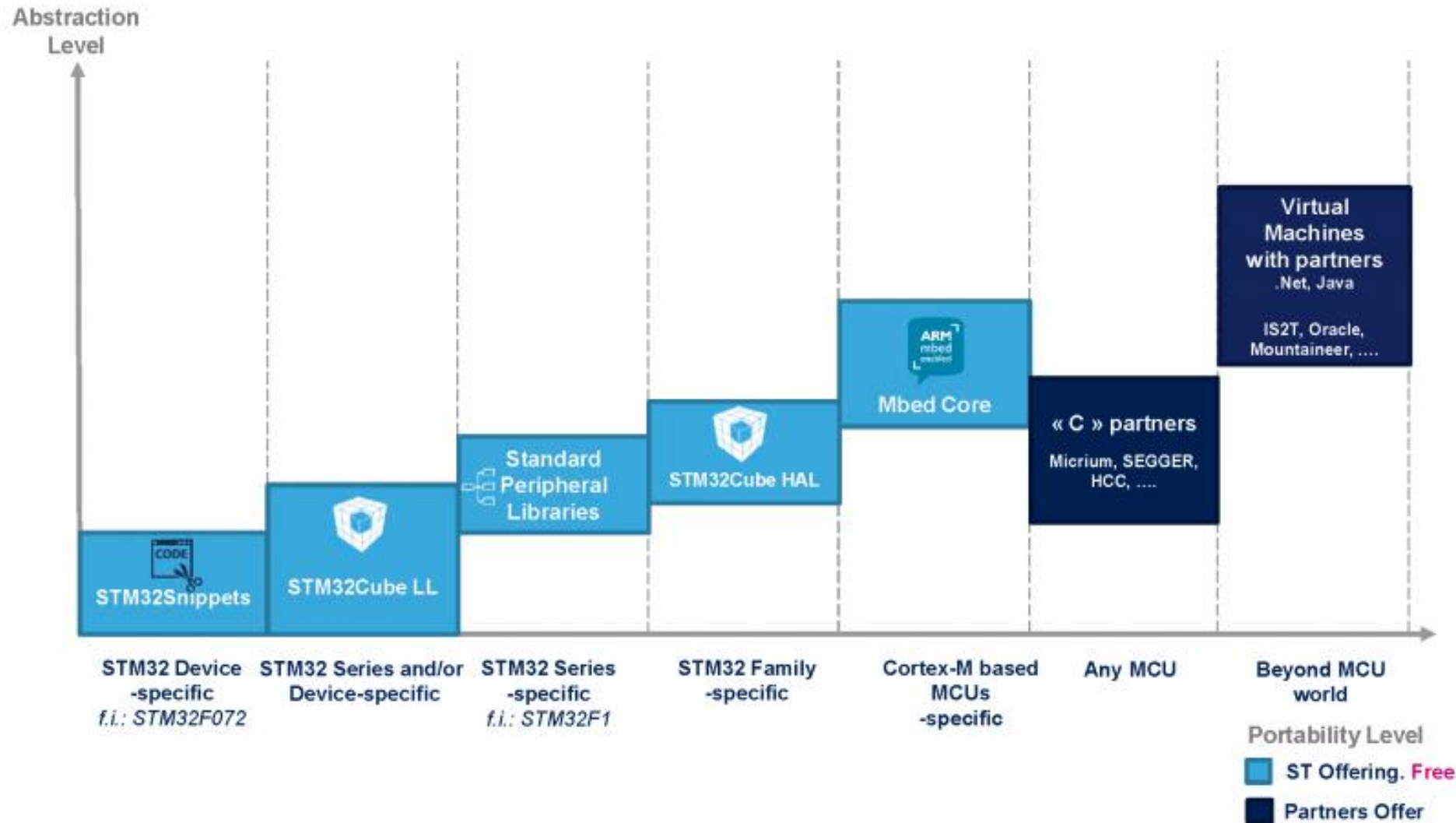
Without CMSIS, an embedded OS or middleware needs to include access functions to use processor features.

With CMSIS, an embedded OS or middleware can use standardized core access functions from device driver library

FIGURE 2.15

CMSIS-Core avoids the need for middleware or OS to carry their own driver code

7. Các thư viện lập trình cho STM32



7.1. STM32snippets

- collections of highly optimized code examples using CMSIS compliant direct register accesses
- to reduce the code overhead allowing
- to maximize the performance of the STM32 MCUs in various applications.
- The more than 100 Snippets per STM32 Series demonstrate efficient use of the STM32 peripherals with the smallest possible memory footprint.

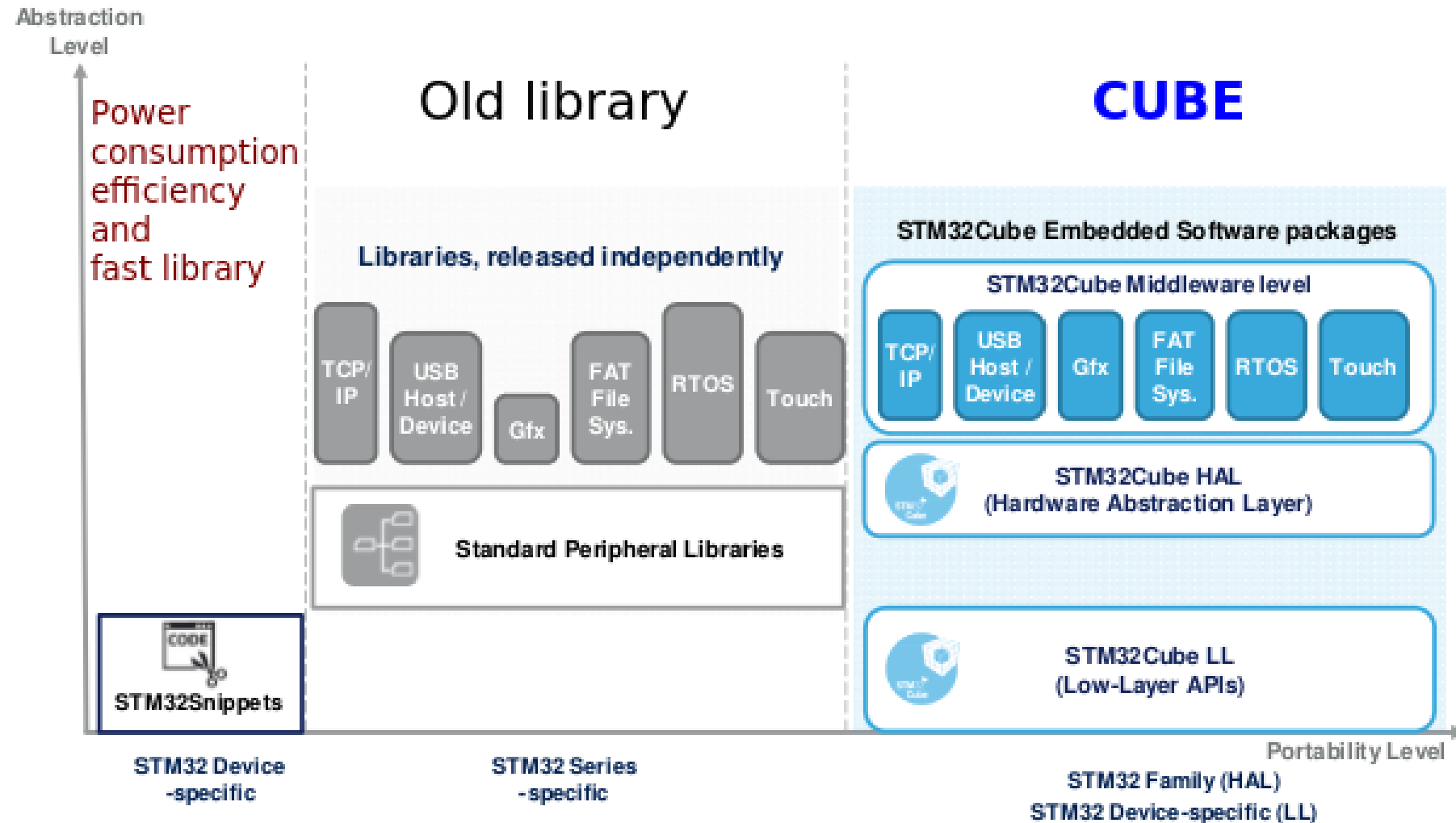
7.1. STM32snippets ..

- STM32Snippets gồm rất nhiều code ví dụ dựa trên các thanh ghi ngoại vi của STM32. Mục đích chính của nó phục vụ cho những người có kiến thức về chip 8 bit, lập trình hợp ngữ, lập trình C khá vững.
- Ưu điểm:
 - Tối ưu cao; Có thể truy cập tới mức thanh ghi; Kích thước code nhỏ
 - Sát với tài liệu tham khảo; Debug ở mức thanh ghi
- Nhược điểm:
 - Khó chuyển đổi code khi đổi sang dòng chip STM32 khác.
 - Phức tạp khi dùng với ngoại vi như USB
 - Developer phải hiểu được kiến thức về các ngoại vi ở mức độ thanh ghi
 - Chỉ hỗ trợ 2 dòng chip là L0 và F0.

