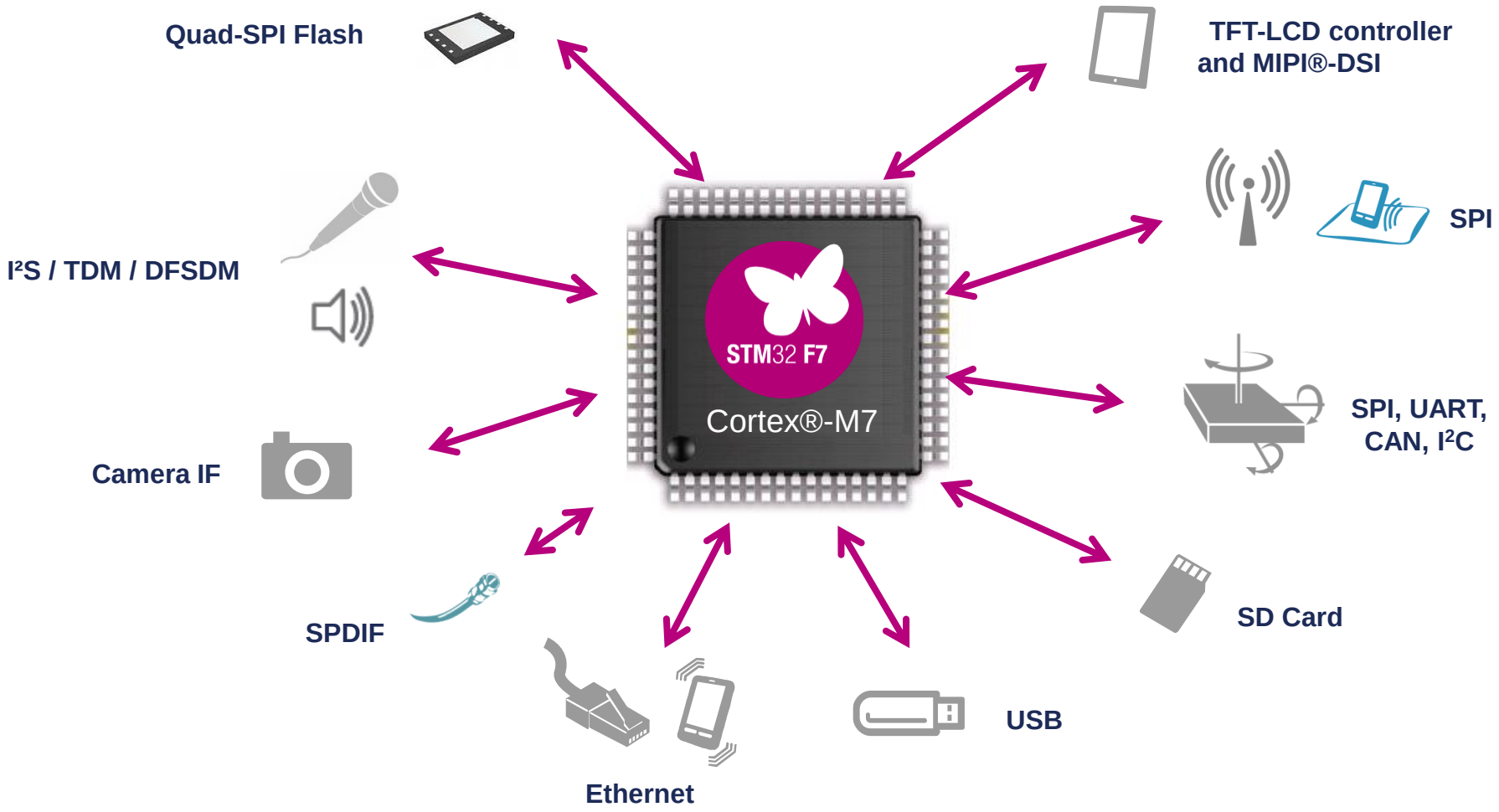
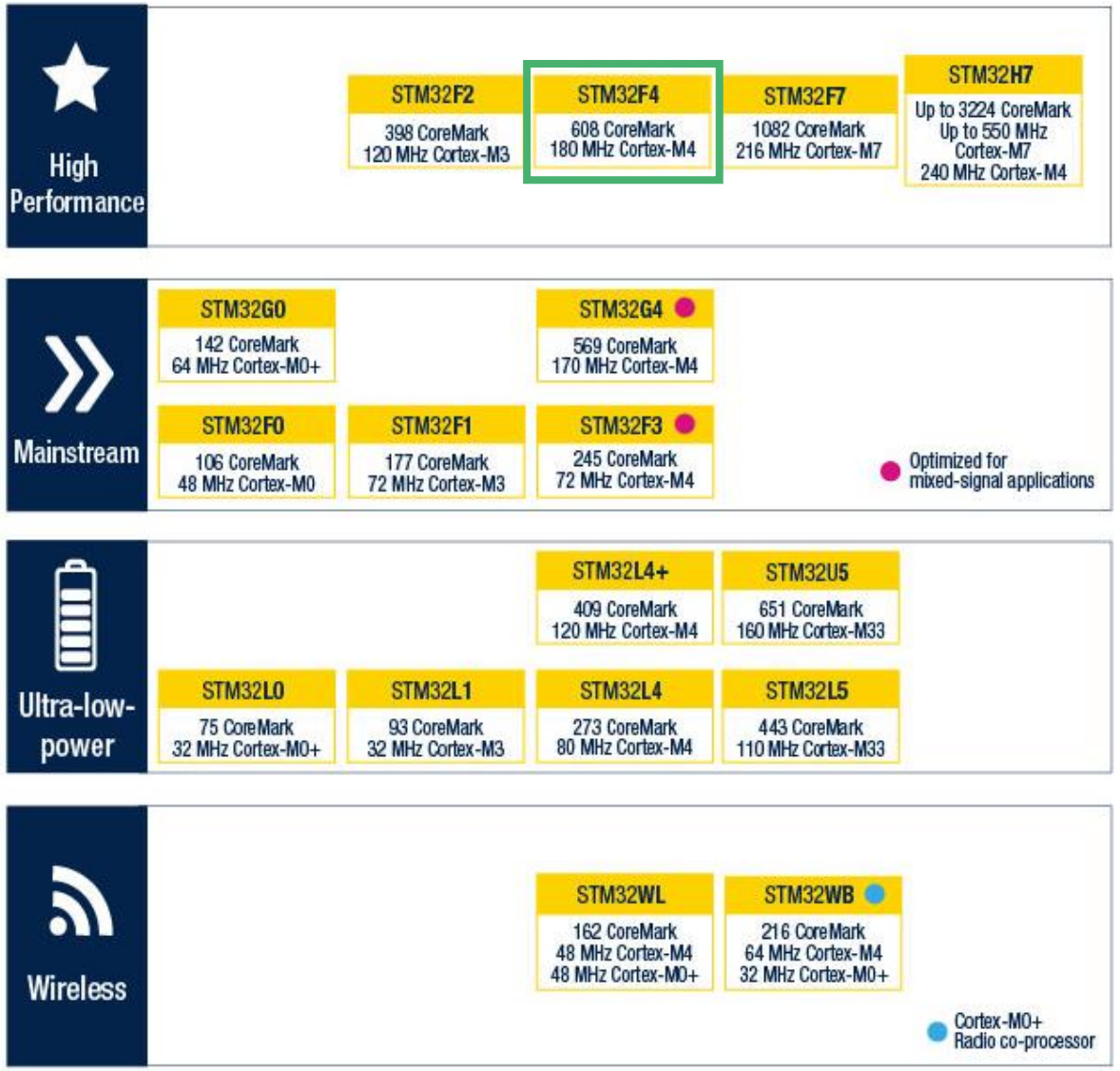
STM32 Overview

Ing. Davide Giacalone

Advanced Research Engineer, SRA

STM32 is more than a CPU





- 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

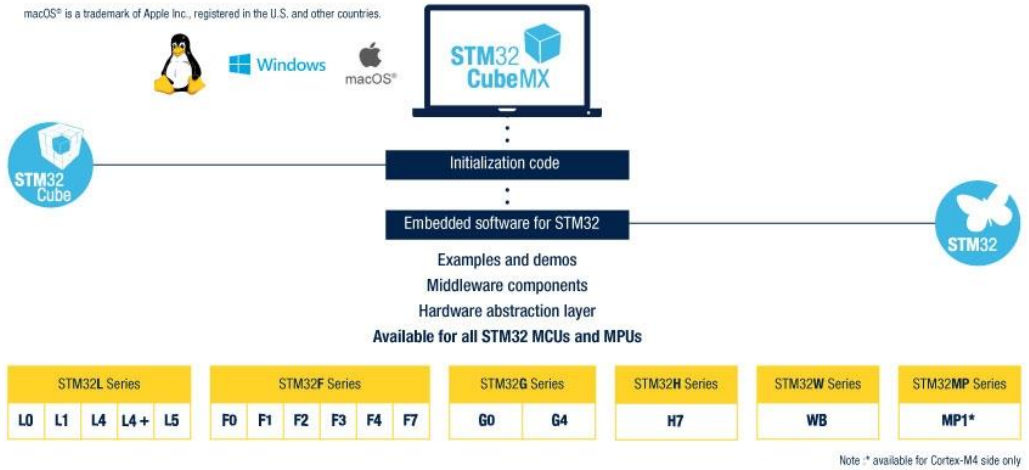
Product line	F_{CPU} (MHz)	Flash (KB)	RAM (KB)	Ethernet I/F IEEE 1588	2x CAN	Camera I/F	SDRAM I/F	Dual Quad-SPI	SAI	SPDIF RX	Chrom-ART Graphic Accelerator™	TFT LCD Controller	MIPI DSI
Advanced lines													
STM32F469[2]	180	512 K to 2056 K	384	•	•	•	•	•	•		•	•	•
STM32F429[2]	180	512 K to 2056 K	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 line	F_{CPU} (MHz)	Flash (KB)	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]

