STM32L4 project using: STM32L476-Discovery, CUBE MX and AC6 (SW4STM32 - System Workbench)

WWW.EMCU.IT

life.augmented

Accelerating Your Success™

HW and SW tools

HW:

- **STM32L476-Discovery**

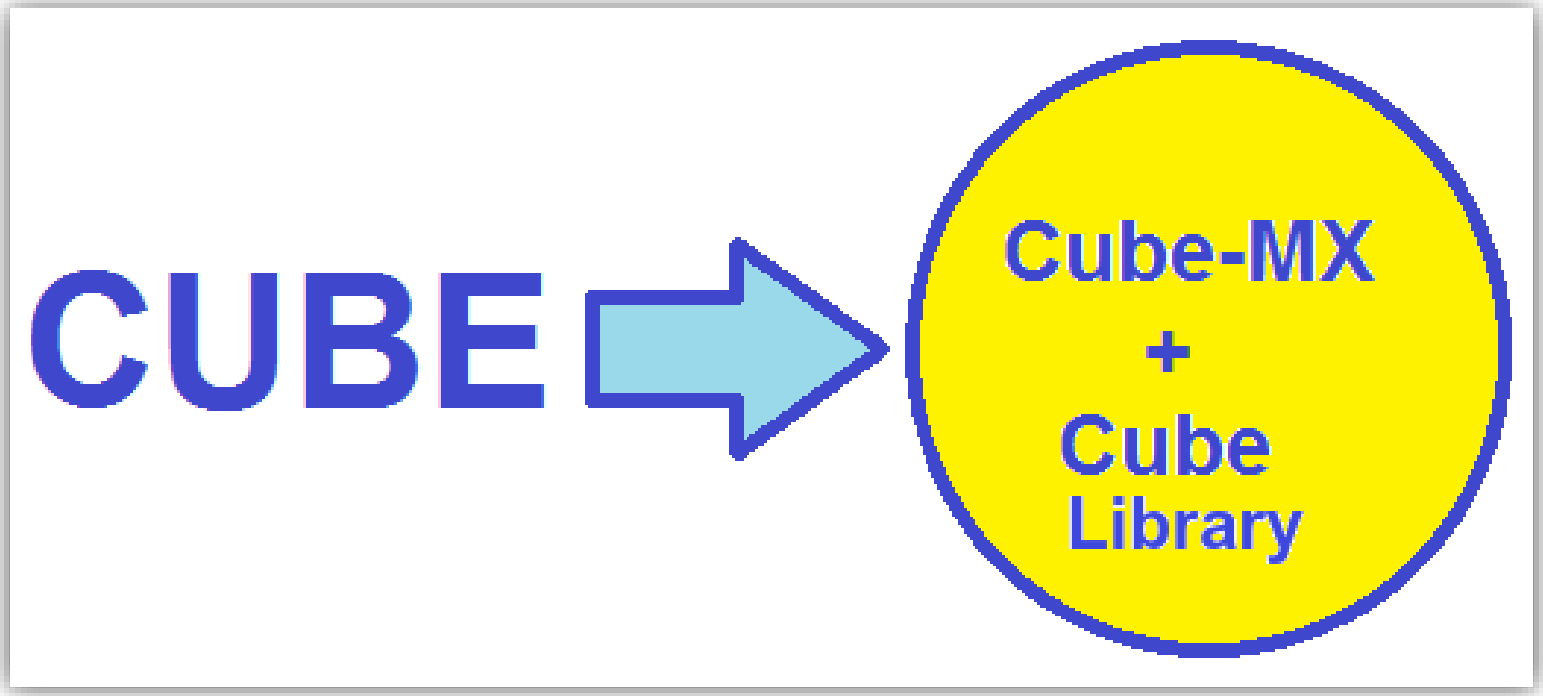
SW:

- **AC6** (SW4STM32 - System Workbench)
- **CUBE MX**
- **STM32L4 HAL Library**
- **See the instructions how to install the SW here.**