7.2. Standard Peripheral Lib (SPL)

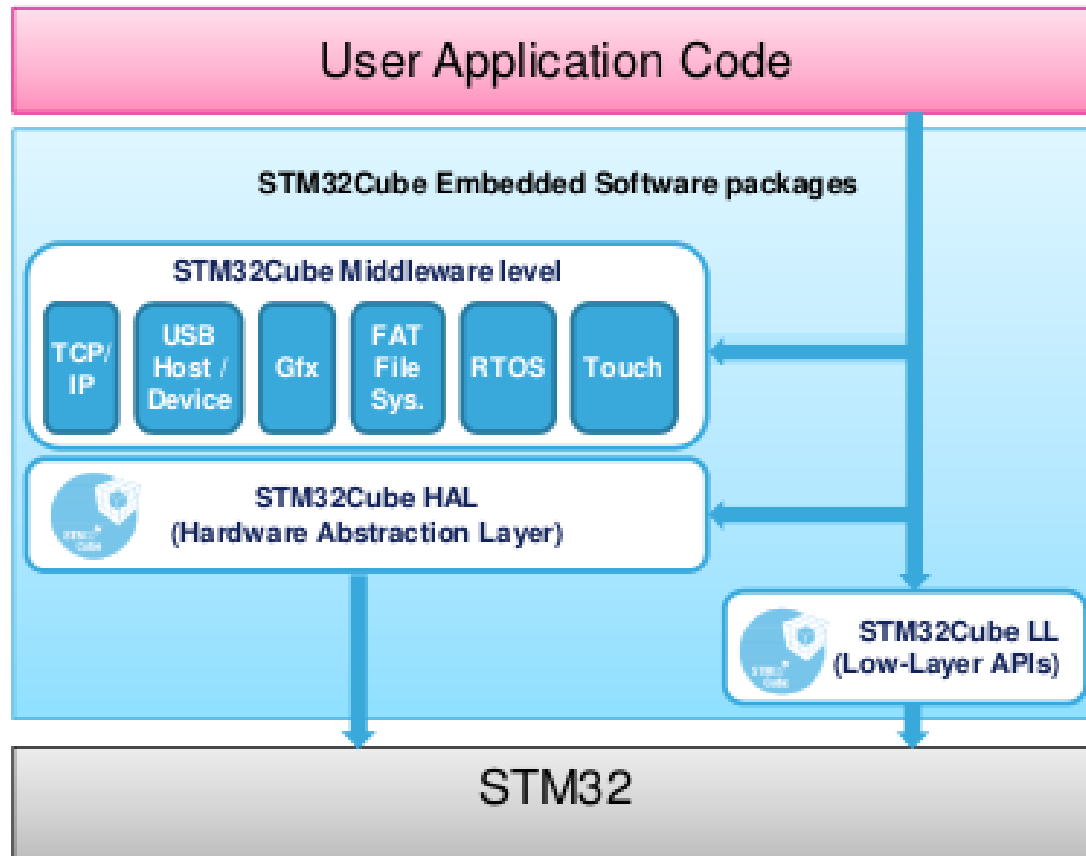
- Đây là thư viện gồm rất nhiều thư viện C dành cho ngoại vi của STM32. Phù hợp cho các developer có kiến thức về C tốt.
- Ưu điểm:
 - Tối ưu mức trung bình, phù hợp với nhiều trường hợp khác nhau
 - Không cần tác động trực tiếp lên thanh ghi như STM32snippets
 - Hỗ trợ 100% ngoại vi; Dễ debug
 - Có thể mở rộng với các middleware phức tạp như USB/TCP-IP/Graphics/Touch Sense
- Nhược điểm:
 - Chỉ hỗ trợ 1 số dòng STM32 nhất định
 - Không có hàm API HAL hỗ trợ chuyển đổi khi thay thế chip STM32 dòng khác
 - Thư viện middleware không thống nhất
 - Không hỗ trợ các dòng STM32 L0, L4, F7

7.3. STM32Cube



7.3. STM32Cube – Embedded software

■ Architecture and User entrypoint



- Three entry points for the user application:
 - Middleware stacks
 - HAL API
 - LL APIs
- Possible concurrent usage of HAL and LL

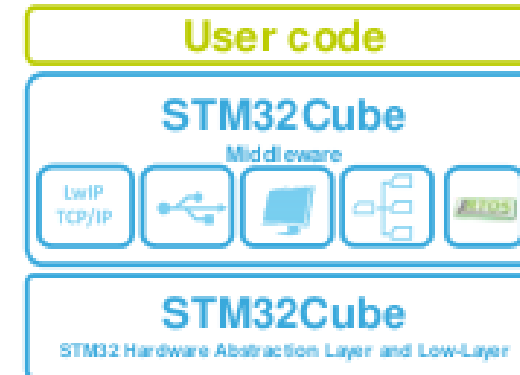
7.3. STM32Cube – HAL API

- Features:

- High level and functional abstraction
- Easy port from one series to another
- 100% coverage of all peripherals
- Integrates complex middleware such as USB/TCP-IP/Graphics/Touch Sense/RTOS
- Can work with STM32CubeMX tool on the PC to generate initialization code

- Limitations:

- May be challenging to low level C programmers in the embedded space.
- Higher portability creates bigger software footprints or more time spent executing adaptation code



Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
+++	+	++	+++	+++

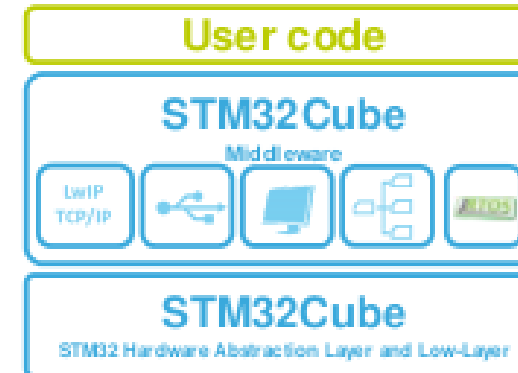
7.3. STM32Cube – LL API

- Features:

- Highly Optimized
- Register Level Access
- Small code expressions
- Closely follows the reference manual
- Debugging close to register level

- Limitations:

- Specific to STM32 devices, not portable directly between series
- Not matching complex peripherals such as USB
- Lack of abstraction means developers must understand peripheral operation at register level
- Available (today) on STM32 L4 series



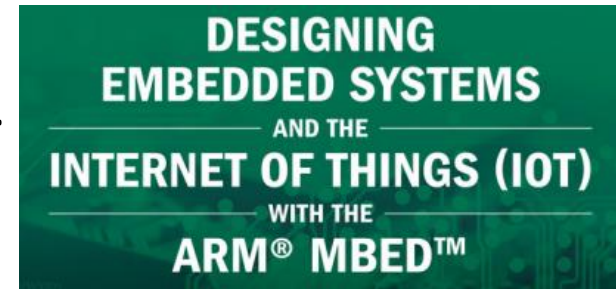
Portability	Optimization (Memory & Mips)	Easy	Readiness	Hardware coverage
	+++			++

7.3. STM32Cube - In conclusion

- First:
 - install the **STM32CubeMX** (graphical tool that allows configuring STM32 microcontrollers) and read this manual.
- Second:
 - install the appropriate **CUBE Library** for your STM32xxx (SW example and driver for TCP/IP, USB, FS, etc)
 - Also are available a **special CUBE libraries** for a particular application (BT, LPUART, etc).

7.4. The Arm® Mbed™ Systems

- Is a platform and operating system based on 32-bit ARM® Cortex-M microcontrollers
- is collaboratively developed by ARM® and its technical partners,
- is designed for Internet of Things (IoT) devices
- provides
 - the operating system,
 - cloud services,
 - tools,
 - and developer ecosystemto make the creation and deployment of IoT solutions possible.

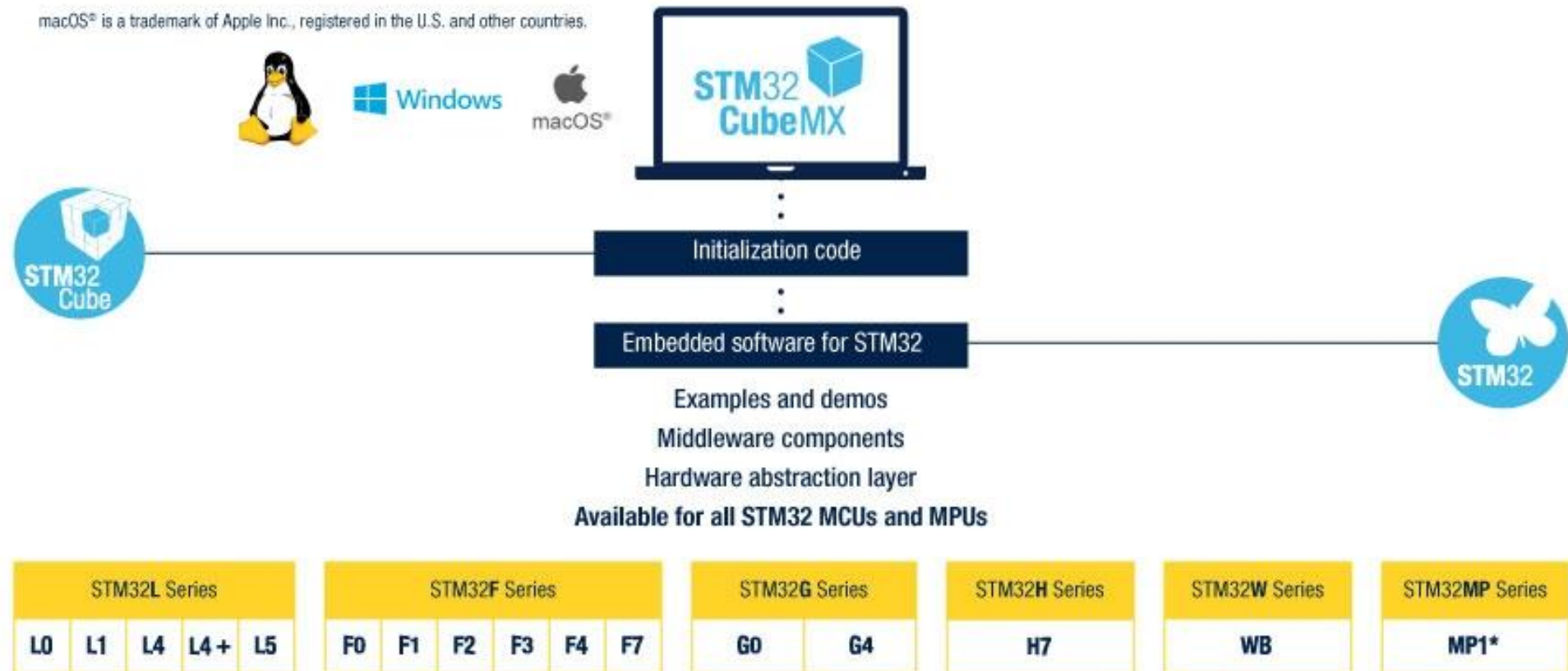
The logo for ARMmbed, featuring the word "ARM" in a bold, blue, sans-serif font, followed by "mbed" in a lighter blue, lowercase, sans-serif font.