Note: 1. Link Power Management
2. The same devices are also found with embedded HW AES encryption (128/256-bit)

STM32 Tools

- **Hardware Development Tools**
  - STM32 Nucleo boards
  - STM32 Discovery kits
  - STM32 Eval boards
- **STM32 Docs**
  - Datasheet
  - Reference Manual
- **STM32CubeMX**
  - Graphical tool
  - Easy configuration of STM32 microcontrollers peripherals
  - C code generation, compliant with STM32 MCU
- **STM32CubeIDE**
  - C/C++ development platform
  - Code generation, compilation, and debug for STM32 microcontrollers



life.augmented

STM32 boards

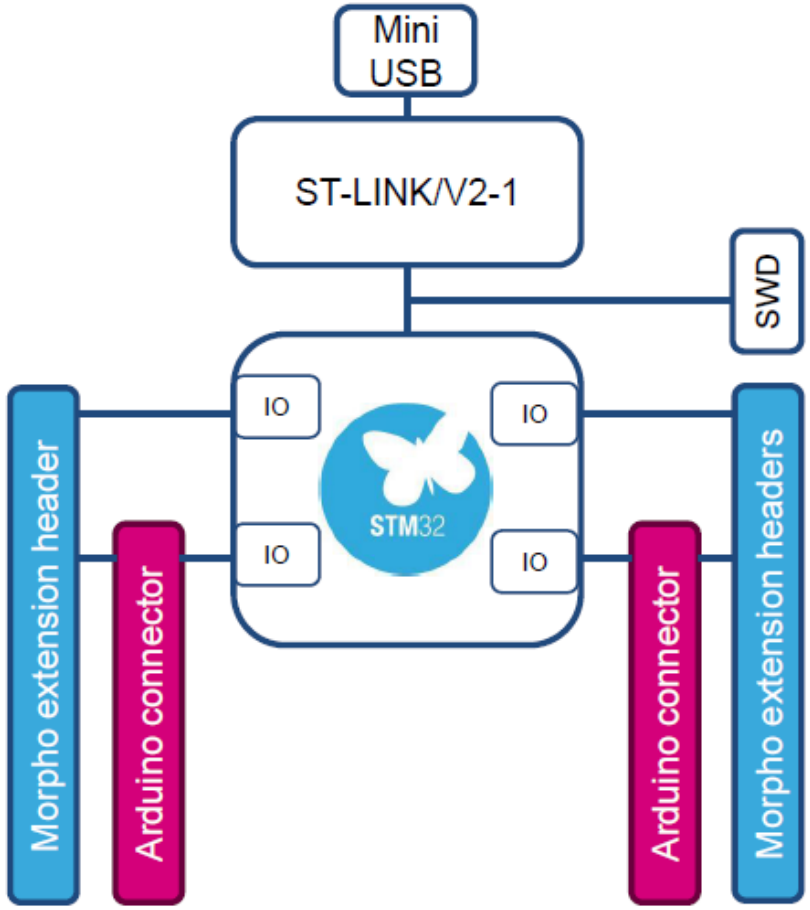
- Developing your own board
- Using existing STMicroelectronics boards
  - STM32 Nucleo boards
  - STM32 Discovery kits
  - STM32 Eval boards

Board	ST-Link	ARDUINO connector	Display
Custom	No		
STM32 Nucleo	Yes (V2.1/V3)	x	
STM32 Discovery	Yes (V2.1/V3)	x	x
STM32 Eval	Yes (V2.1/V3)	x	x

life.augmented

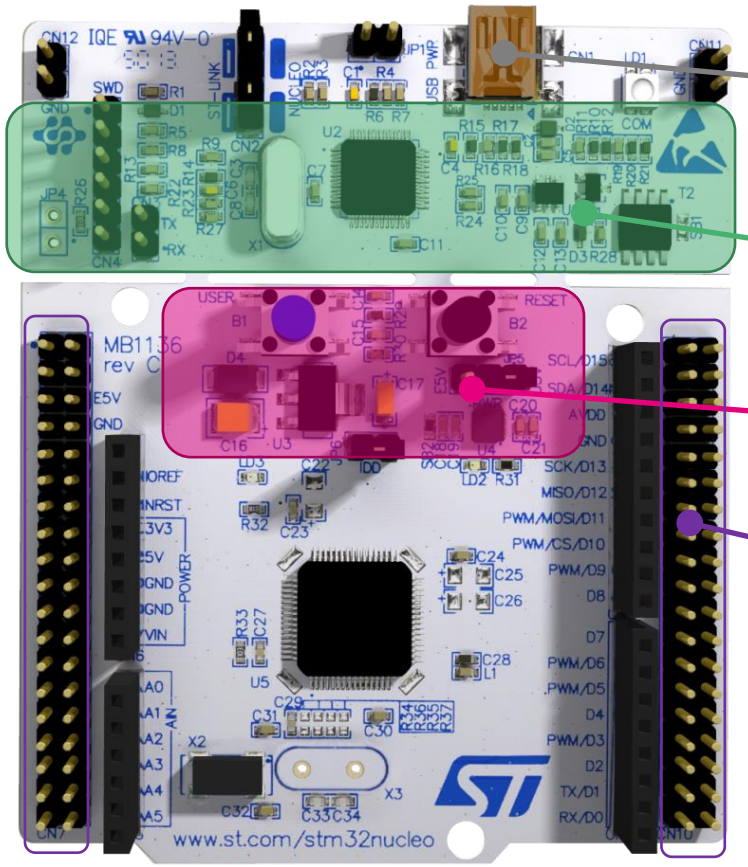
STM32 Nucleo Structure

- Each Nucleo board is based on one of the STM32 MCU product lines
- Two types of extension resources :
  - Arduino Uno v3 connectivity.
  - Morpho headers for easy access to all MCU peripherals.
- Integrated ST-LINK/V2-1 debugger and programmer :
  - supports drag-and-drop flash programming.
  - can target on-board STM32 or external STM32-based application.