CUBE

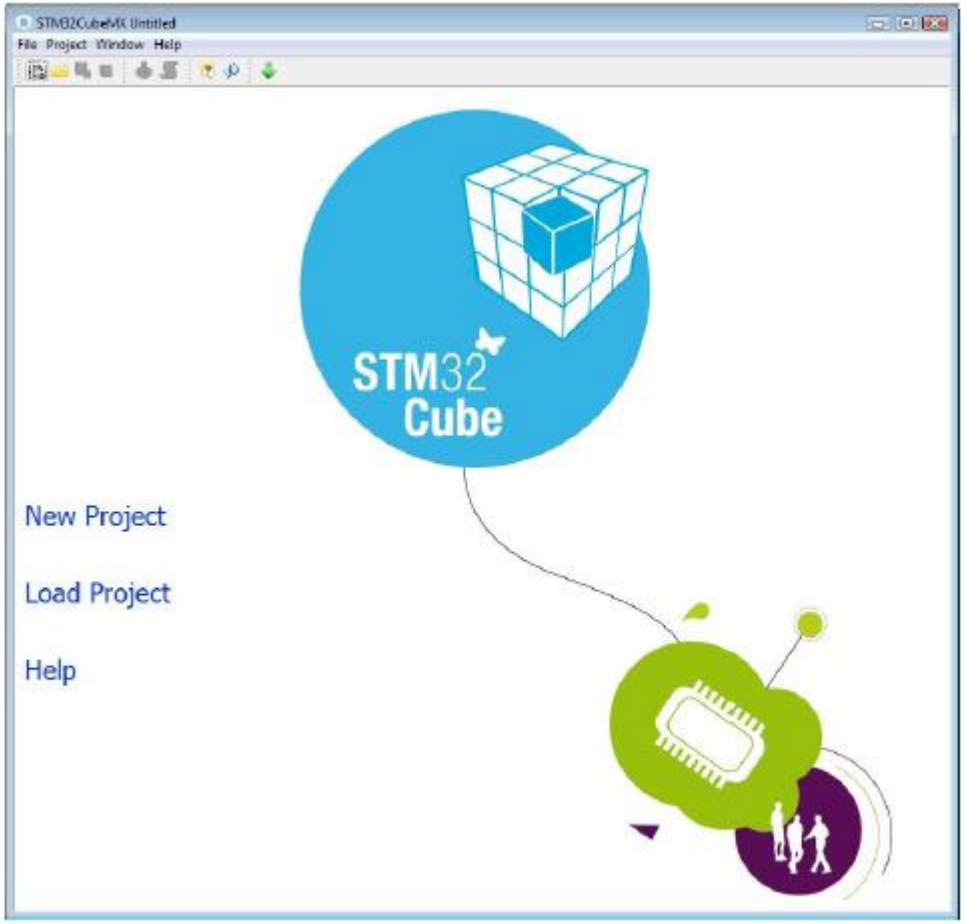
CUBE

Introduction to CubeMX 1/3

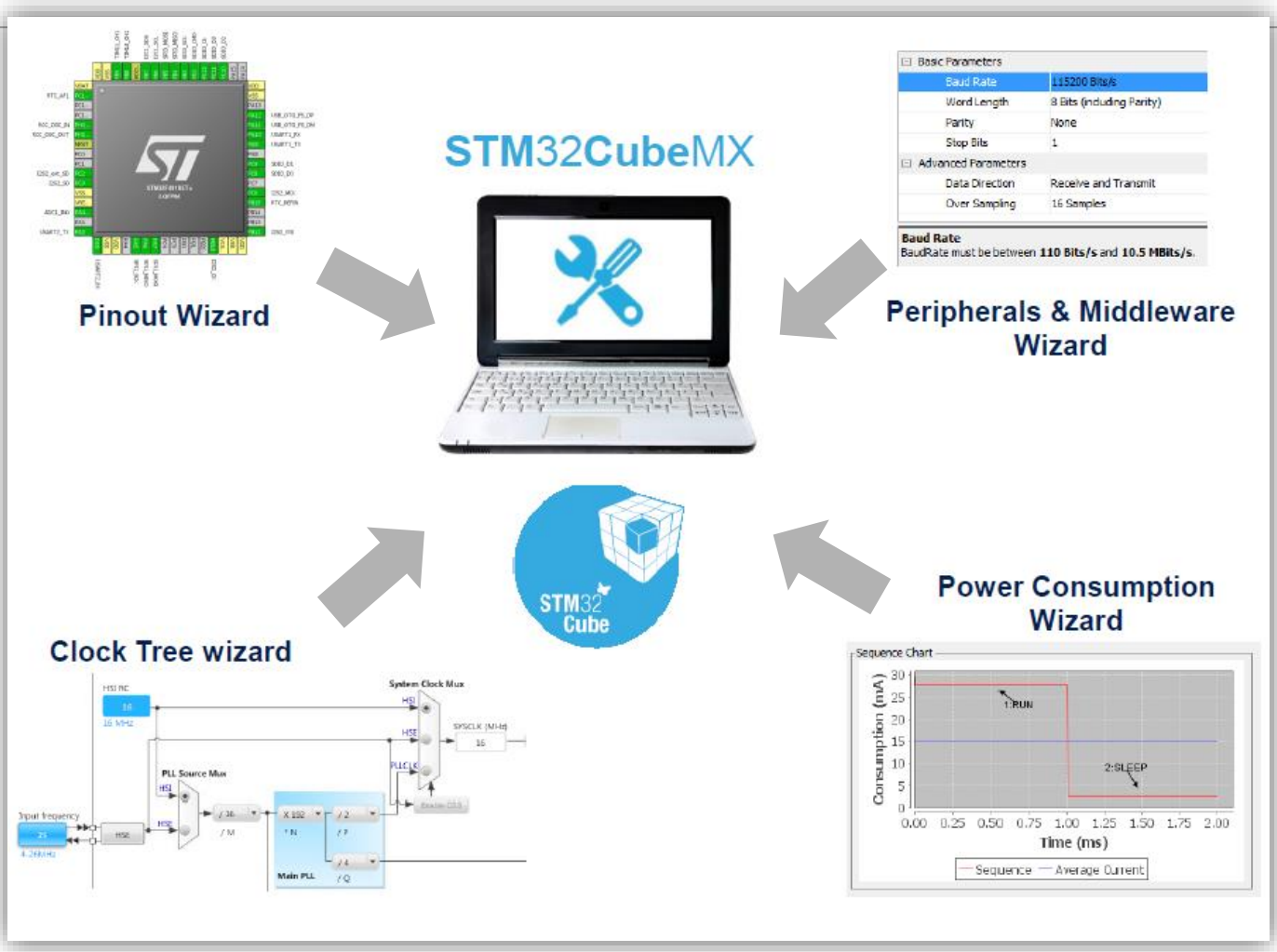


Introduction to CubeMX 2/3

- MCU selector
- Pinout configuration
- Clock tree initialization
- Peripherals and middleware parameters
- Code generation
- Power consumption calculator