8. Cài đặt phần mềm

- PHẦN MỀM KHỞI TẠO CODE **STM32CUBE MX**
- PHẦN MỀM LẬP TRÌNH **ARM KEIL UVISION5**
- Phần mềm mô phỏng **Proteus**

8.1. STM32CubeMX

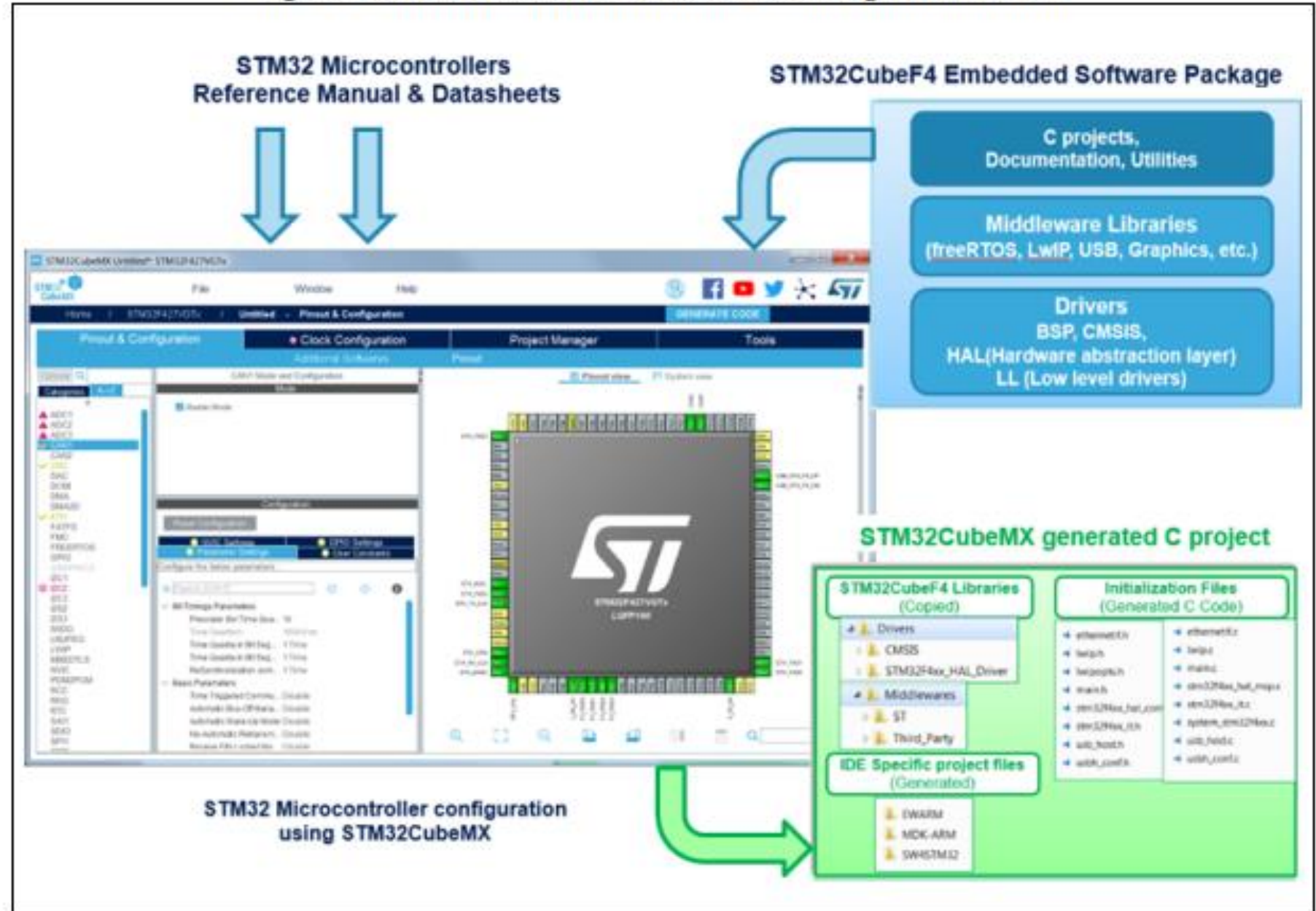
- PHẦN MỀM KHỞI TẠO CODE
- Công cụ giúp khởi tạo phần cứng, ngoại vi, xung nhịp... cho vi điều khiển STM32.



Note :* available for Cortex-M4 side only

STM32CubeMX

Figure 1. Overview of STM32CubeMX C code generation flow

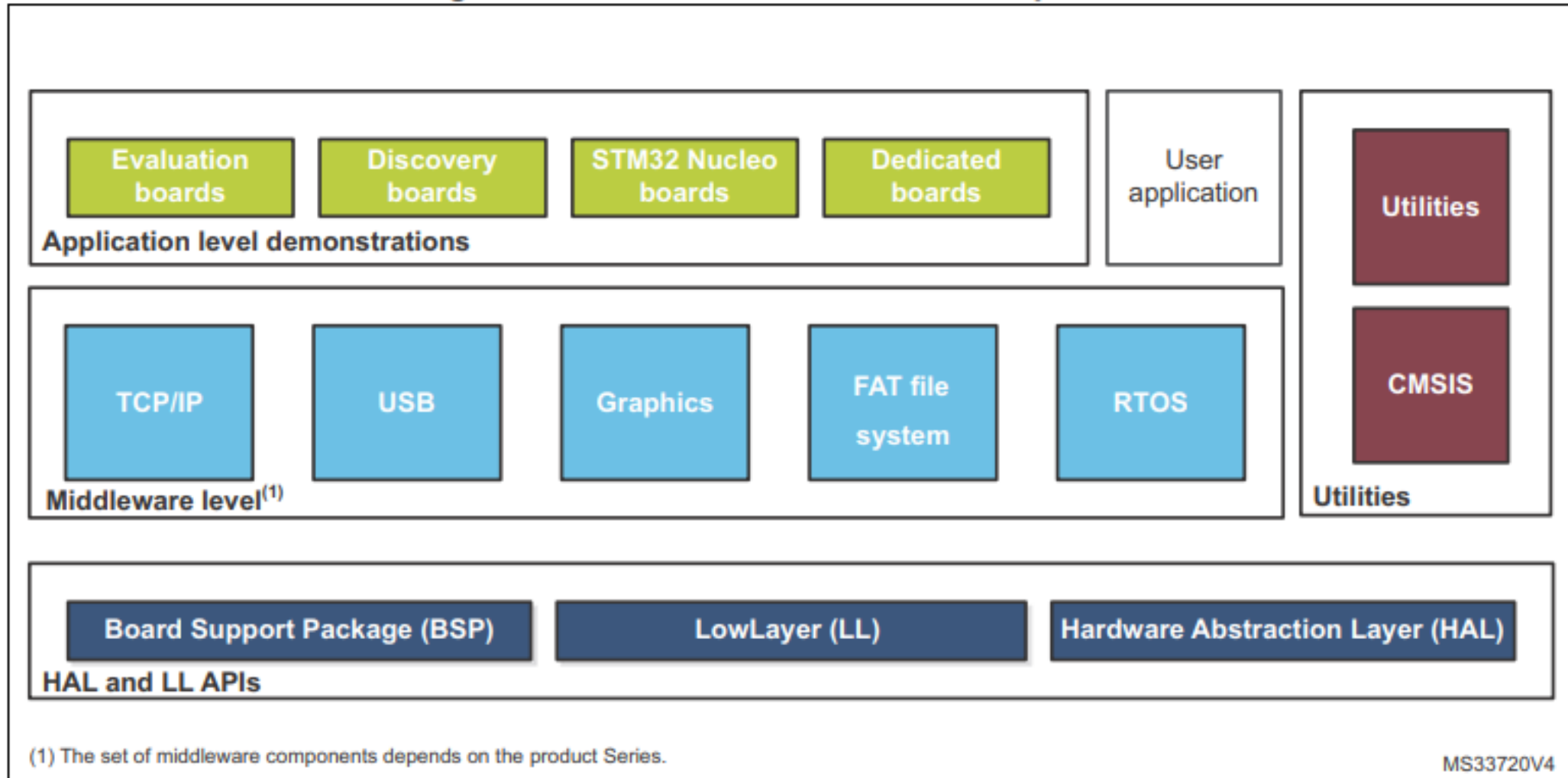


8.1. STM32CubeMX ..

- STM32CubeMX tự động download các driver mới nhất của ST dành cho các dòng chip của mình.
- ST đã không còn phát triển Standard Peripheral Libraries nữa, thay vào đó họ phát triển cấu trúc firmware mới bao gồm lớp cách ly phần cứng (HAL) bao gồm:
 - Các driver cho ngoại vi, lớp Middleware bao gồm hỗ trợ TCP/IP, USB, Graphics, FAT file system, Touch library, và hệ điều hành mã nguồn mở RTOS.
- Cấu trúc firmware mới này có mức độ trừu tượng cao hơn, tập trung vào các tính năng phần cứng chung thay vì tập trung thuần túy vào phần cứng. Mức độ trừu tượng cao hơn giúp phát triển các API thân thiện và có thể dễ dàng chuyển từ phần cứng này sang phần cứng khác.

The STM32CubeF4 firmware package comes with a rich set of examples running on STMicroelectronics boards. The examples are organized by board and provided with preconfigured projects for the main supported toolchains (see [Figure 1](#)).

Figure 1. STM32CubeF4 firmware components



8.2. Tổ chức Firmware của STM32 F4

Level 0:

- Board Support Package (**BSP**): cung cấp các API liên quan đến các thành phần phần cứng trên các board (ví dụ driver LCD, MicroSD).
 - Khi sử dụng các board của ST ví dụ như STM32F4 Discovery, các API này giúp chúng ta nhanh chóng cấu hình/sử dụng các phần cứng có sẵn trên đó ví dụ LED, Buttons, Gyroscope...
- Hardware Abstraction Layer (**HAL**): Cung cấp các driver ở mức thấp và các phương thức giao diện phần cứng để giao tiếp với các mức trên (application, libraries và stacks).
 - HAL API chia làm 2 nhóm: nhóm thứ nhất cung cấp các API chung đối với tất cả các serie STM32, và nhóm API mở rộng riêng cho từng dòng chip.