life.augmented

STM32 Nucleo key assets



Flexible board power supply
Through USB or external source

Integrated ST-Link/V2-1
Mass storage device flash programming

2 push buttons, 2 color LEDs

Direct access to all STM32 I/Os:
through Morpho extension headers

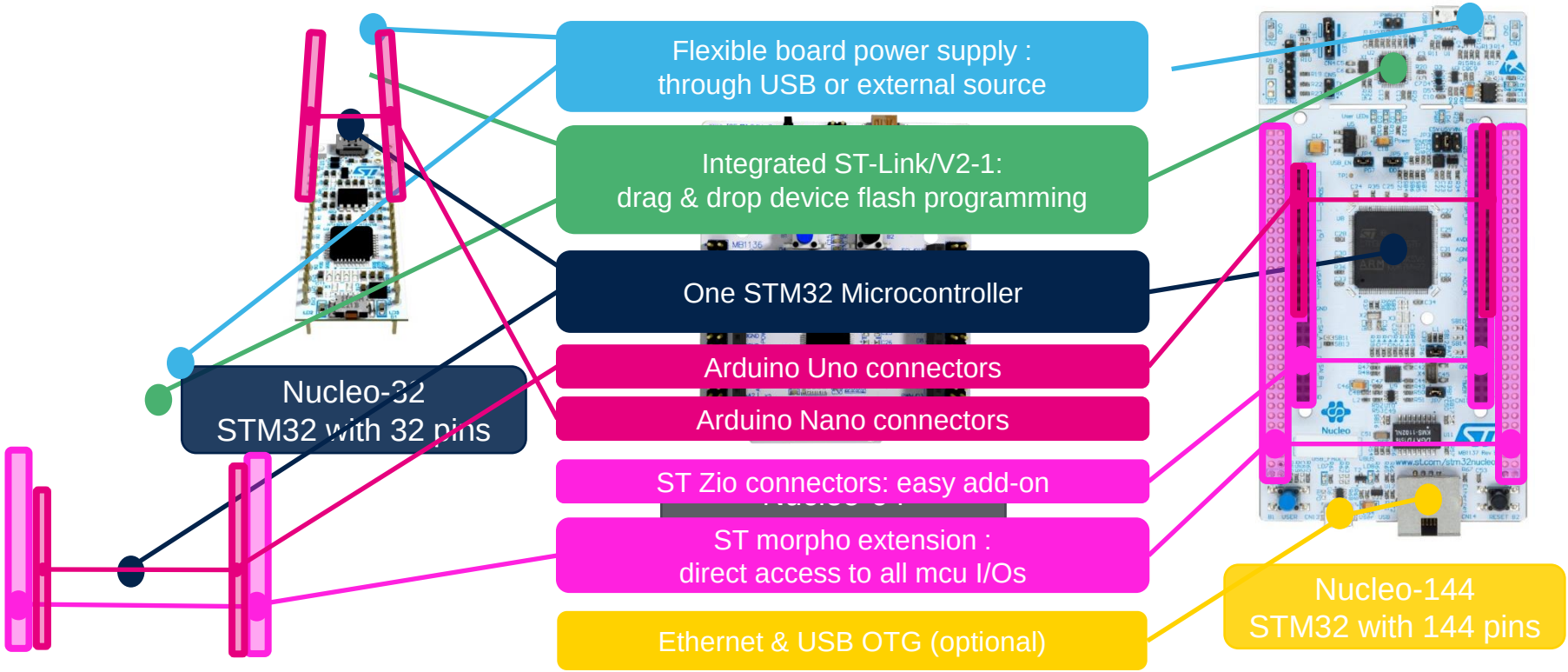
Flexible prototyping

Unlimited expansion capabilities

Simply expose the whole STM32 portfolio to the communities

life.augmented

Enlarging the STM32Nucleo family to cover whole STM32 portfolio

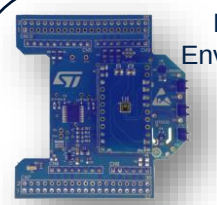


STM32 complete product range from ultra-low power to high performance

life.augmented

Examples of Nucleo Expansion boards

Sense	Connect	Power Drive	Move Actuate	Translate
Motion & environmental sensors	BLE	Power management	Motor drive	Audio amplifier
Proximity sensor	Wi-Fi	LED boost	Actuator	OpAmp
Microphone	Sub-GHz			
	NFC			


Motion & Environmental sensors


Proximity & Light sensor


BLE


WiFi 802.11 b/g/n


Dynamic NFC


Sub-1GHz


LED driver


Motor driver


Audio Microphones

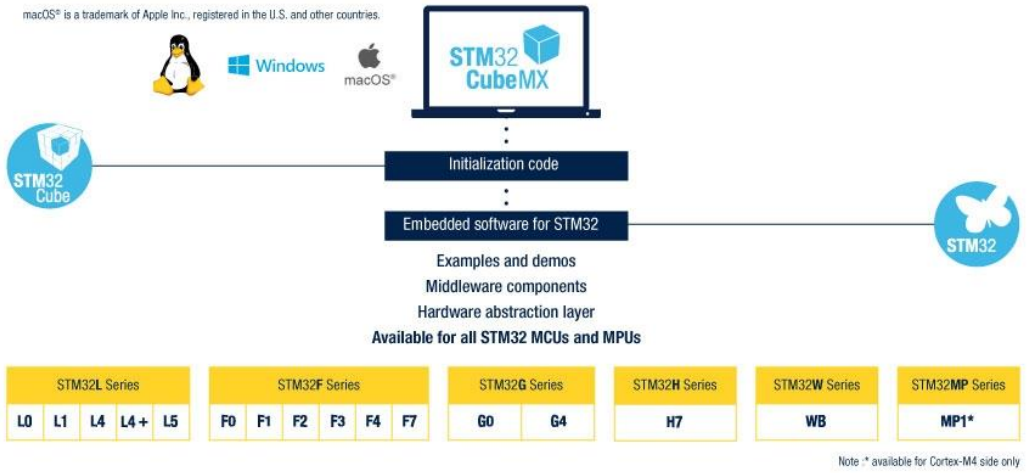

Security

For more information on STM32 Nucleo expansion boards:http://www.st.com/en/ecosystems/stm32-nucleo-expansion-boards.html

life.augmented

Outline

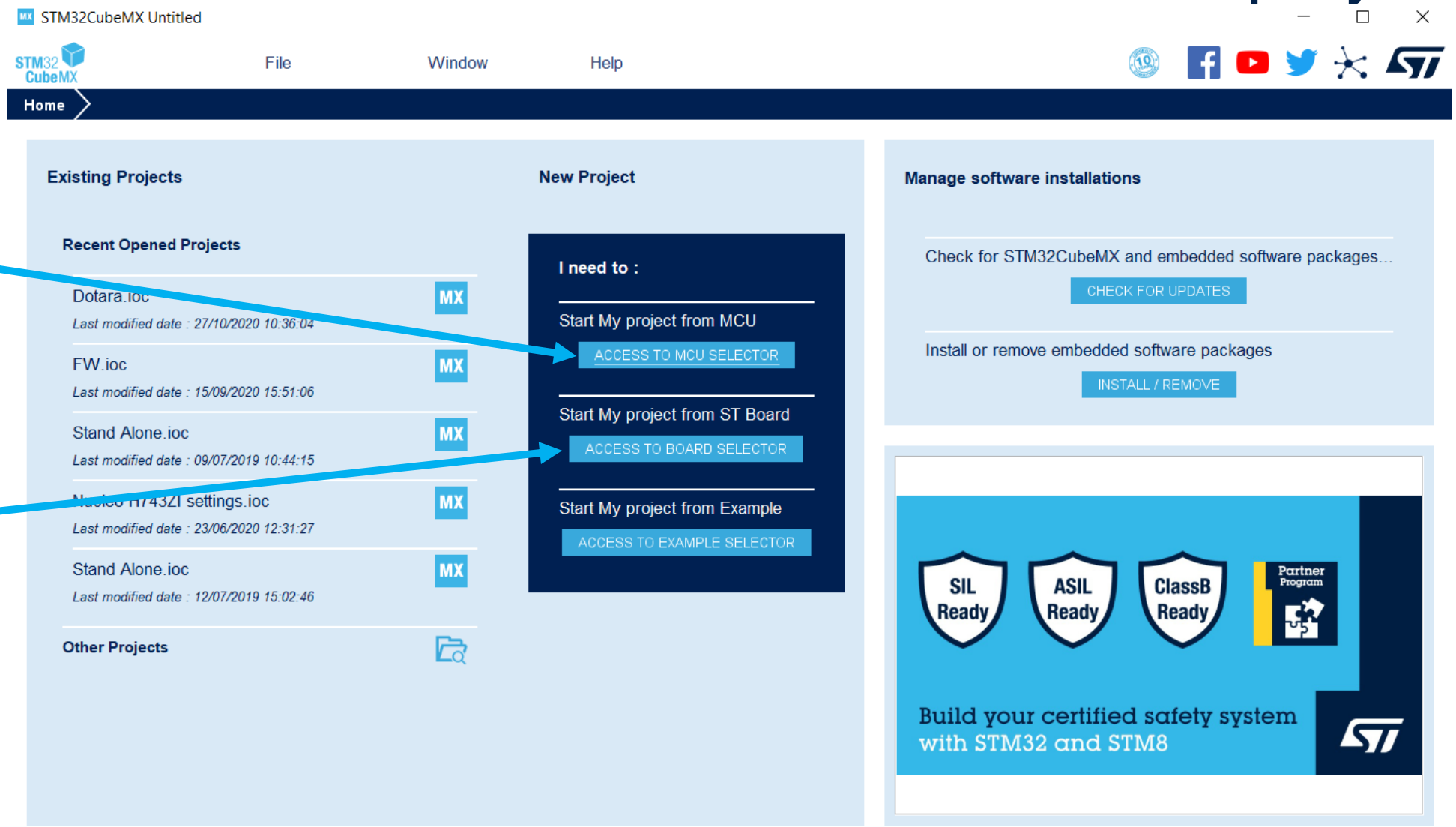
- Updater settings and manage embedded software packages
- Create new project (standalone MCU or board)
- Pinout
- Peripheral configuration
- Clock tree
- NVIC configuration
- Additional Software
- Project generation