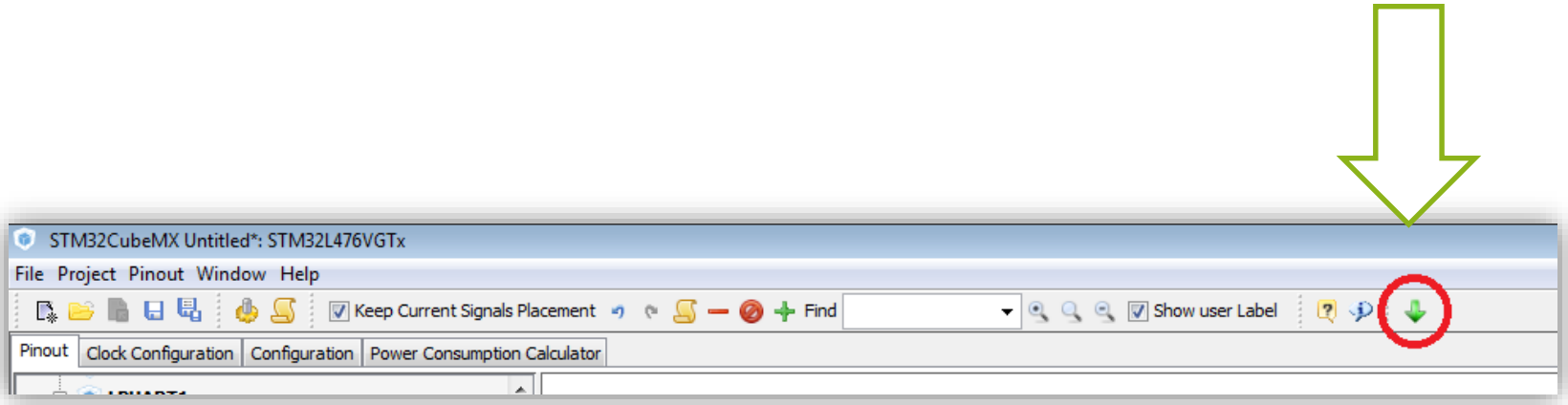


Introduction to CubeMX 3/3

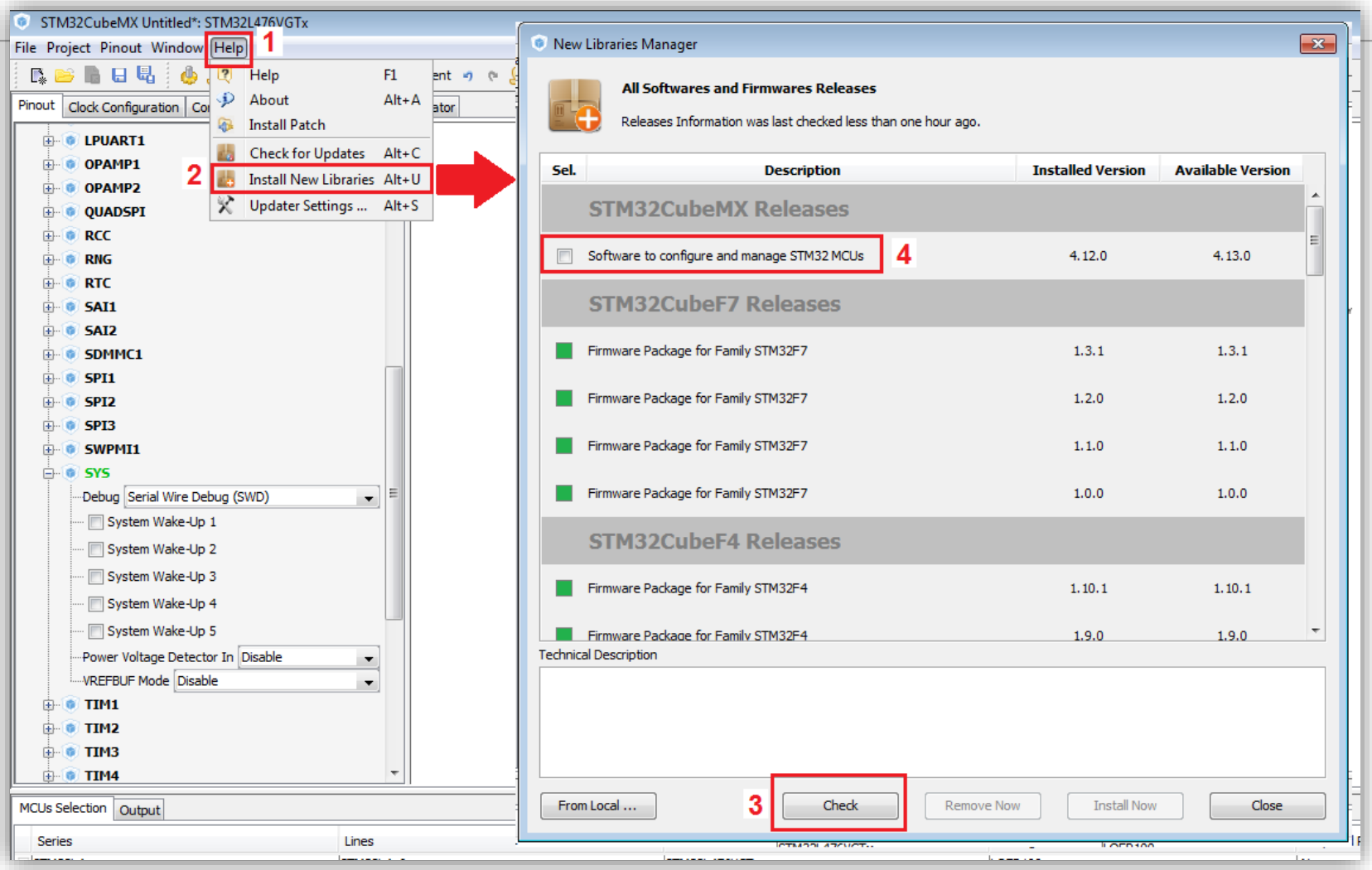


CubeMX request update 1/2

The **green arrow** indicate that are presents some updates.

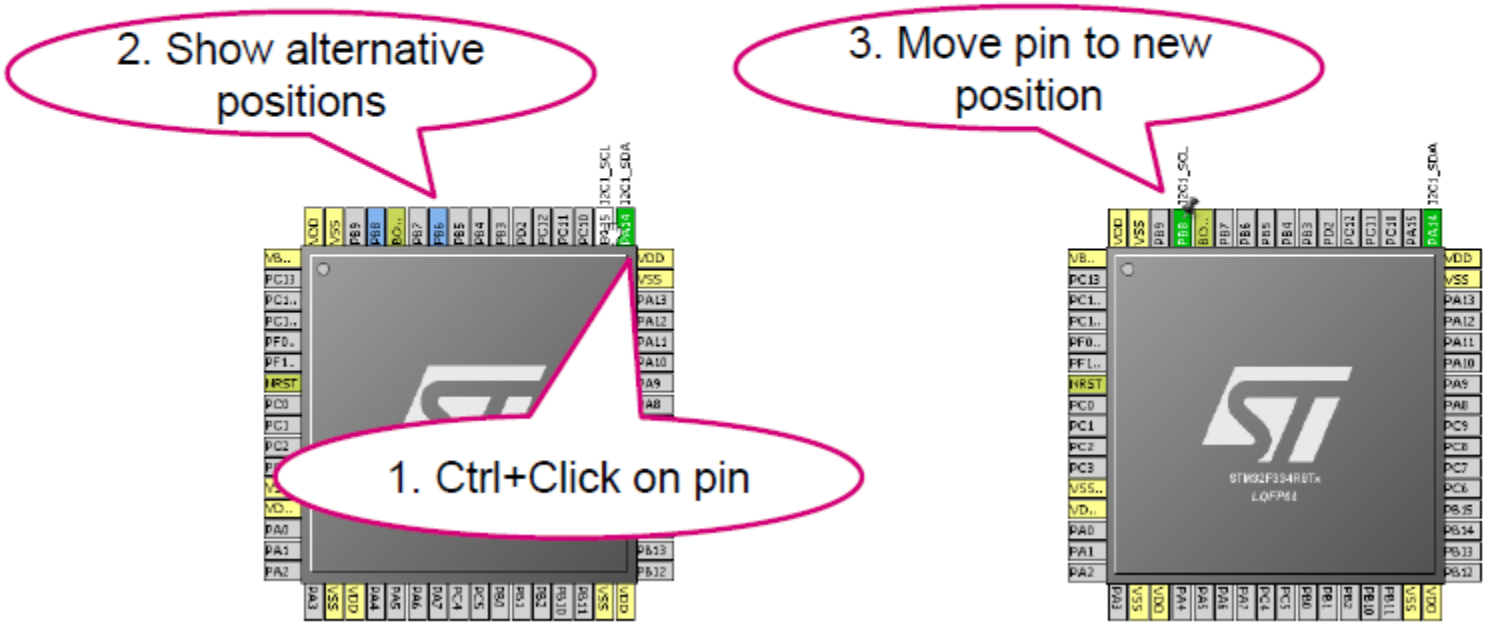


CubeMX request update 2/2



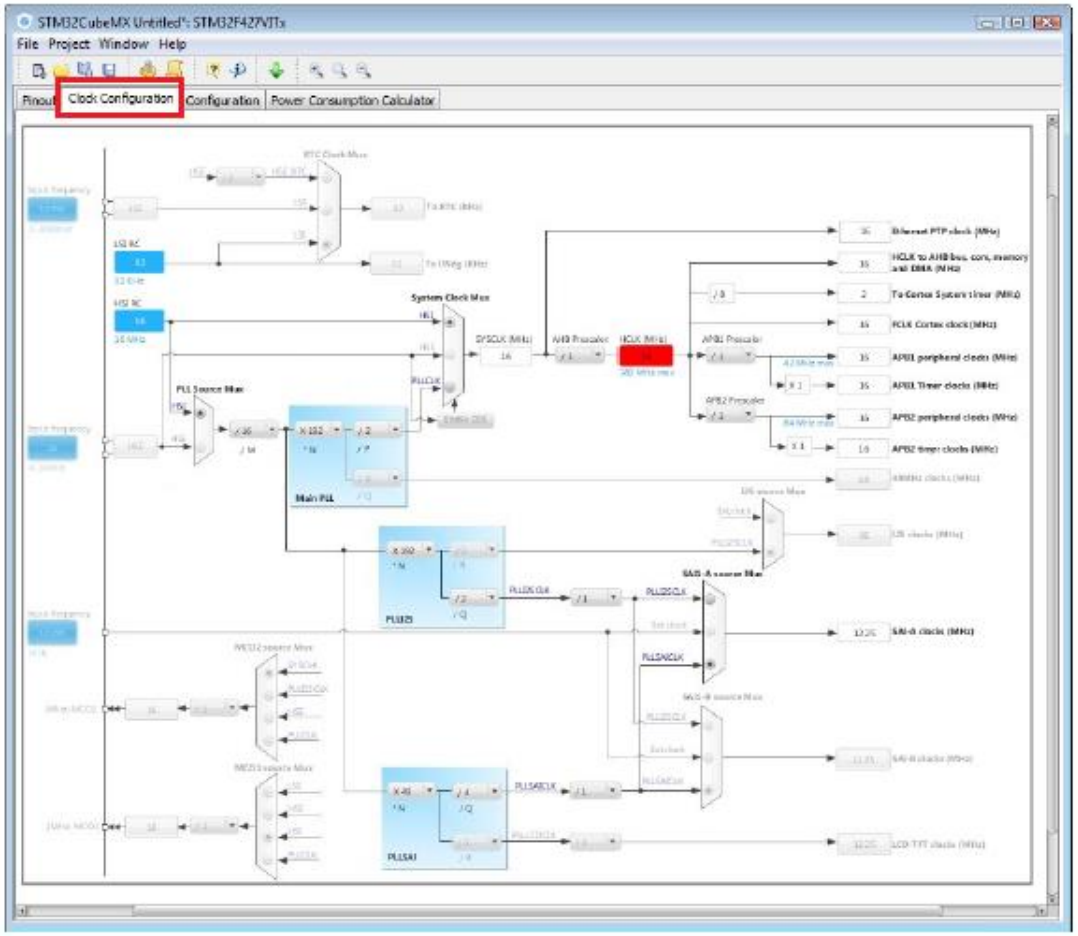
CubeMX: Pinout configuration

- Signals can be set/moved directly from the pinout view
  - To see alternate pins for a signal Ctrl+Click on the signal, you can then drag and drop the signal to the new pin (keep pressing the Ctrl key)