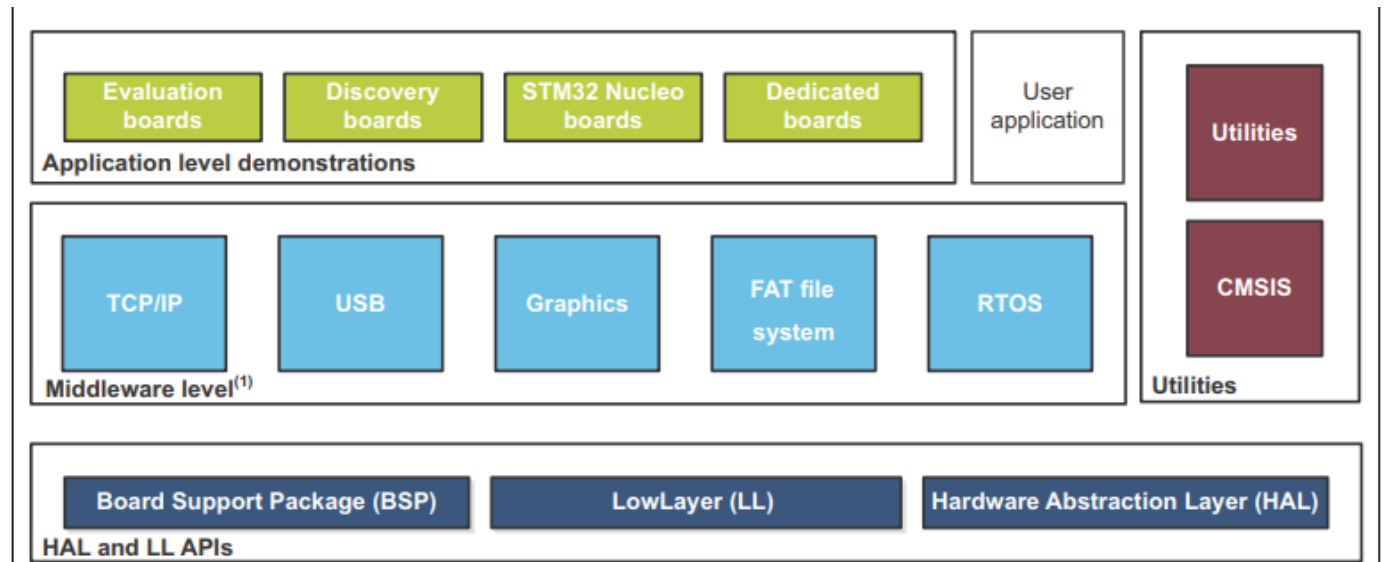
8.2. Tổ chức Firmware của STM32 F4

Level 1:

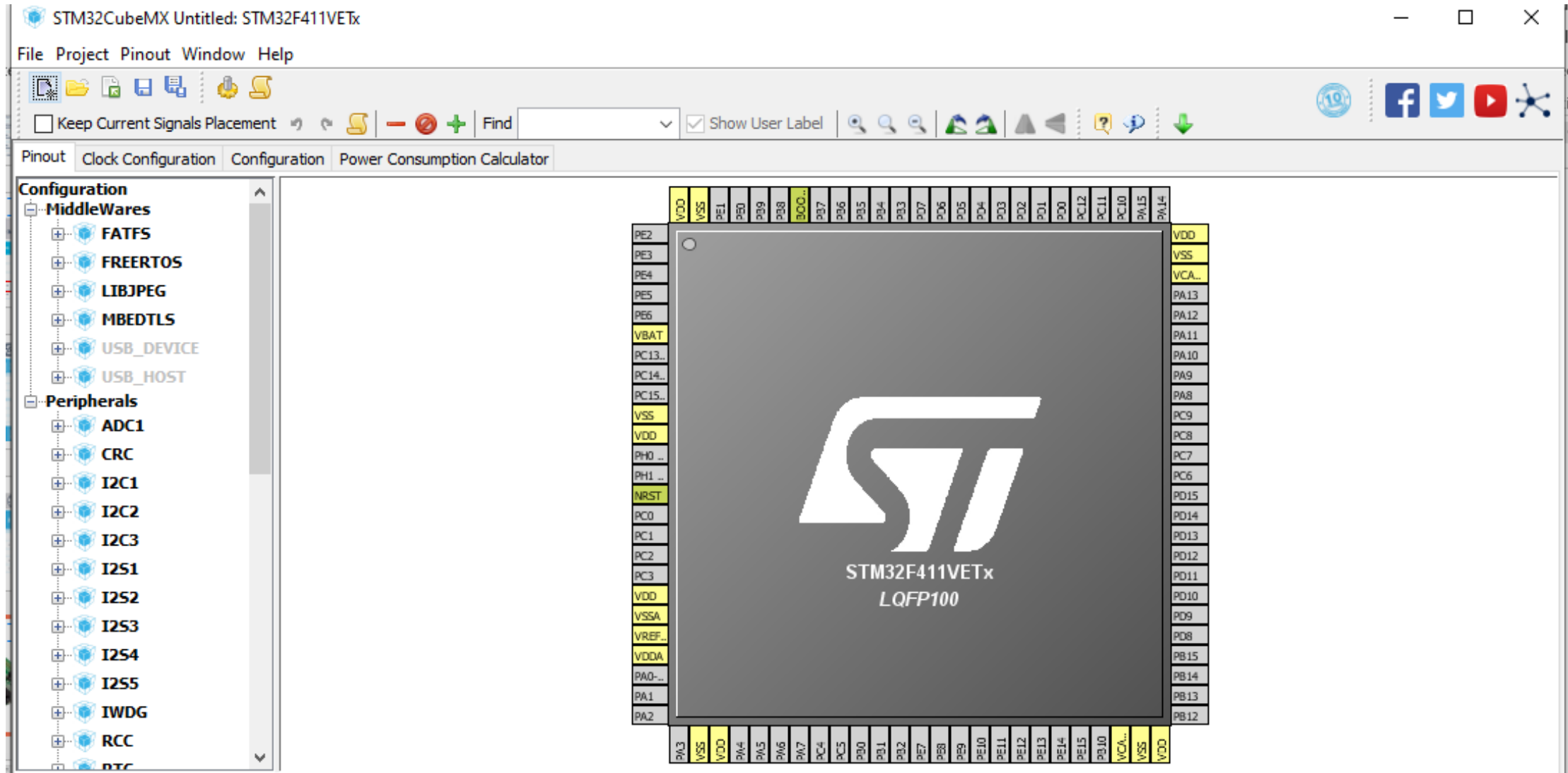
- Các thành phần **Middlewares**: Bao gồm các thư viện USB Device, STMTouch, thư viện đồ họa STemWin, hệ điều hành FreeRTOS, FatFS.

Level 2:

- Dựa trên lớp dịch vụ Middleware, lớp trừu tượng mức độ thấp và sử dụng các ngoại vi.

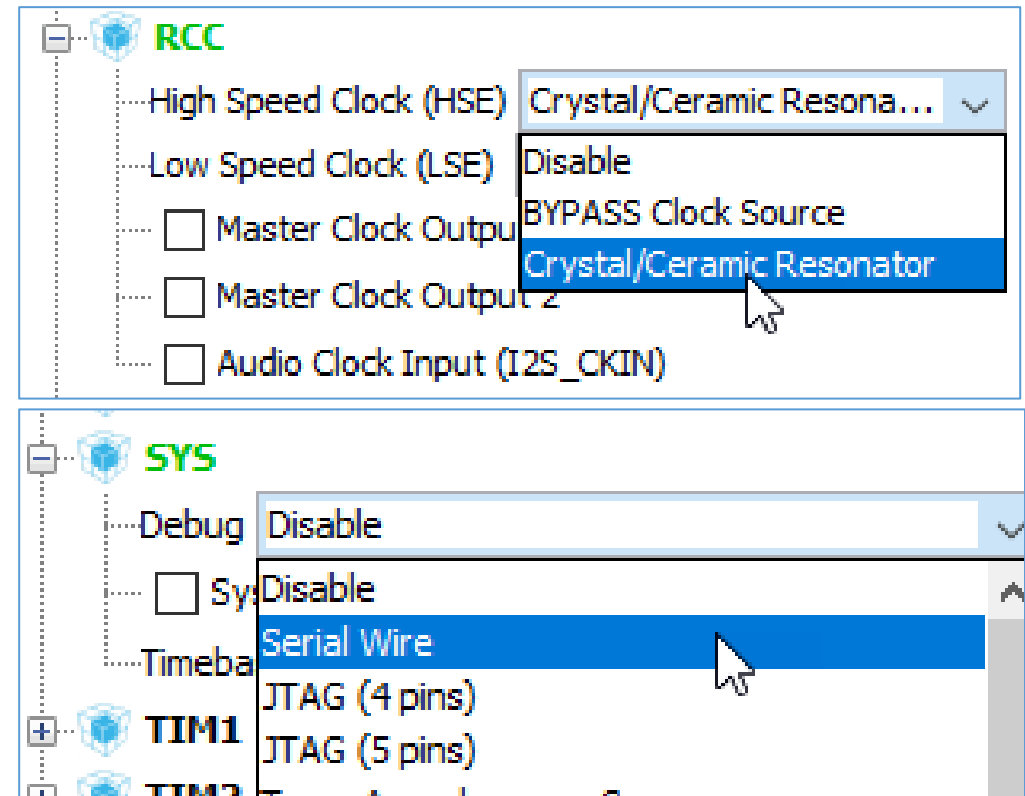


8.3. Pinout tab



8.3. Pinout tab ..

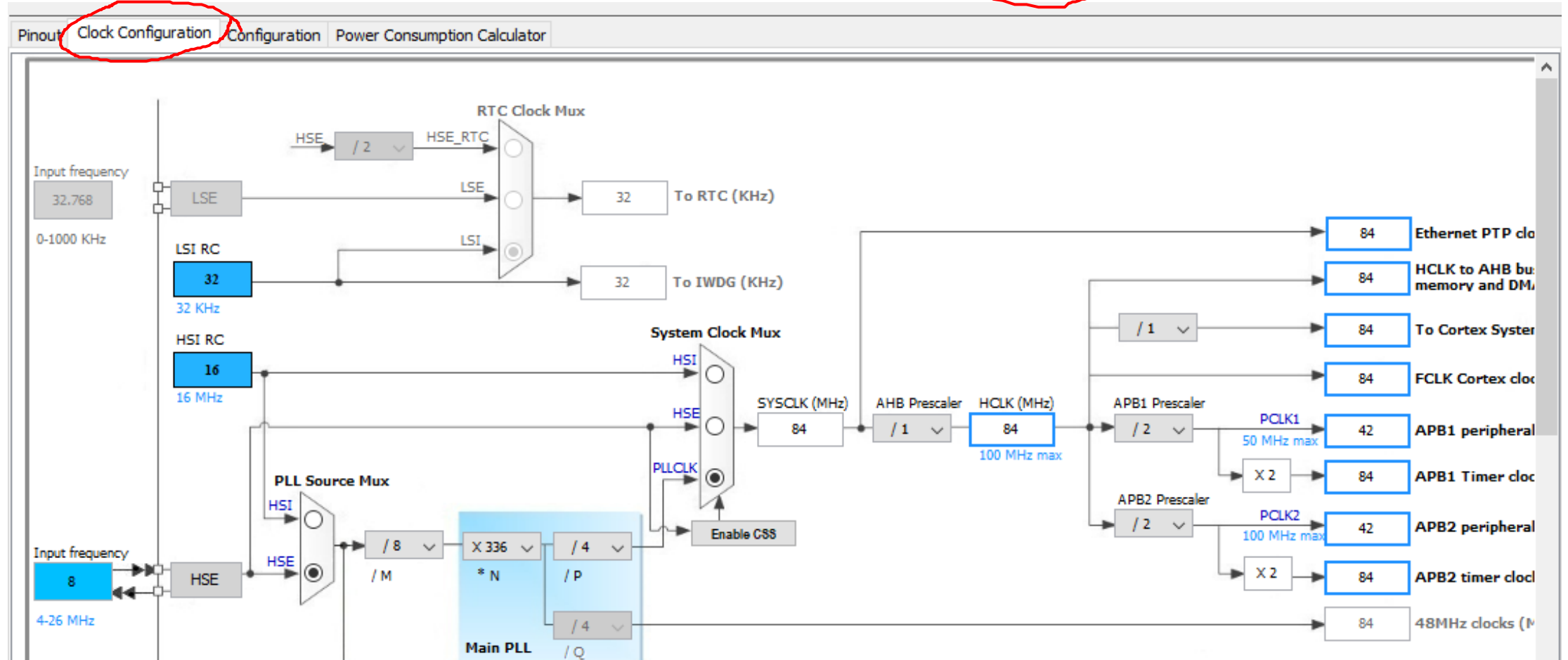
- **Middleware**: chọn/cho phép các middleware sẽ sử dụng
- **Peripherals**: chọn các ngoại vi sẽ sử dụng
 - **RCC** (System Reset & Clock Control):
Nguồn clock cho hệ thống ở ngoài
 - Crystal: Nguồn tạo dao động là thạch anh ở ngoài chip (có trên board mạch)
 - **SYS** - debug: cấu hình nạp code cho vi điều khiển
 - Serial Wire: vi điều khiển được cấu hình nạp code thông qua chân SWDIO và SWCLK (sử dụng mạch nạp ST-Link và kết nối với vi điều khiển thông qua các chân này).