Create new project

MCUs selector

Board selector

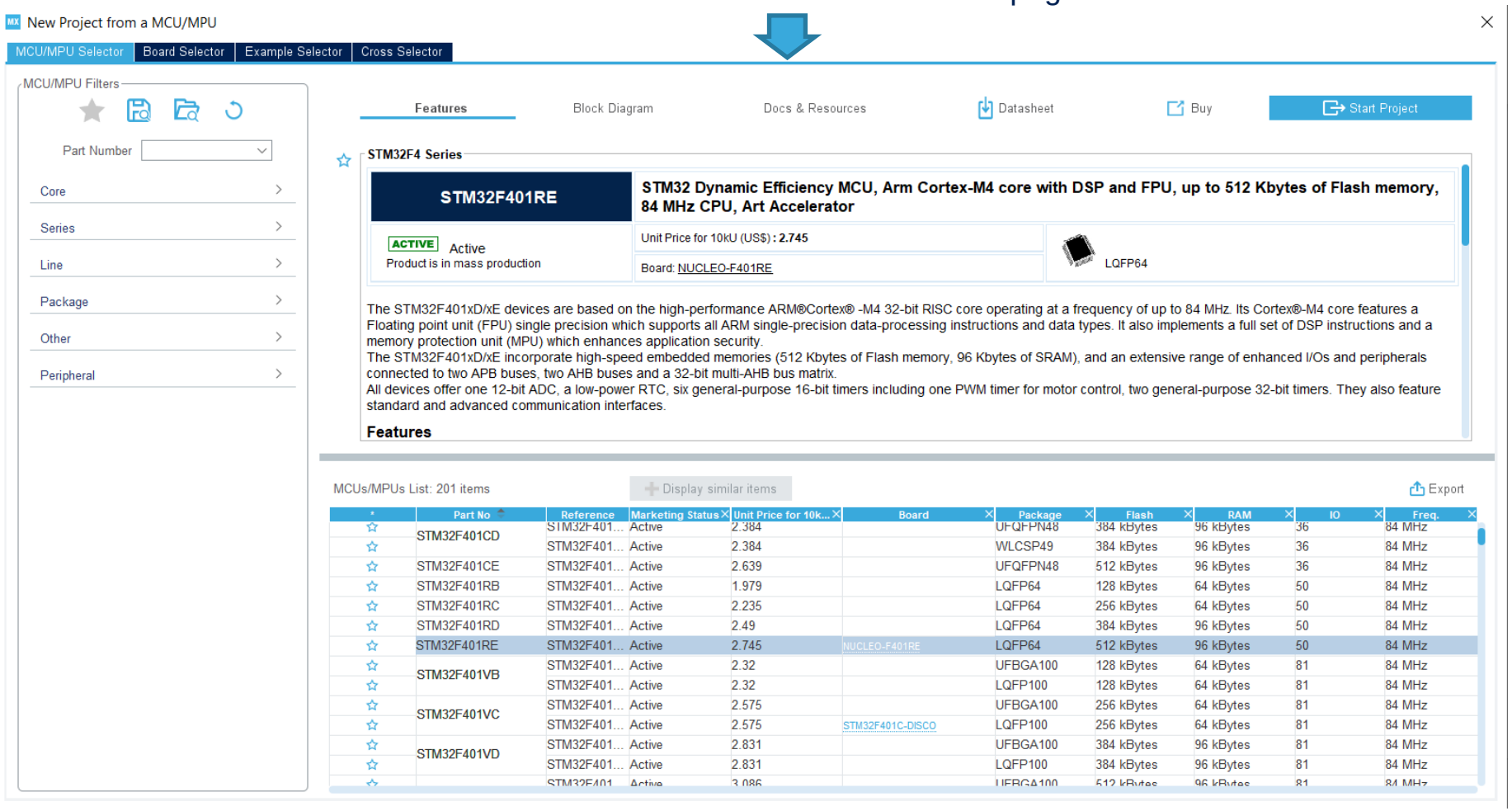


Create new project – MCUs selector

You can filter MCUs with:

- Part Number
- Core
- Series
- Lines
- Peripherals
- …….