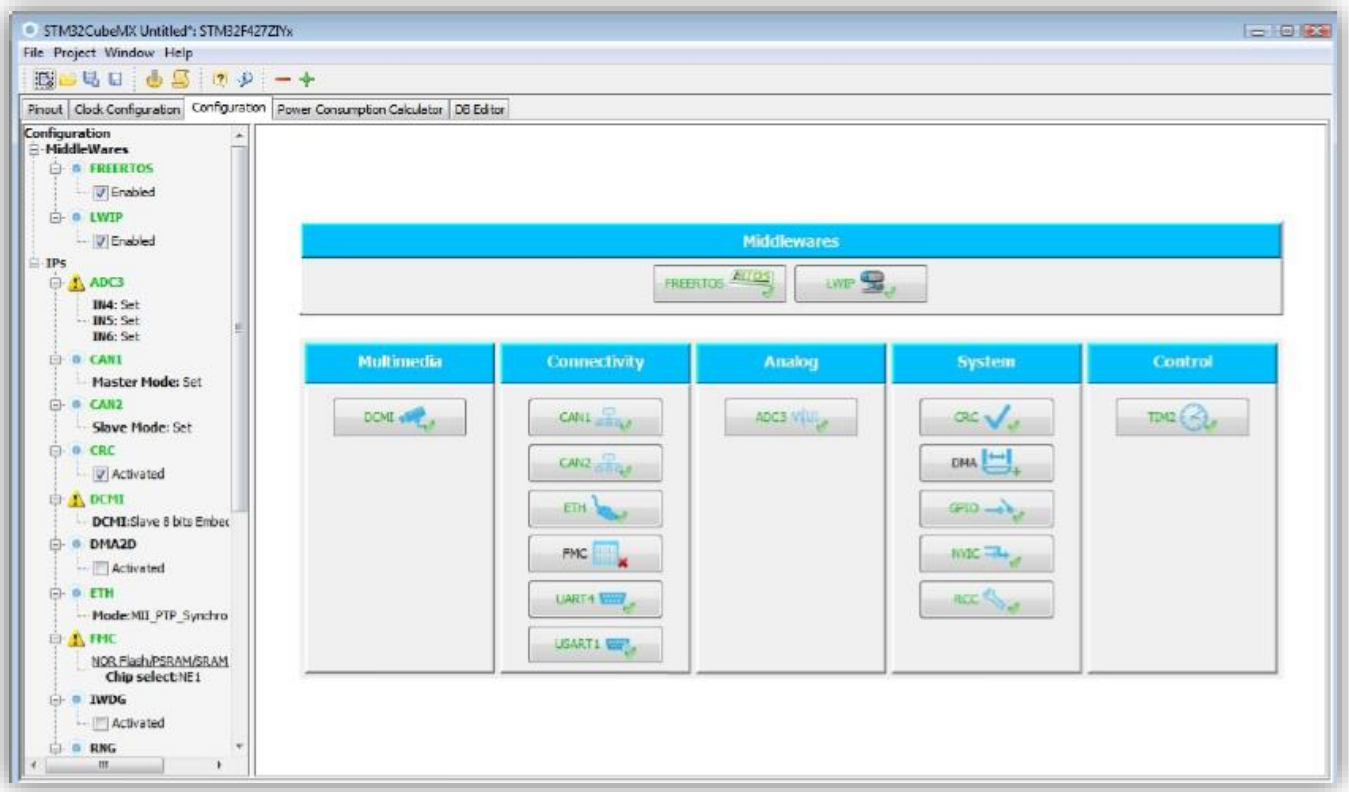
CubeMX: Clock tree

- Immediate display of all clock values
- Management of all clock constraints
- Highlight of errors



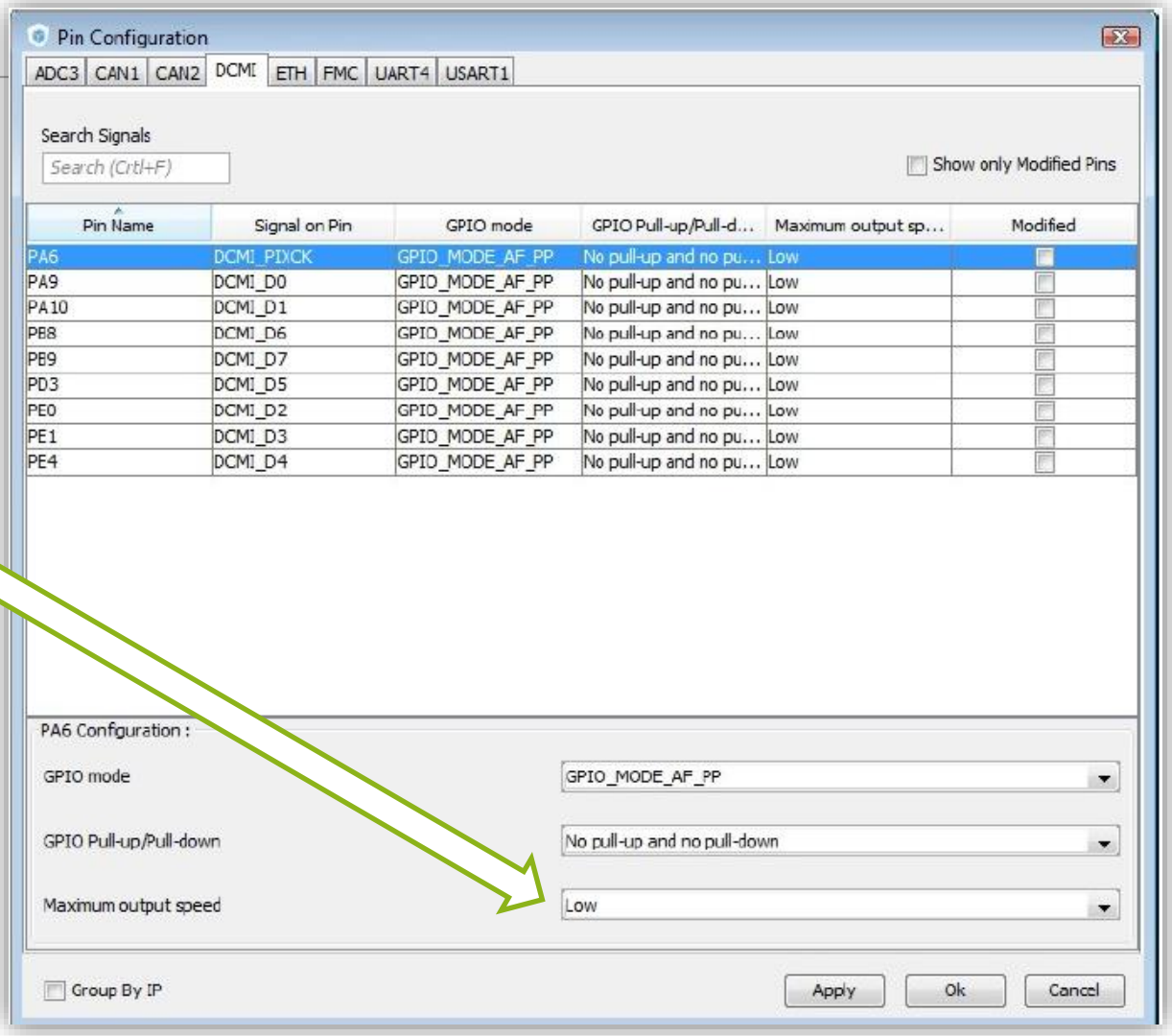
CubeMX: Peripheral and middleware configuration