8.4. Clock configuration

- Cấu hình tần số hoạt động của CPU và các ngoại vi

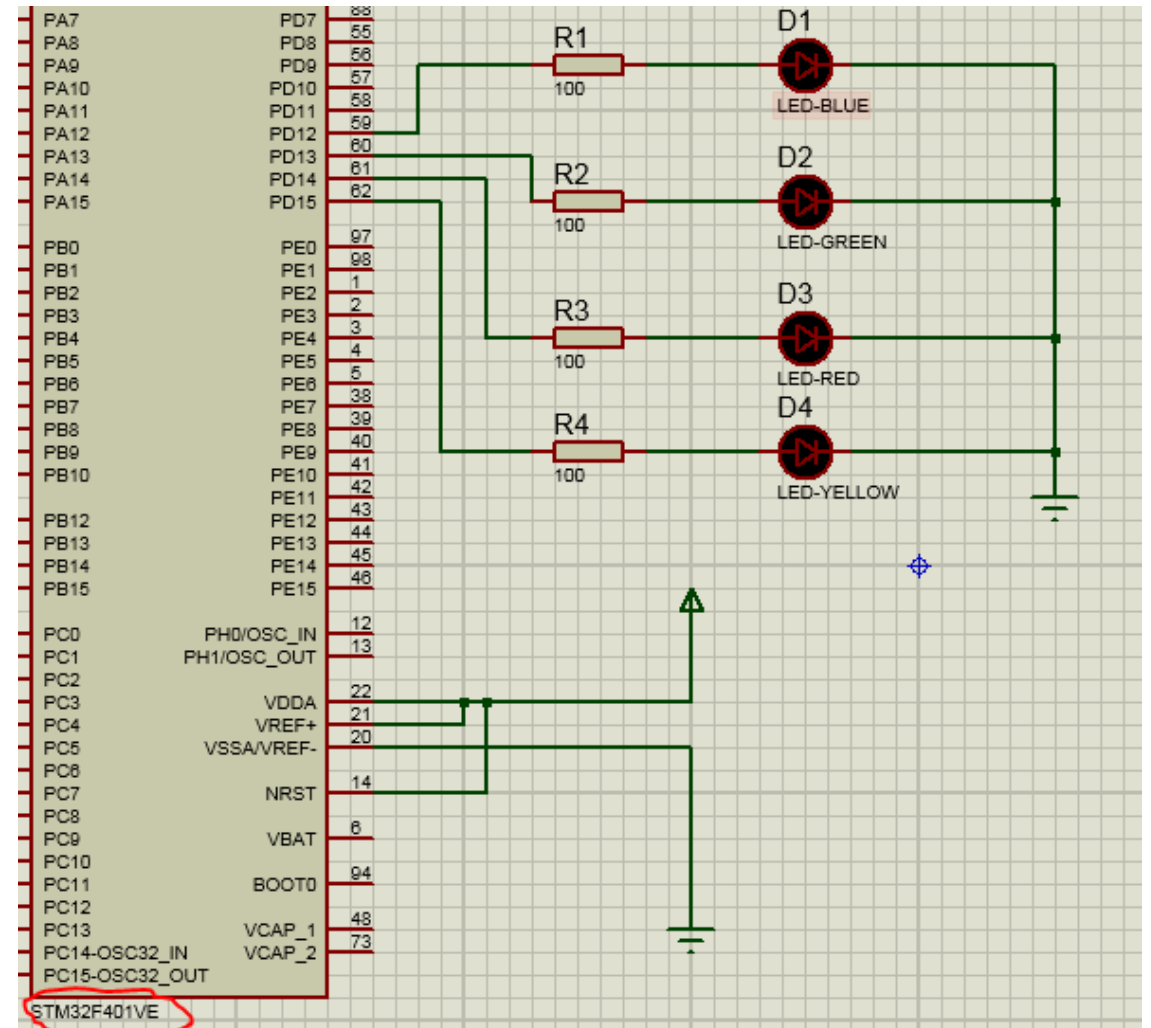


9. THỰC HÀNH

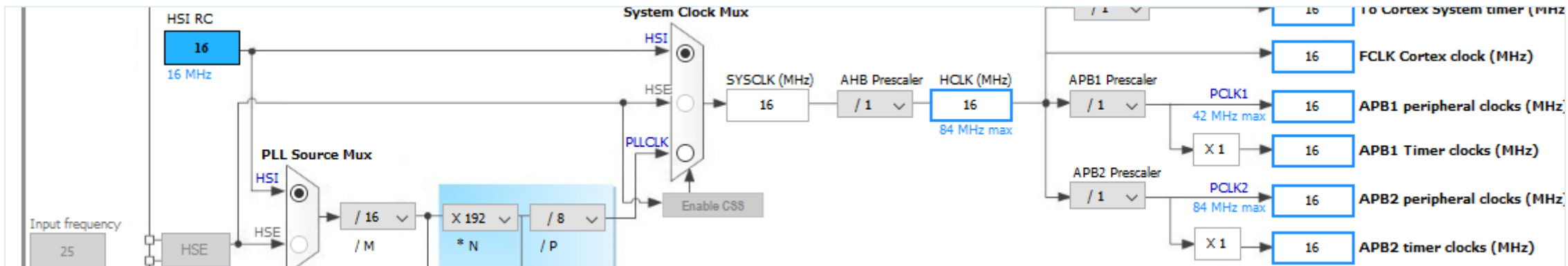
Thực hành các bài lập trình trên KIT phát triển STM32 Disco

Bài 1.0. Lập trình GPIO cơ bản

- Hệ thống được thiết kế như hình, Yêu cầu Lập trình bật/tắt Led sau mỗi giây
- VĐK: STM32F401VE
- Nguồn xung: HSI – 16MHz



Bài 1. Lập trình GPIO cơ bản ..

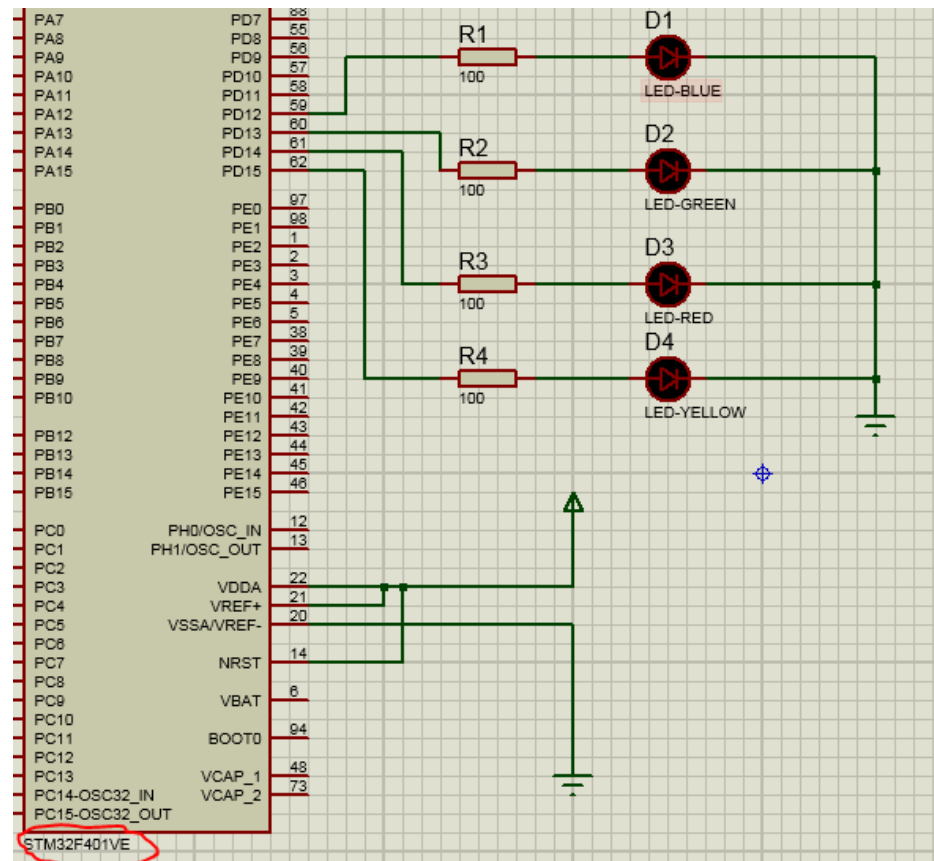


Bài 1. Lập trình GPIO cơ bản ..

```
10  SystemClock_Config();
11  /* Initialize all configured peripherals */
12  MX_GPIO_Init();
13  /* USER CODE BEGIN WHILE */
14  while (1)
15  {
16      HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
17      HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
18      HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);
19      HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_15);
20
21      HAL_Delay(1000);
22  }
23
```

Bài 1.1. Lập trình GPIO cơ bản

- *Hệ thống được thiết kế như hình, Yêu cầu Lập trình:*
- *Sáng lần lượt các led từ trên xuống dưới, sau đó tắt hết và lặp lại*
- VĐK: **STM32F401VE**
- Nguồn xung: **HSI – 16MHz**
- Hàm sử dụng:

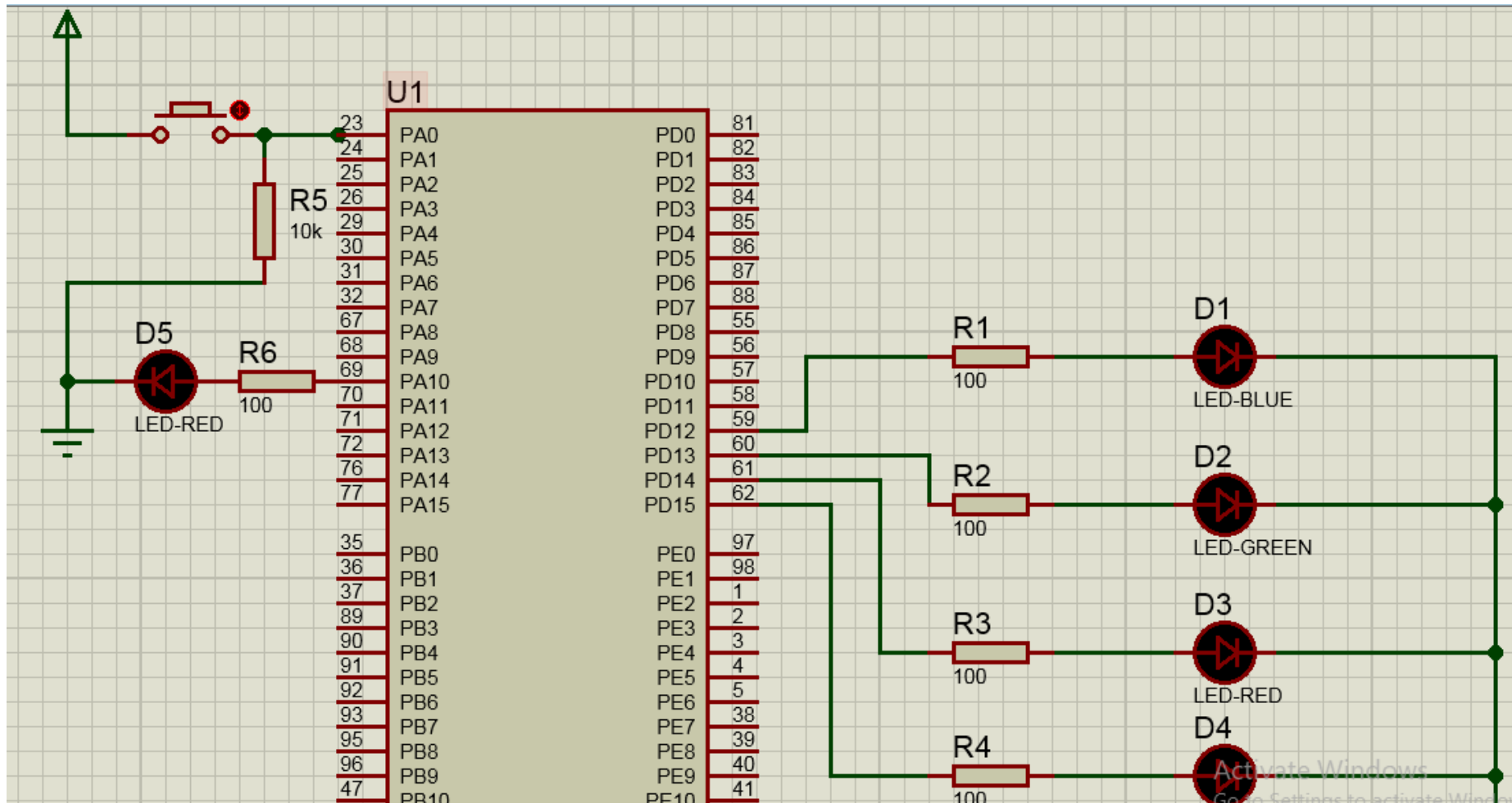


```
HAL_GPIO_WritePin(cổng, chân, GPIO_PIN_SET); //đặt mức cao lên pin
```

```
HAL_GPIO_WritePin(cổng, chân, GPIO_PIN_RESET); //đặt mức thấp lên pin
```

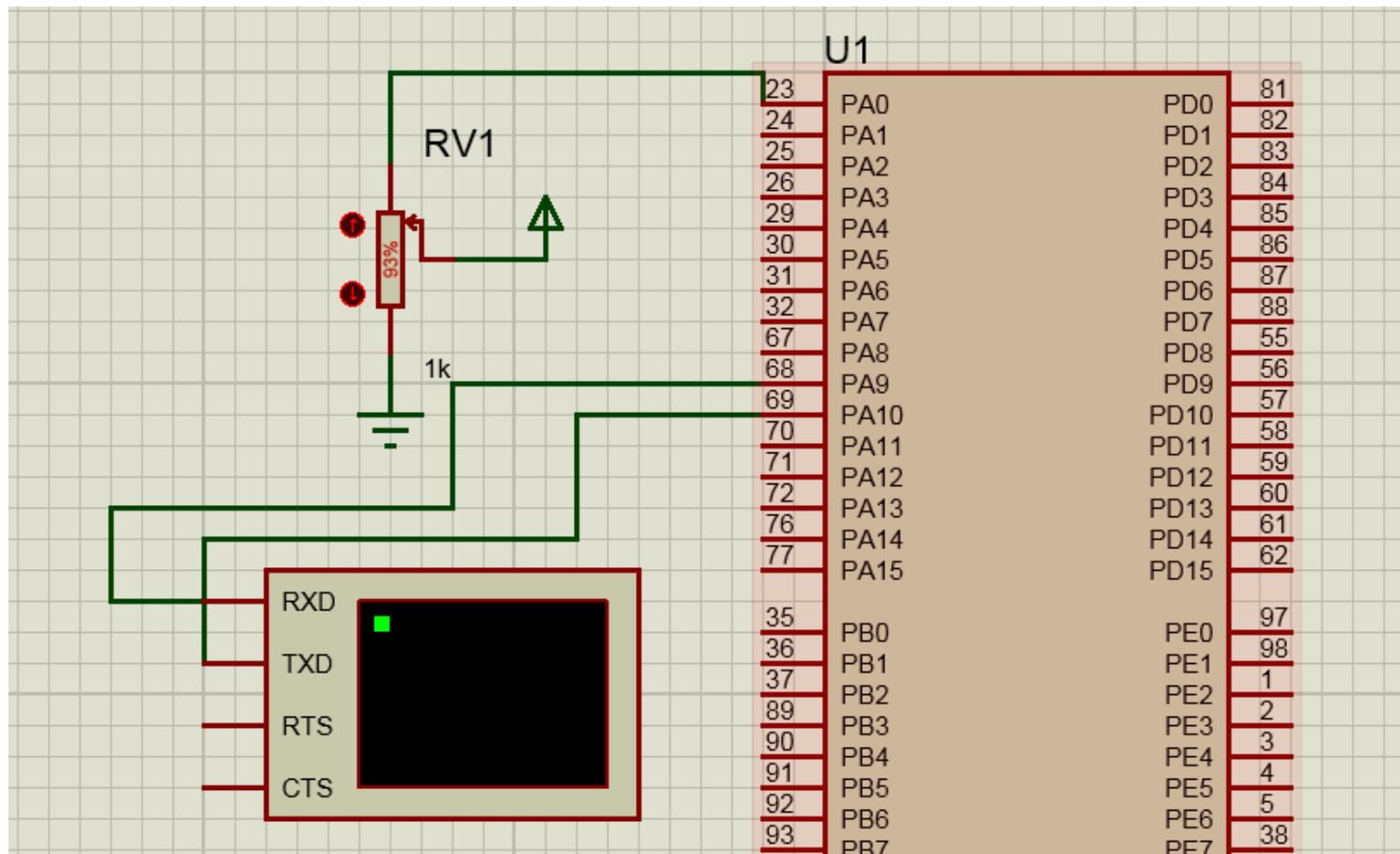
Bài 2.0. Lập trình ngắt ngoài

- Lập trình bật/tắt Led **D5**, khi nhấn vào nút bấm được nối ở chân **A0**



Bài 3. ADC

- Bài toán: Đọc dữ liệu từ biến trở ở chân A0, gửi dữ liệu đọc được qua module Bluetooth (giao tiếp **UART**)



Bài 3. ADC ...

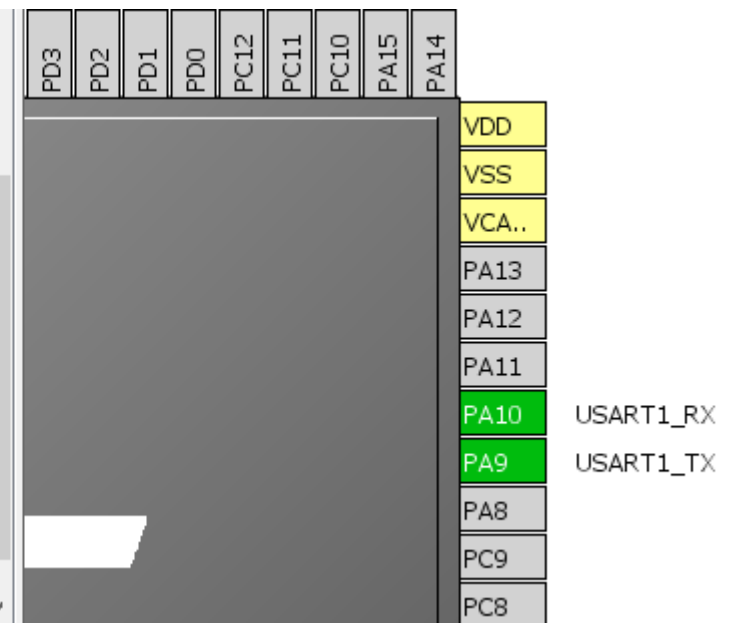
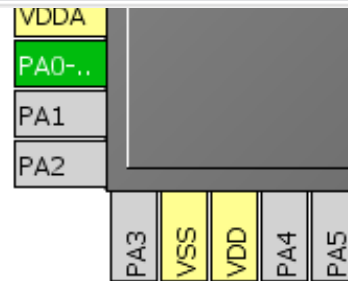
Peripherals

ADC1

- ☒ IN0
- ☐ IN1
- ☐ IN2
- ☐ IN3
- ☐ IN4
- ☐ IN5
- ☐ IN6
- ☐ IN7
- ☐ IN8
- ☐ IN9
- ☐ IN10
- ☐ IN11
- ☐ IN12
- ☐ IN13
- ☐ IN14