Link to Documentation and web page

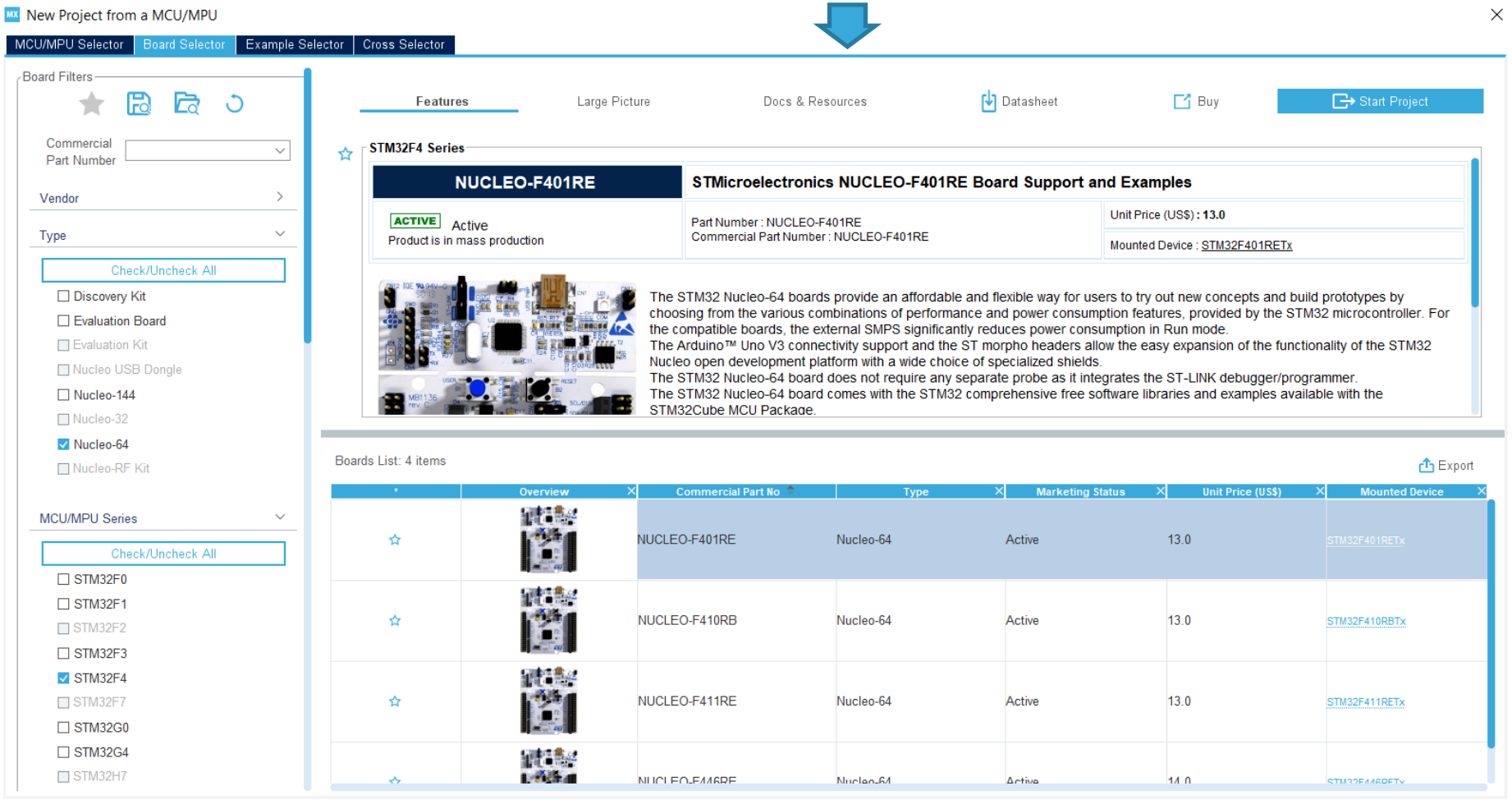


Create new project – Board selector

Link to Documentation and web page

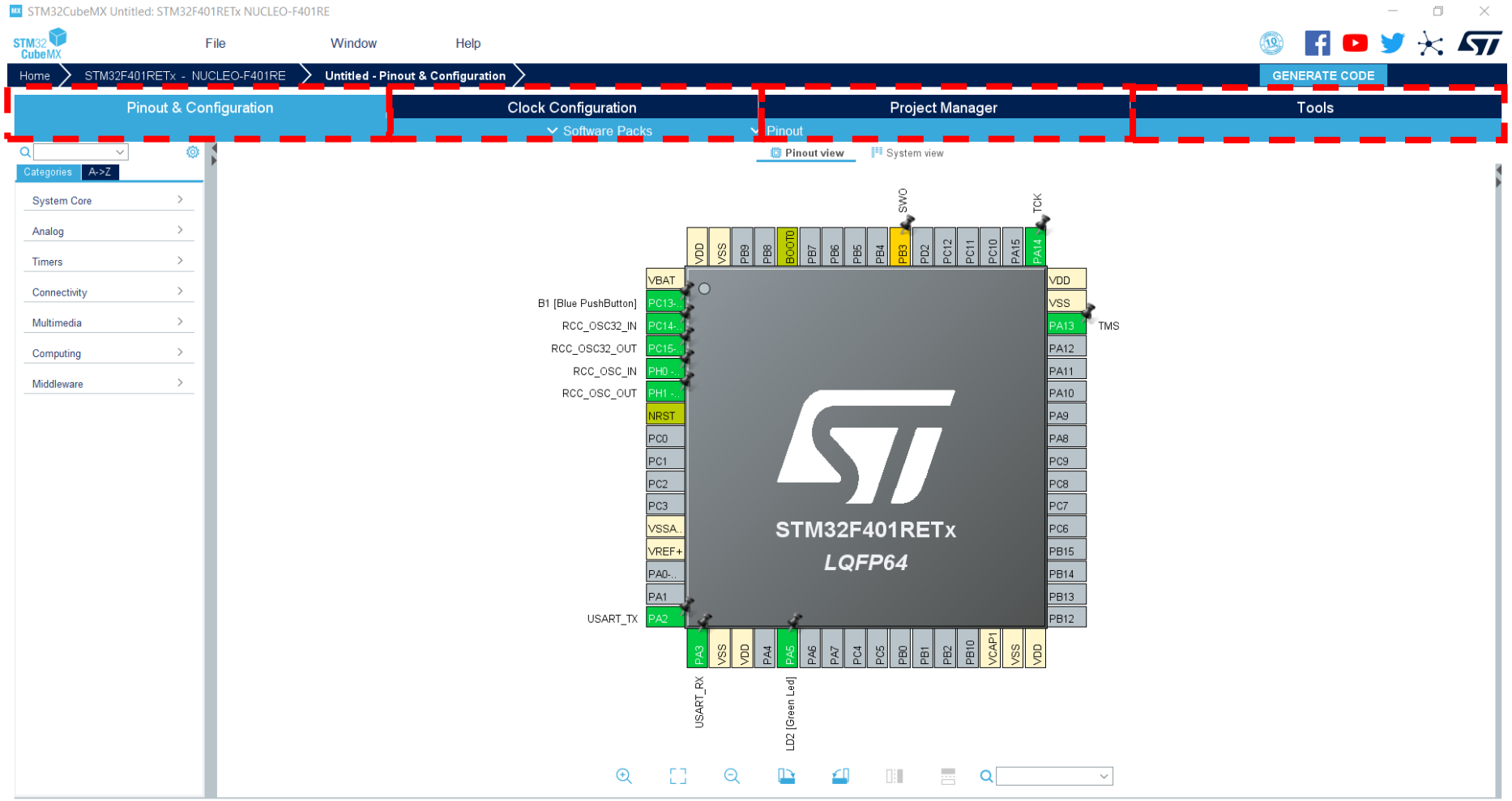
You can filter Boards with:

- Part Number
- Vendor
- Type
- MCU series



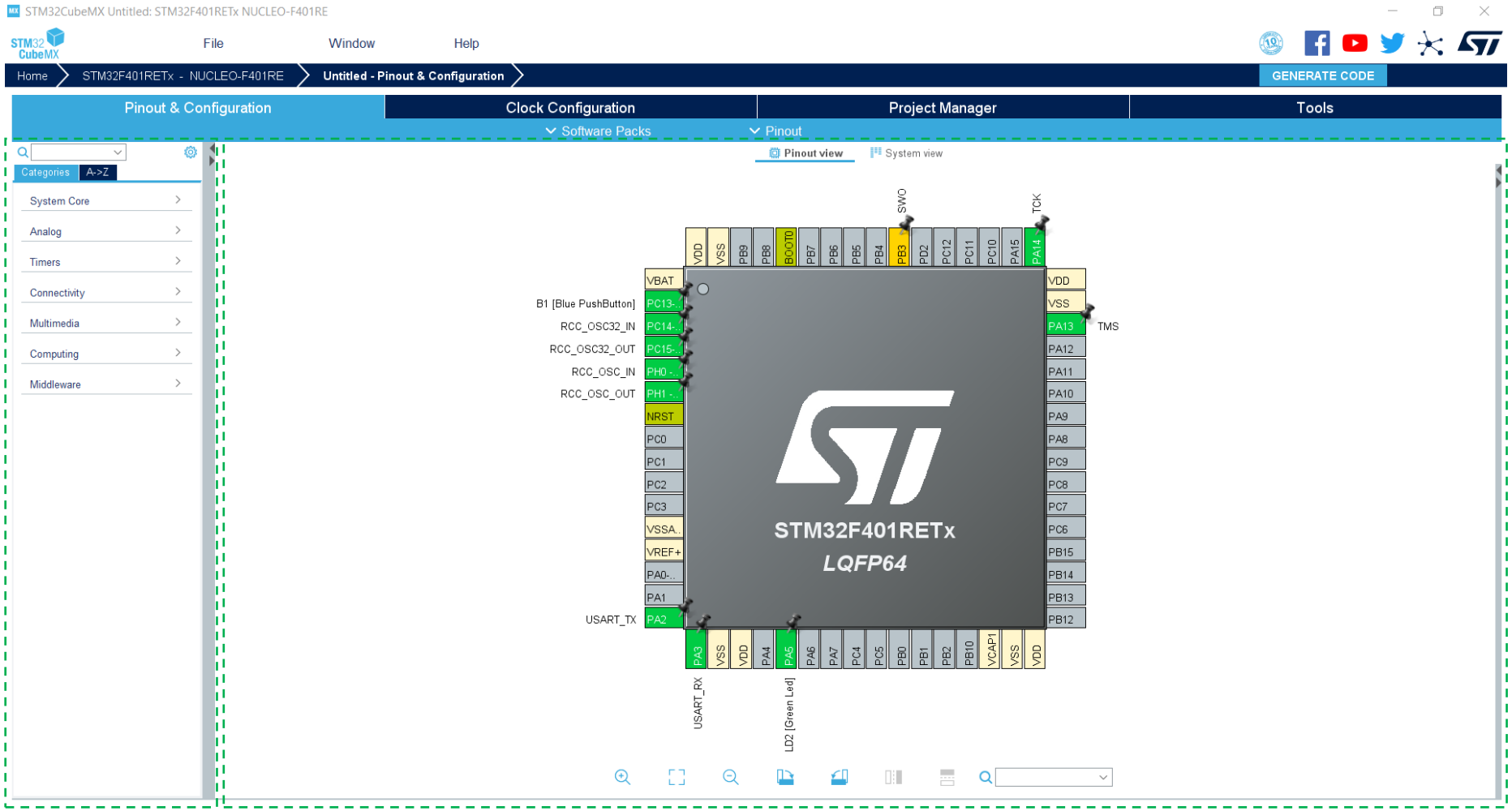
CubeMX view

- Pinout & Configuration
- Clock Configuration
- Project Manager
- Tools



Pinout & Configuration view

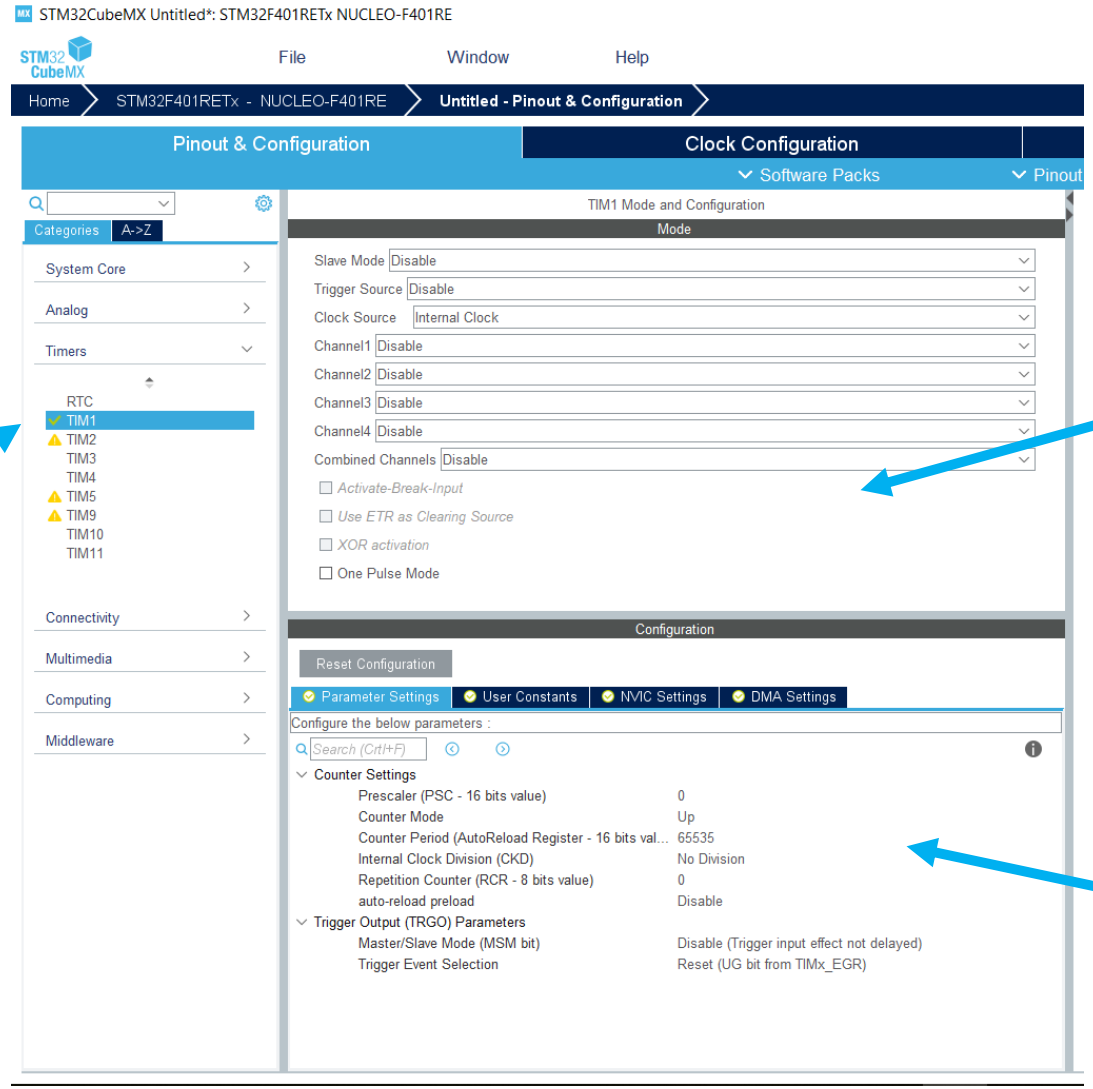
Peripherals Configuration section



MCU pinout section

Peripherals Configuration section (set parameters)

From the **Peripherals Configuration section**, the user can select the peripherals from a list and configure the peripheral modes required for the application. STM32CubeMX assigns and configures the pins accordingly.