- Global view of used peripherals and middleware
- Highlight of configuration errors
  - + Not configured
  - v OK
  - x Error
- Read only tree view on the left with access to IPs / Middleware having no impact on the pinout

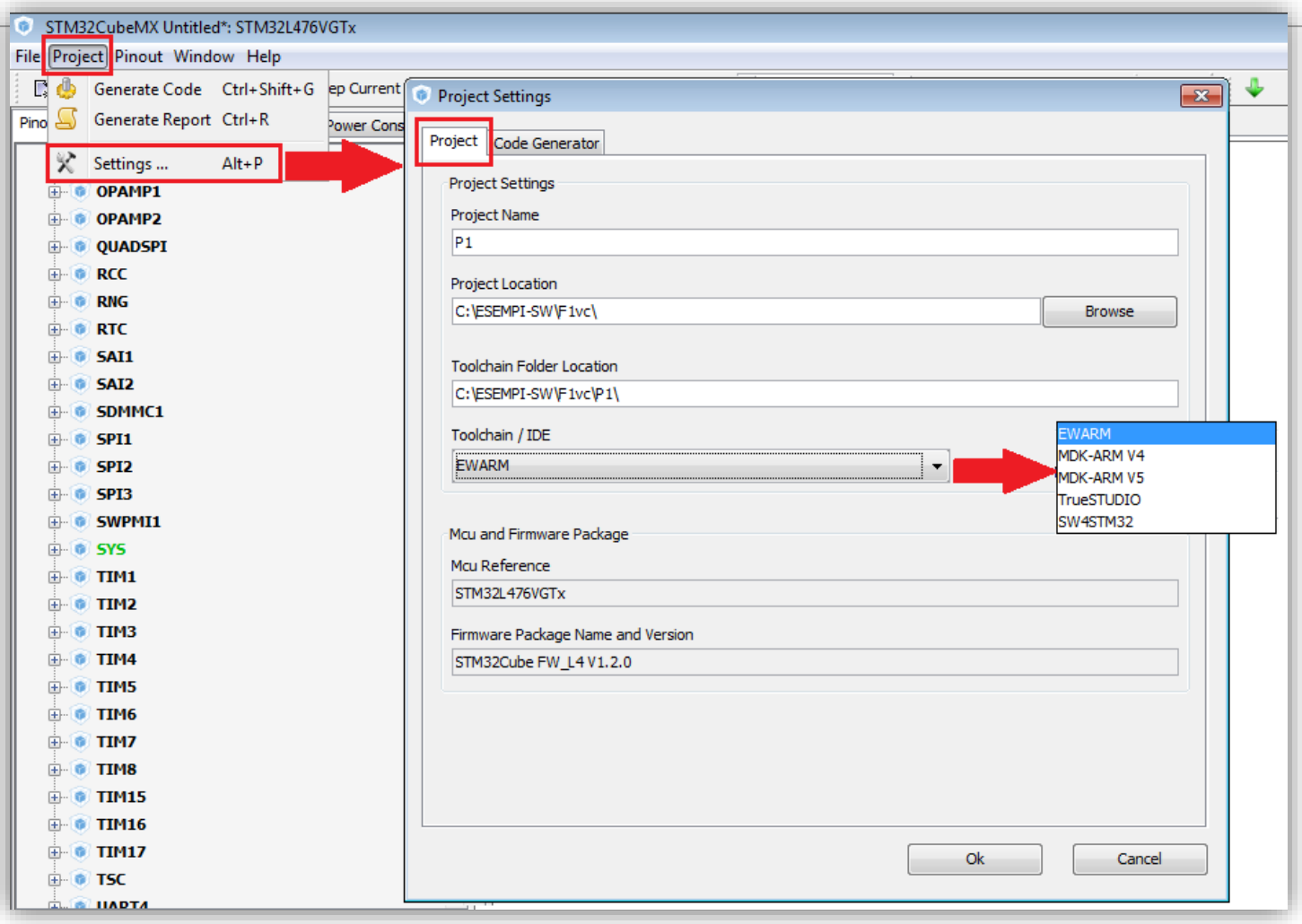


CubeMX: GPIO Panel

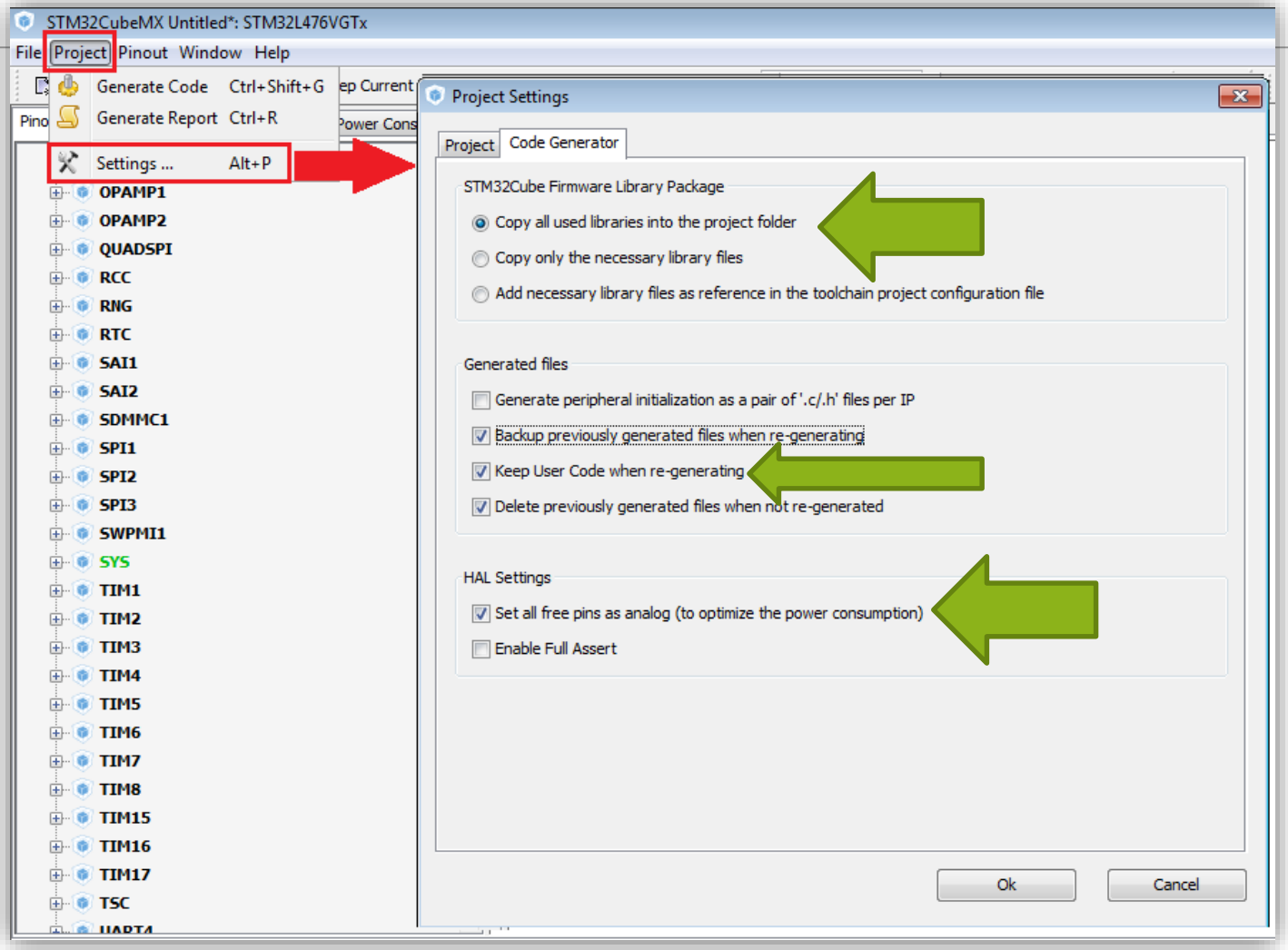
- Most of the GPIO parameters are set by default to the correct value
- You may want to change the maximum output speed
- You can select multiple pin at a time to set the same parameter