- ☐ SPI4
- ☐ SYS
- ☐ TIM1
- ☐ TIM2
- ☐ TIM3
- ☐ TIM4
- ☐ TIM5
- ☐ TIM9
- ☐ TIM10
- ☐ TIM11
- ☒ USART1
 - Mode: Asynchronous
 - Hardware Flow Control (RS232): Disable
- ☐ USART2
- ☐ USART6

ADC1_IN0



ADC1 Configuration

☒ Parameter Settings
 ☒ User Constants
 ☒ NVIC Settings
 ☒ DMA Settings
 ☒ GPIO Settings

Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 global interrupt	<input checked="" type="checkbox"/>	0	0

USART1 Configuration

☒ Parameter Settings
 ☒ User Constants
 ☒ NVIC Settings
 ☒ DMA Settings
 ☒ GPIO Settings

Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

NVIC Configuration

☒ NVIC
 ☒ Code generation

Priority Group: 4 bits for pre-emption priority 0 bits for subpriority
 ☐ Sort by Preemption Priority and Sub Priority

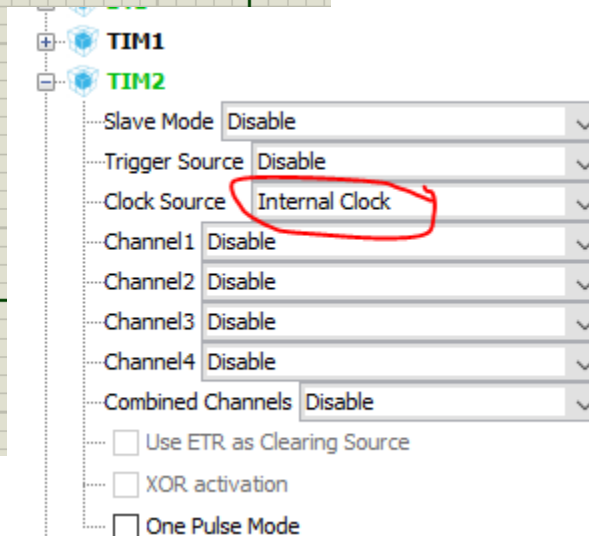
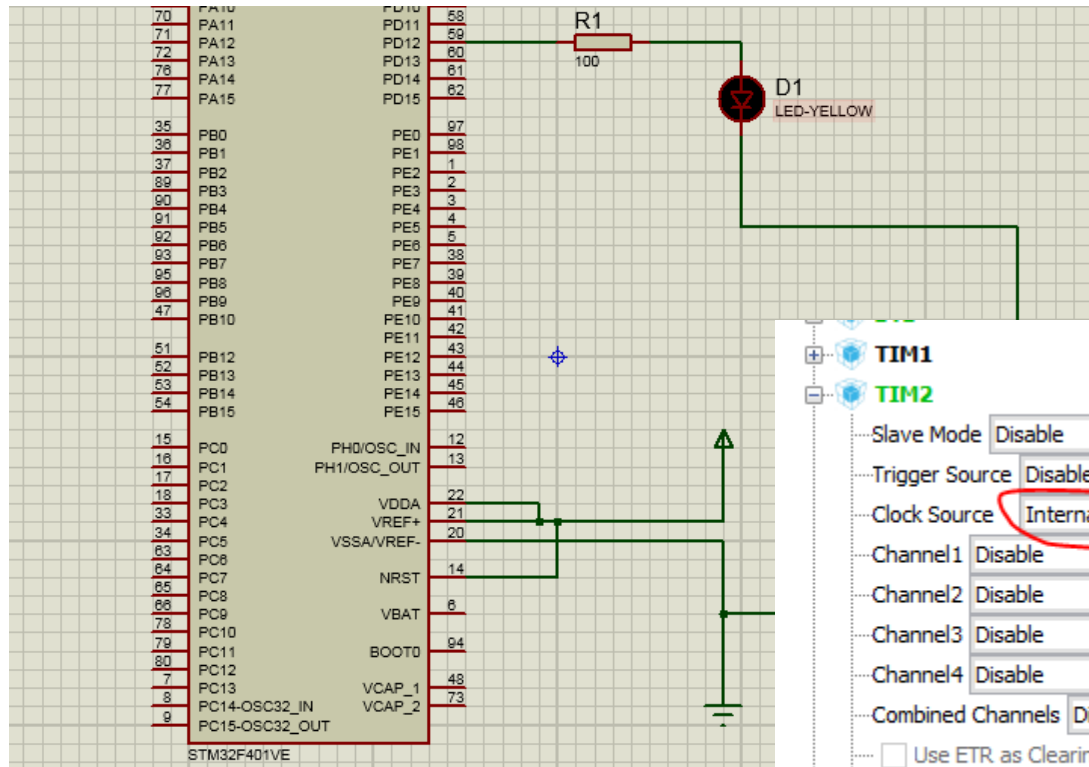
Search:

☐ Show only enabled interrupts

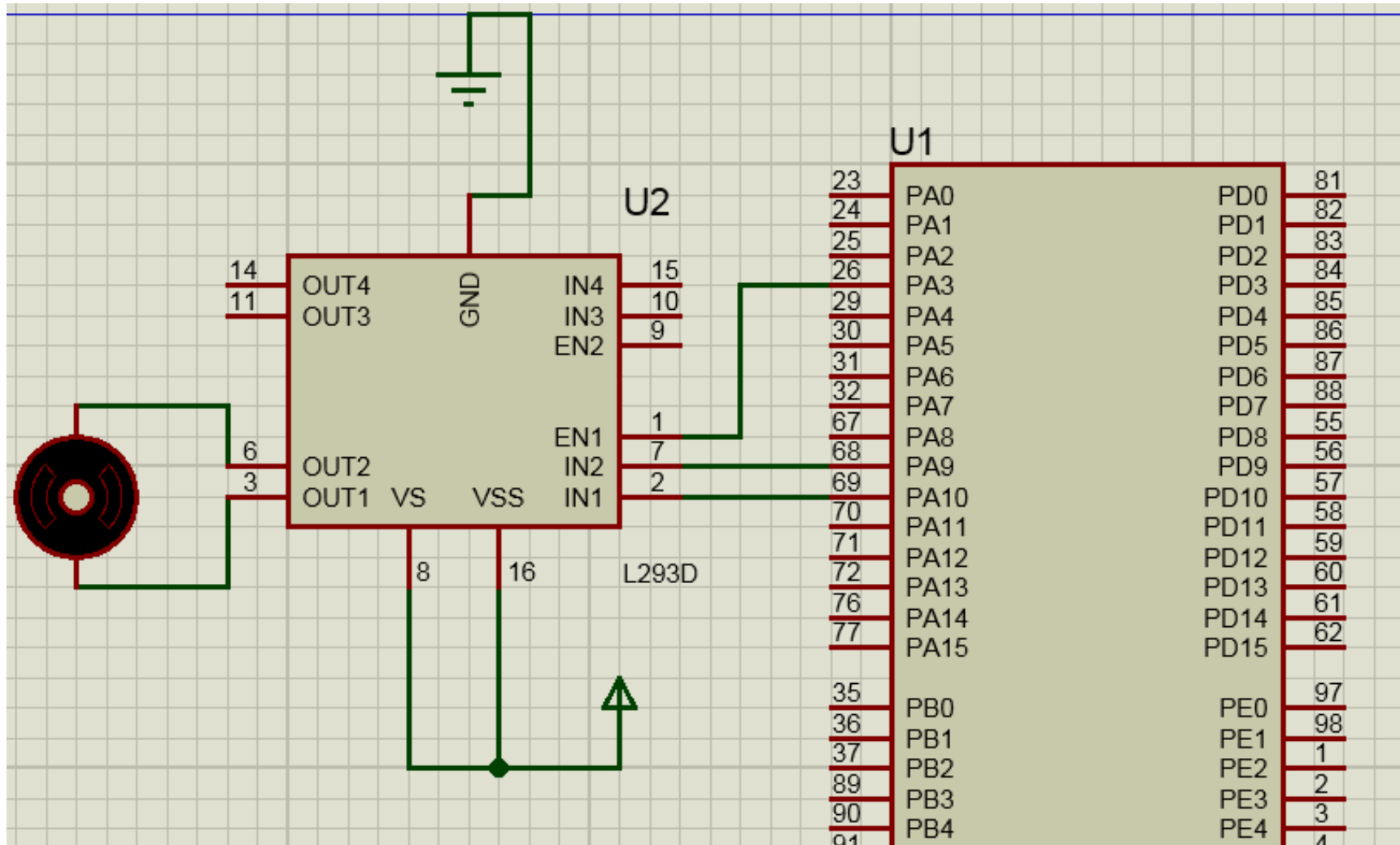
Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
ADC1 global interrupt	<input checked="" type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	1	0
FPU global interrupt	<input type="checkbox"/>	0	0

BỘ ĐỊNH THỜI (TIMER)