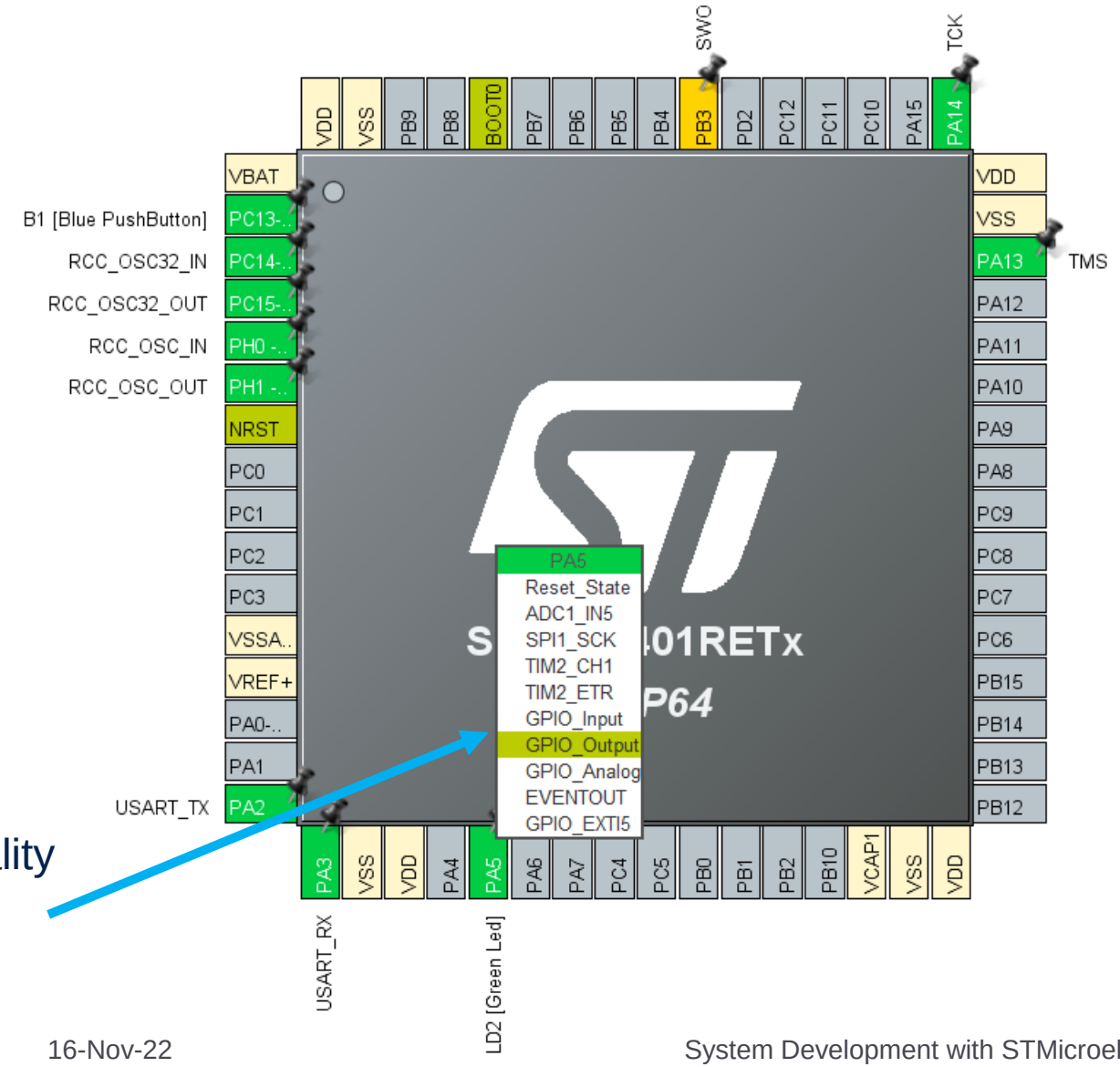
Peripheral selection

Peripheral mode selection

Peripheral parameters setting

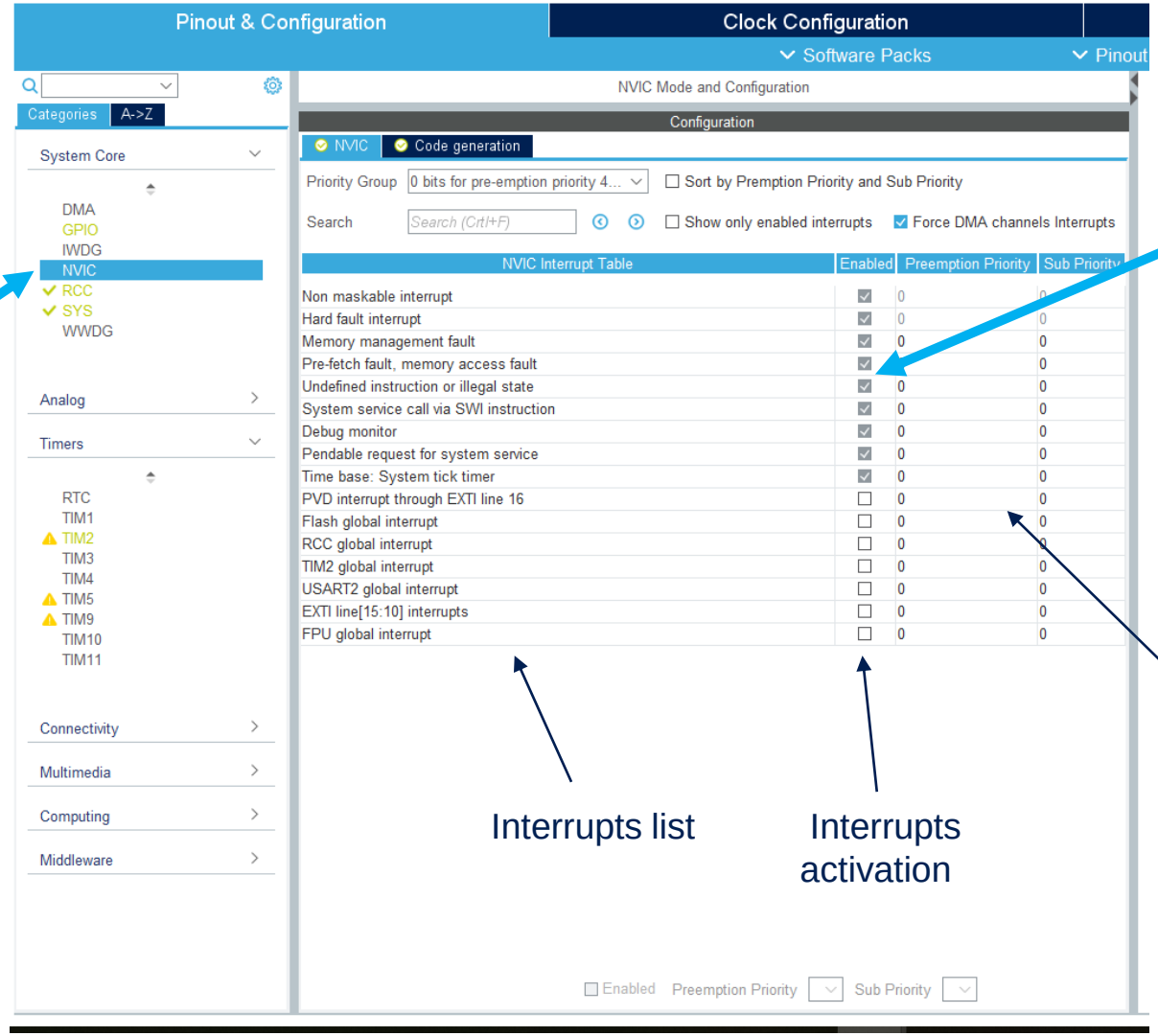
MCU pinout section

For more advanced users, it is also possible to directly map a peripheral function to a physical pin using the Chip view. The signals can be locked on pins to prevent STM32CubeMX conflict solver from moving the signal to another pin.