CubeMX generate the code for some GUI 1/3

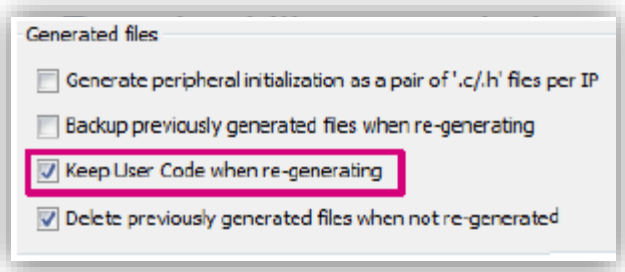


CubeMX generate the code for some GUI 2/3



CubeMX generate the code for some GUI 3/3

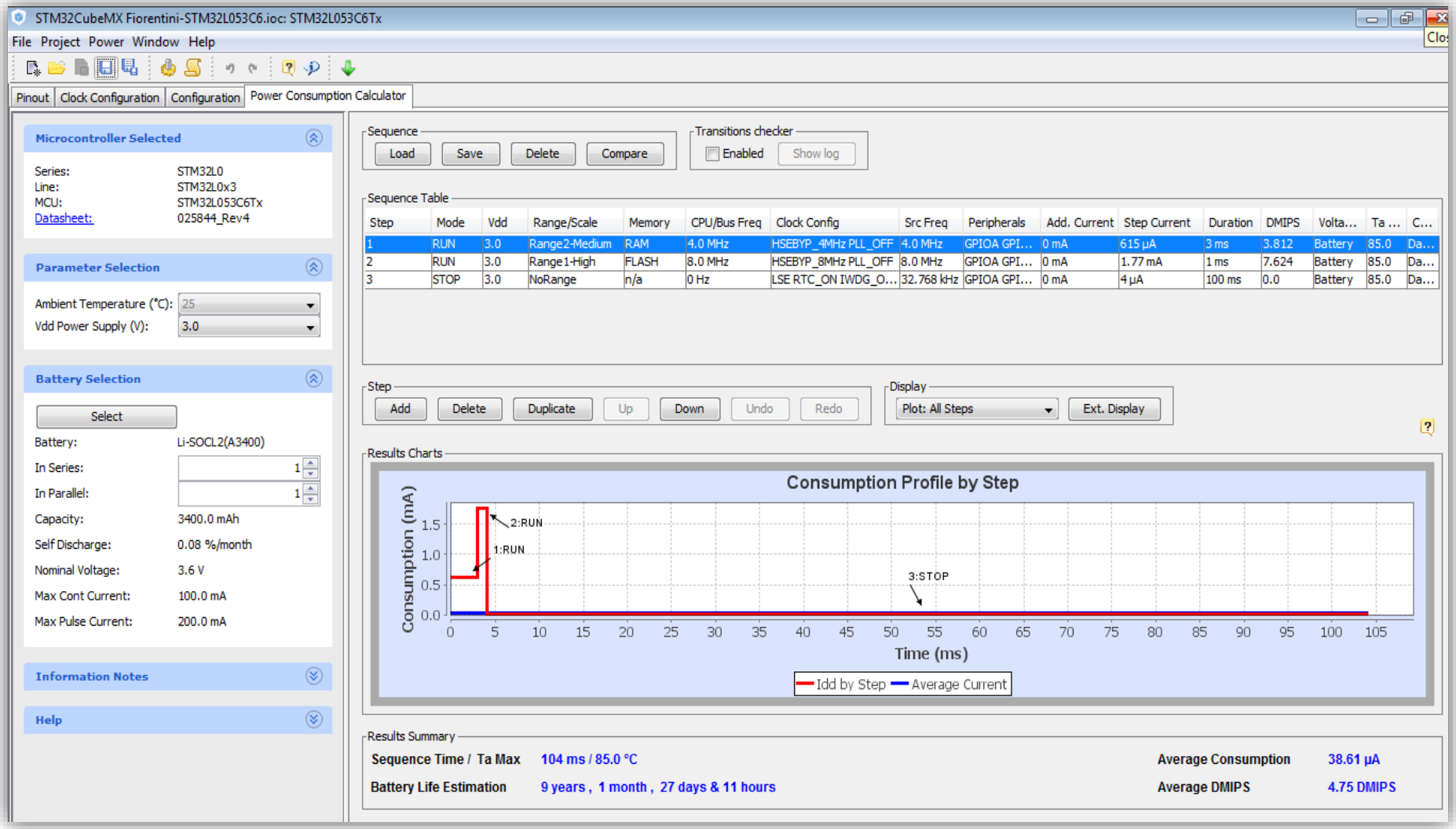
- Generation of all the C initialization code
- Automatic integration with partners toolchains
- User code can be added in dedicated sections and will be kept upon regeneration



main.c

```
22   ******************************************************************************
23   */
24  /* Includes ------------------------------------------------------------------*/
25  #include "stm32f4xx_hal.h"
26  #include "cmsis_os.h"
27  #include "lwip.h"
28  #include "usb_device.h"
29
30  /* Define structures */
31  ADC_HandleTypeDef hadc1;
32
33
34  /* USER CODE BEGIN 0 */
35
36  /* USER CODE END 0 */
37  /* Private function prototypes -----------------------------------------------*/
38  static void SystemClock_Config(void);
39  static void StartThread(void const * argument);
40  static void MX_GPIO_Init(void);
41  static void MX_ADC1_Init(void);
42  static void MX_NVIC_Init(void);
43
44  int main(void)
45  {
46  /* USER CODE BEGIN 1 */
47
48  /* USER CODE END 1 */
49  /* MCU Configuration--------------------------------------------------------*/
50    /* Reset of all peripherals, Initializes the Flash interfa
51    HAL_Init();
52    /* Configure the system clock */
```