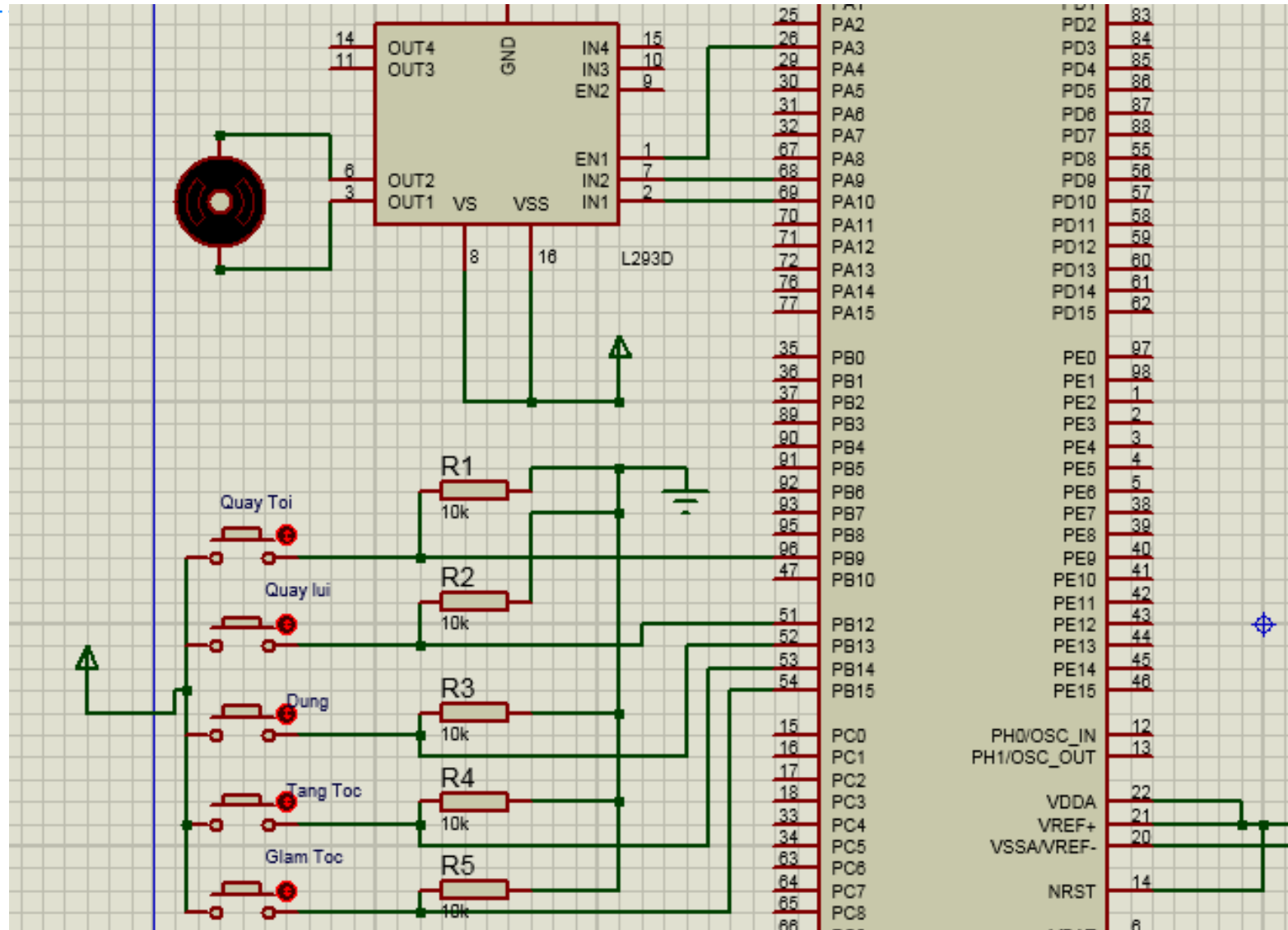
- Bật/ Tắt led sử dụng Timer



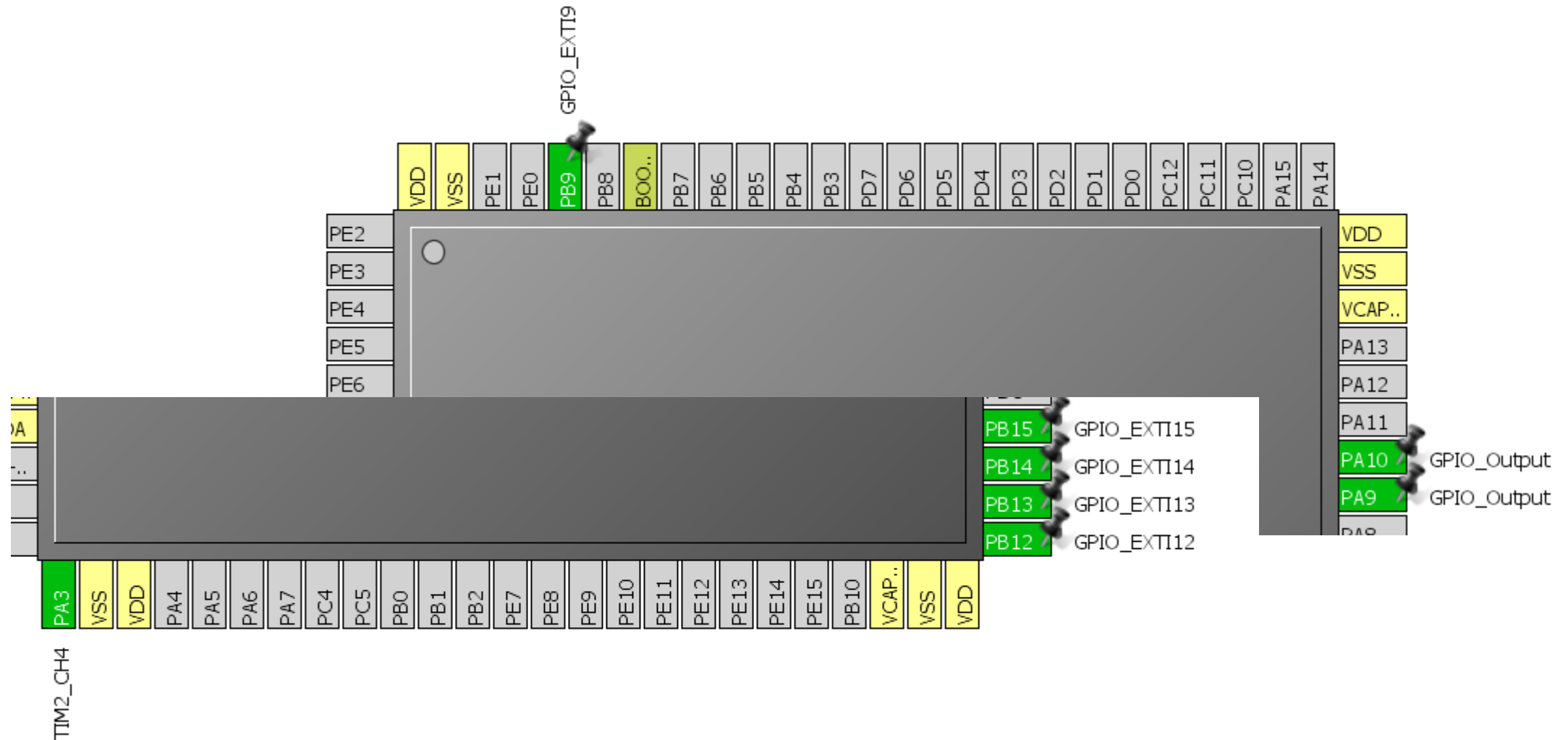
Timer – Điều chế độ rộng xung



Kết hợp: Ngắt ngoài + Băm xung



Kết hợp: Ngắt ngoài + Băm xung ...



Cấu hình

TIM2 Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Configure the below parameters :

Search :

Counter Settings

Prescaler (PSC - 16 bits value)	1599 ✓
Counter Mode	Up
Counter Period (AutoReload Register - 32 bits val...)	99 ✓
Internal Clock Division (CKD)	No Division

Trigger Output (TRGO) Parameters

Master/Slave Mode	Disable (no sync between this TIM
Trigger Event Selection	Reset (UG bit from TIMx_EGR)

PWM Generation Channel 4

Mode	PWM mode 1
Pulse (32 bits value)	0
Fast Mode	Disable

Prescaler (PSC - 16 bits value)
Prescaler must be between 0 and 65 535.

TIM2 Configuration

✓ Parameter Settings ✓ User Constants ✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0

NVIC Configuration

✓ NVIC ✓ Code generation

Priority Group 4 bits for pre-emption priority 0 bits for subpriority ☐ Sort by Preemption Priority and Sub Priority

Search

☐ Show only enabled interrupts

Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line[9:5] interrupts ✓	<input checked="" type="checkbox"/>	0	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0
EXTI line[15:10] interrupts ✓	<input checked="" type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

Kiểm tra (maicuongtho@gmail.com)

.file thiết kế mạch, và main.c

- **Câu 1.** Hệ thống nhúng có 10 led được gắn trên Cổng A của STM32F401, Lập trình ứng dụng sáng led theo qui tắc sau
 - Led ở vị trí chẵn sáng, lẻ tắt, thời gian 1s, tiếp đến
 - Led ở vị trí lẻ sáng, chẵn tắt, thời gian 1s, tiếp đến
 - Tắt cả led đều sáng 1s
 - Tắt tất cả các Led → lặp lại từ đầu
- **Câu 2.** HTN có 1 nút bấm và 1 động cơ DC Moto, cầu L293D, và 10 led như câu 1. Lập trình ứng dụng cho HTN theo yêu cầu sau
 - 10 Led hoạt động như câu một
 - Ban đầu động cơ không quay, khi nhấn nút bấm thì động cơ bắt đầu quay

Ôn tập

- 1) Kiến trúc và các thành phần cơ bản của HTN
- 2) Chuẩn giao tiếp có dây của STM32F4: **UART**
- 3) Quá trình số hóa tín hiệu tương tự (ADC)
- 4) Cấu trúc cơ bản của Bộ định thời
- 5) Lập trình GPIO cơ bản
- 6) Lập trình ngắt và điều khiển động cơ
- 7) Hệ thống nhúng thời gian thực là gì, hệ điều hành thời gian thực?. Lập trình Nháy led trên HTN STM32 với FreeRTOS

Bài tập

- 1) Viết chương trình đọc nút nhấn trên chân PC13 sử dụng ngắt ngoài, nhấn lần 1 đèn trên chân PB9 sáng, lần thứ 2 led trên chân PB9 tắt... lặp đi lặp lại như vậy.
- 2) Viết chương trình đọc 2 nút nhấn trên chân PC13 và PB8 sử dụng ngắt ngoài. Nút nhấn trên chân PC13 có mức ưu tiên cao hơn chân PB8. Khi nhấn nút trên chân PC13 led trên chân PB9 sáng, và khi nhấn nút trên chân PB8 led trên chân PB9 tắt. Lặp đi lặp lại quá trình đó.
- 3) Cấu hình ngắt tràn TIM4 với tần số là 1Hz. Led trên chân PB9 sáng tắt với tốc độ là 1s sáng, 1s tắt. Lặp đi lặp lại quá trình này.
- 4) Cấu hình ngắt tràn TIM1 với tần số là 2Hz. Led trên chân PB9 sáng tắt với tốc độ là 0.5s sáng, 0.5s tắt. Lặp đi lặp lại quá trình này.

Bài tập ..

- 5) Cấu hình PWM với duty cycle 20%, tần số 1Hz trên chanel1,2,3,4 của TIM2.
- 6) Cấu hình PWM với duty cycle 20%, tần số 1Hz trên chanel 1(20%) và chanel 1N(80%) của TIM1.
- 7) Đọc giá trị biến trở sử dụng kênh ADC1 channel 0 trên chân PA0, hiển thị giá trị ADC lên máy tính thông qua UART.
- 8) Đọc giá trị cảm biến nhiệt độ trong bộ ADC, nếu giá trị lớn hơn 35°C thì bật led ở chân PB9 cảnh báo, hiển thị giá trị nhiệt độ lên máy tính thông qua UART.