Pin functionality assignment

Peripherals Configuration section (set interrupts)



Interrupts Configuration

System Interrupts

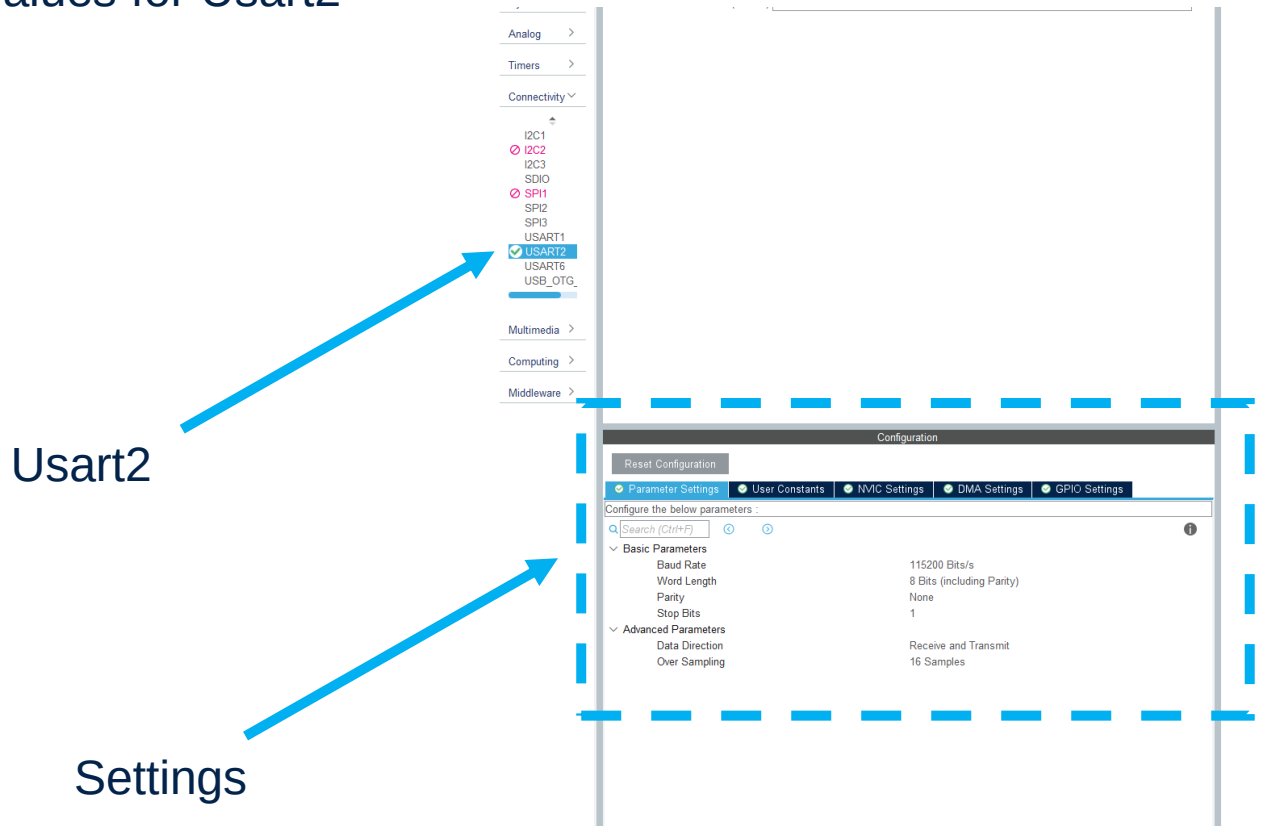
Interrupts list

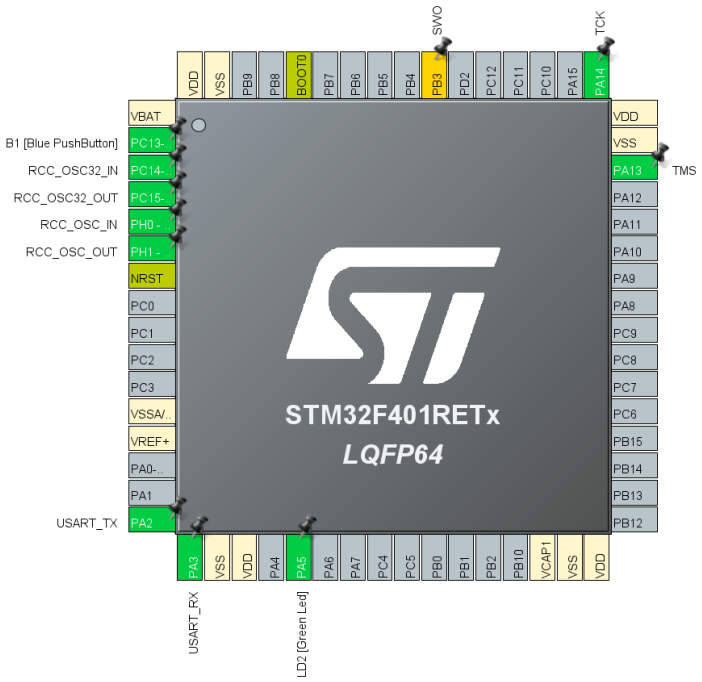
Interrupts activation

Preemption and sub priority settings

Hello World on uart (1/2)

- On CubeMx, check Parameter Settings values for Usart2





life.augmented

Hello World on uart (2/2)

```
56 /* USER CODE BEGIN PFP */
57
58 int _write(int fd, char* ptr, int len) {
59     HAL_UART_Transmit(&huart2, (uint8_t *) ptr, len, HAL_MAX_DELAY);
60     return len;
61 }
```

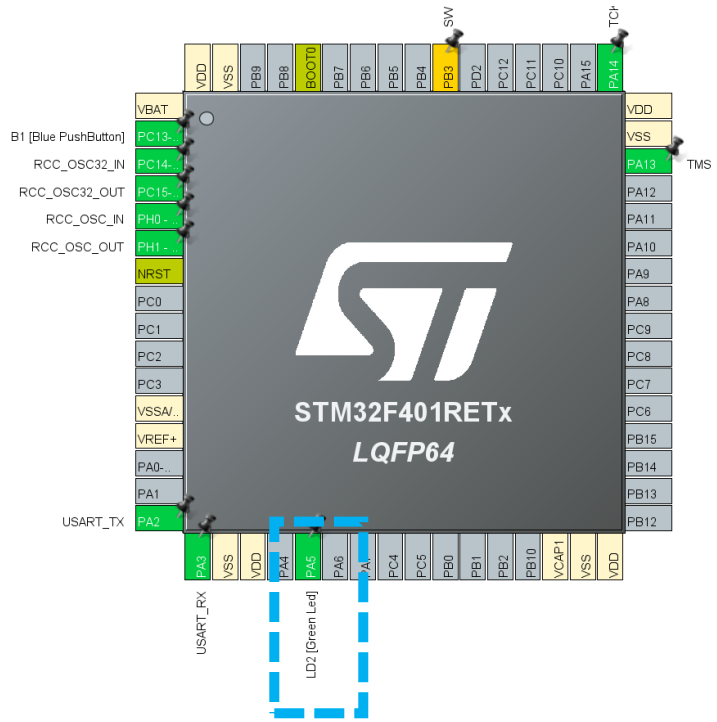
```
/* USER CODE BEGIN 2 */

printf("hello\r\n");
```

- On STM32CubeIDE:
  - Define the _write function
  - Write the printf function

Led blinking, solution 1 (not precise)

- On CubeMx, verify PA5 is set as GPIO_output



- Toggle the pin status
- Add a delay

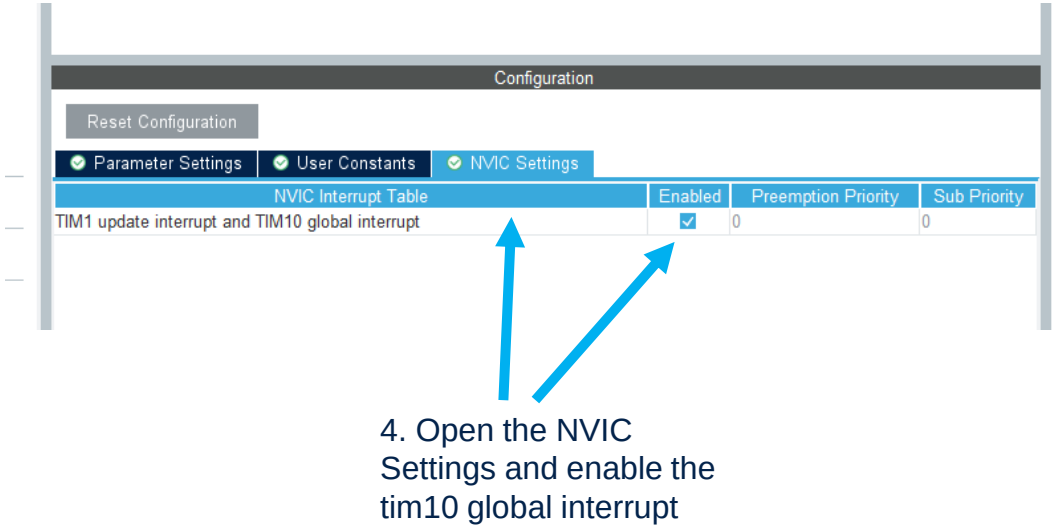
```
90  /* USER CODE BEGIN 2 */
91
92  /* USER CODE END 2 */
93
94  /* Infinite loop */
95  /* USER CODE BEGIN WHILE */
96  while (1)
97  {
98      HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
99      HAL_Delay(500);
100
101   /* USER CODE END WHILE */
102
103   /* USER CODE BEGIN 3 */
104 }
105 /* USER CODE END 3 */
```

life.augmented

Led blinking, with timer and interrupt (1/2)

- Enable timer 10 and set its params

- Enable the related interrupt



life.augmented

Led blinking, with timer and interrupt (2/2)

```
55 /* USER CODE BEGIN PFP */
56
57 volatile uint8_t time_elapsed = 0;
58
59 /* USER CODE END PFP */
```

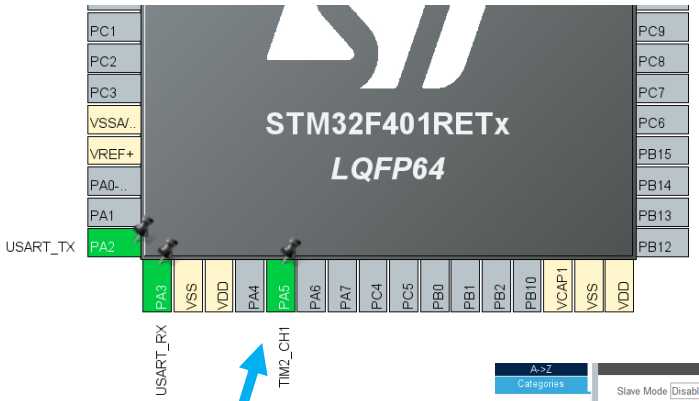
```
96   /* USER CODE BEGIN 2 */
97
98   HAL_TIM_Base_Start_IT(&htim10);
99
100  /* USER CODE END 2 */
```

```
262 /* USER CODE BEGIN 4 */
263
264 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
265 {
266     if(htim == &htim10)
267     {
268         time_elapsed = 1;
269     }
270 }
271
272
273 /* USER CODE END 4 */
```

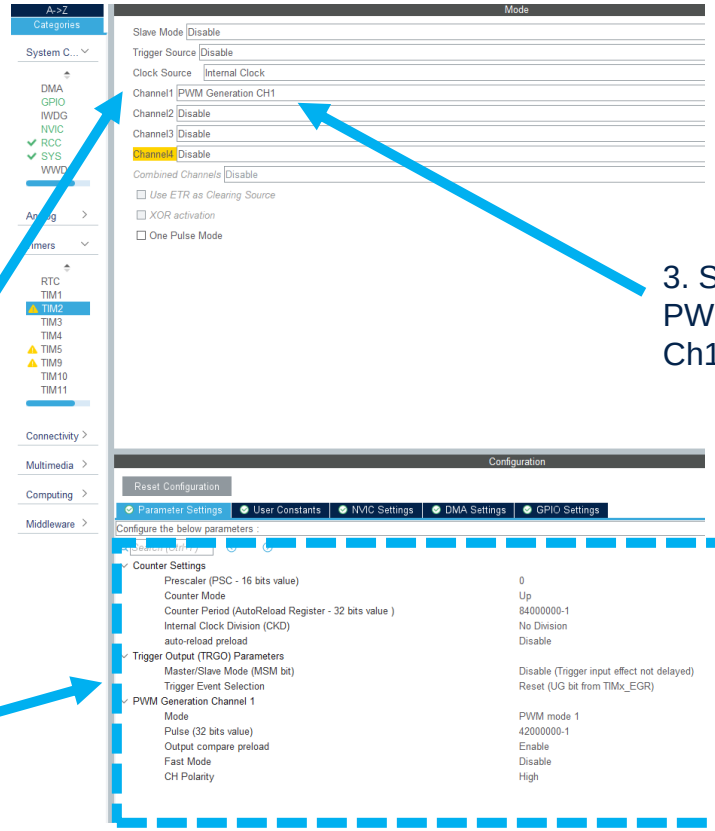
```
100  /* USER CODE END 2 */
101
102  /* Infinite loop */
103  /* USER CODE BEGIN WHILE */
104  while (1)
105  {
106      if(time_elapsed)
107      {
108          time_elapsed = 0;
109          HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
110      }
111
112    /* USER CODE END WHILE */
```

- Declare a global flag
- Make the timer 10 start in interrupt mode
- Redefine a callback function related to the elapsed time and set the flag
- Manage the flag in the main loop

Led blinking, with PWM



1. Set PA5 as TIM2_CH1



2. Enable Clock Source

3. Set channel 1 as PWM Generation Ch1

4. Adjust Counter period and Pulse values

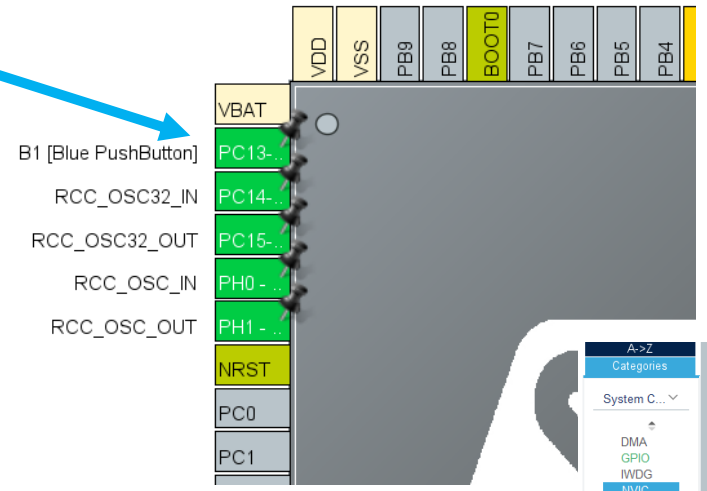
```
104
105  HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
106
107  /* USER CODE END 2 */
```

5. In the c code, let the timer 2 pwm start
Remember to remove all the references to LD2

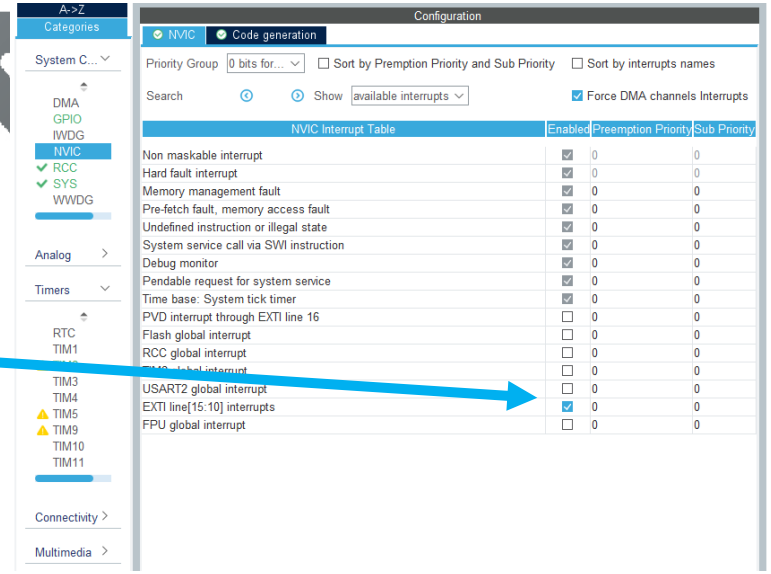
life.augmented

Button Interrupt (1/2)

1. Check the PC13 pin is set as GPIO_EXTI13



2. Enable the EXTI line[15:10] interrupts



Button Interrupt (2/2)

```
46 /* USER CODE BEGIN PV */
47 volatile uint8_t button_pressed = 0;
```

```
298 /* USER CODE BEGIN 4 */
299
300 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
301 {
302   /* Prevent unused argument(s) compilation warning */
303   if(GPIO_Pin == GPIO_PIN_13)
304   {
305       button_pressed = 1;
306   }
307 }
```

```
109   /* Infinite loop */
110   /* USER CODE BEGIN WHILE */
111   while (1)
112   {
113       if(button_pressed)
114       {
115           button_pressed = 0;
116           printf("yes, you pressed a button!\r\n");
117       }
118
```

- Declare a global flag
- Redefine a callback function related to the pressed button and set the flag
- Manage the flag in the main loop