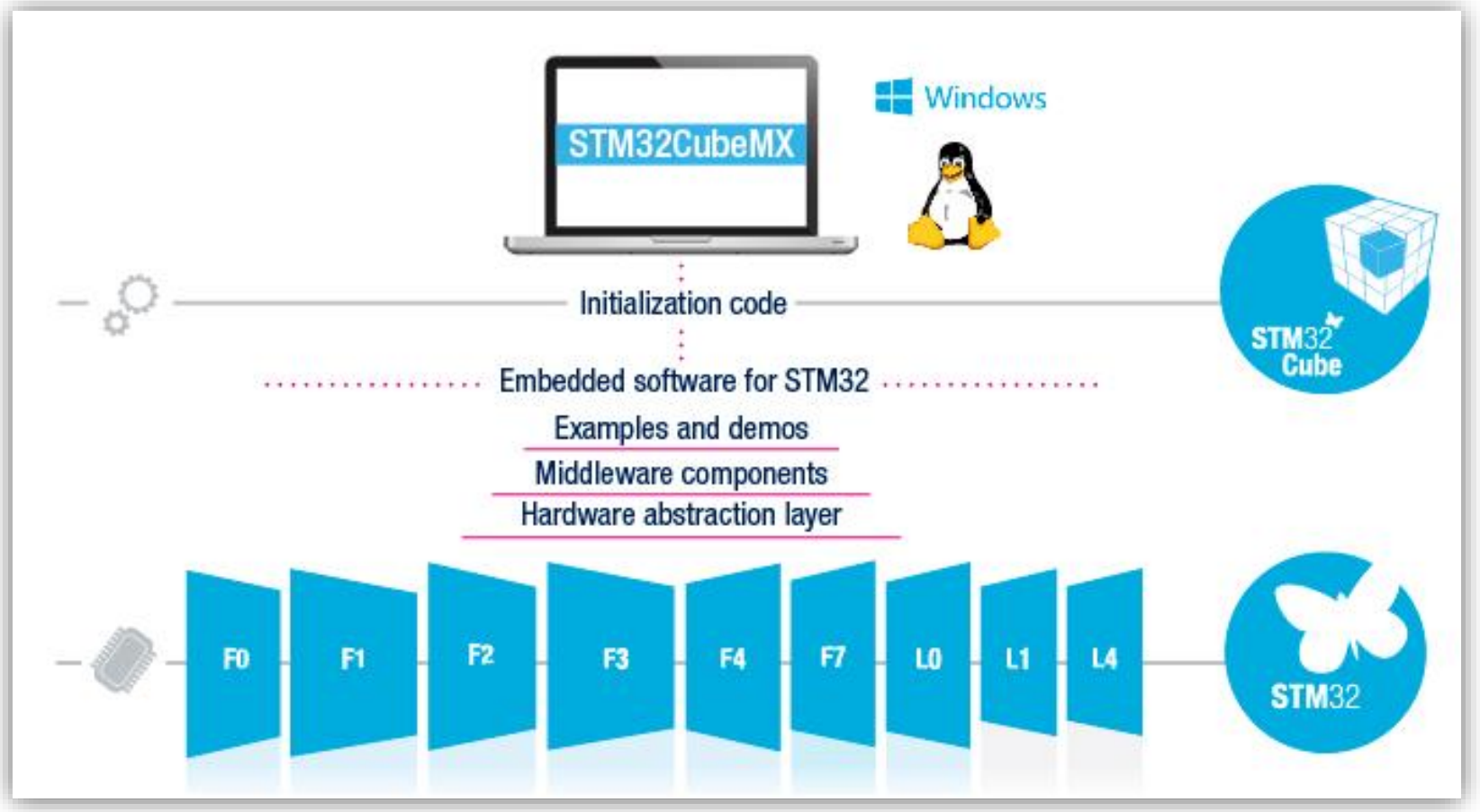
CubeMX: Power consumption calculator



HAL library

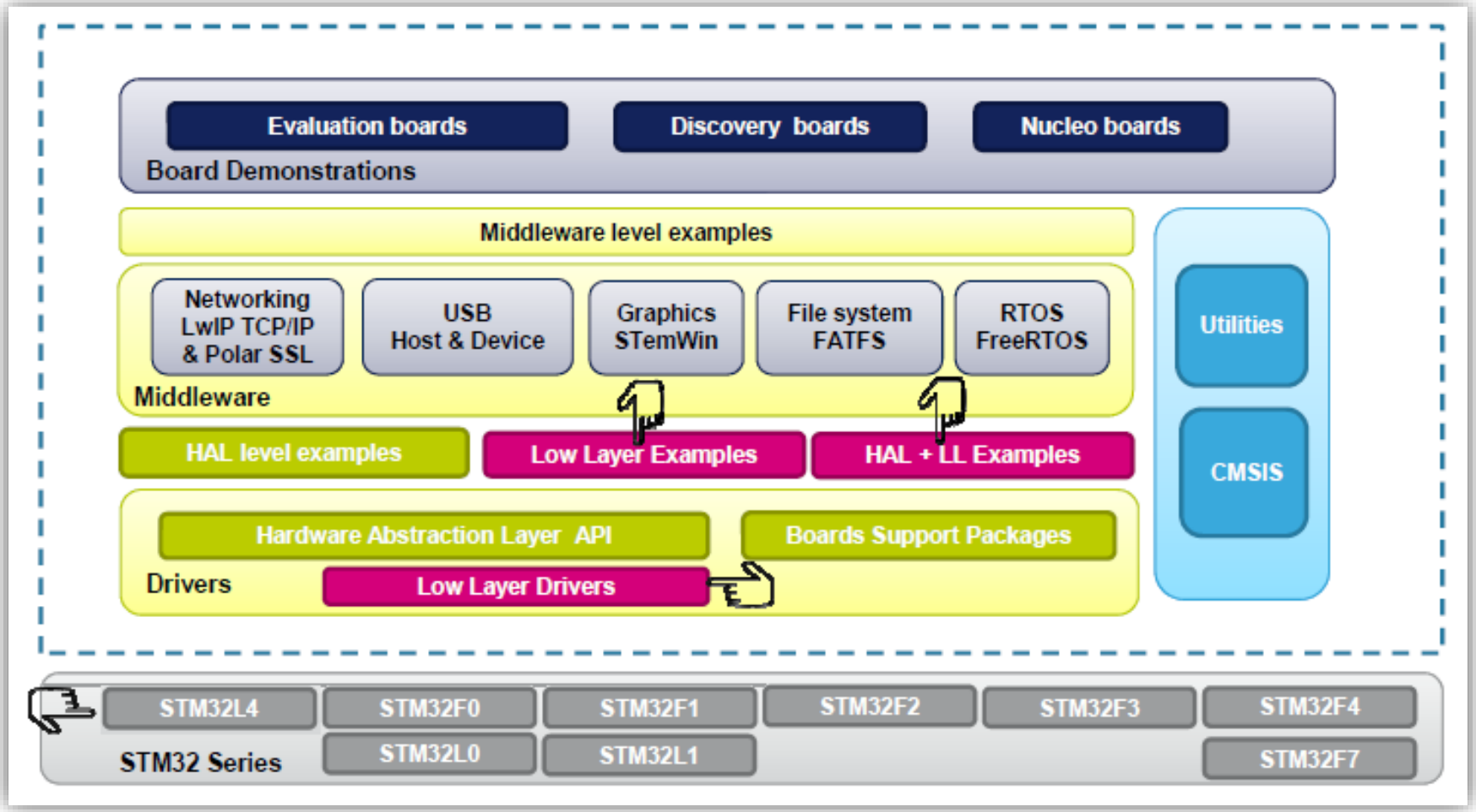
HAL library

HAL library – HAL == hardware abstraction layer



HAL library

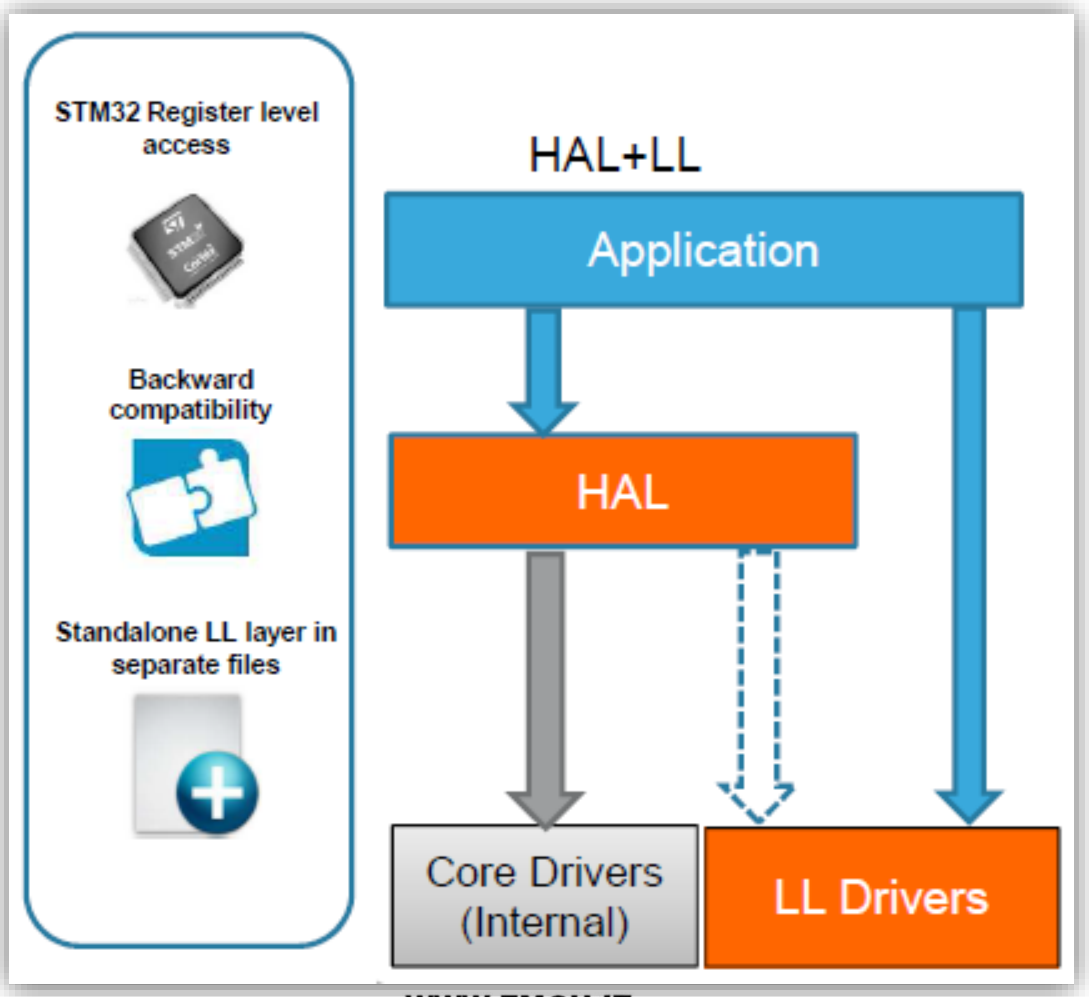
The HAL library are **here**.



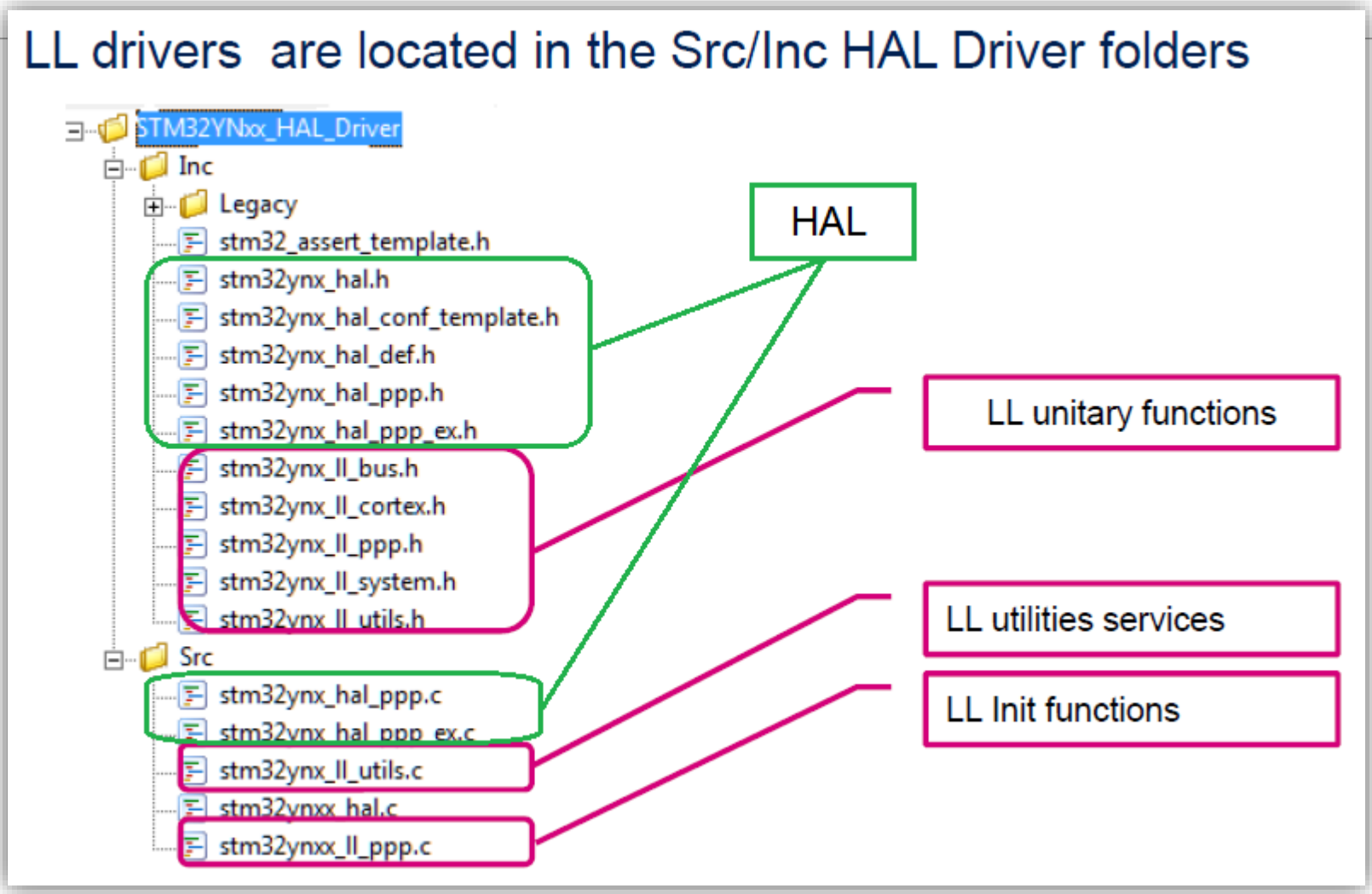
HAL library

- **STM32Cube HAL & LL are complementary** and covers a wide range of applications requirements:
  - **HAL** offers **high level** and **functionalities oriented APIs**, with **high portability level** and **hide product/IPs complexity to end user**
  - **LL** offers **low level APIs at registers level,** w/ **better optimization** but **less portability** and require **deep knowledge of the product/IPs specification**
- The **new Low Layer (LL)** is offering the following services:
  - **Unitary static inline functions for direct register access** (provided in *.h files)
    - One-shot operations that can be used by the HAL drivers or from application level.
    - Independant from HAL and can be used in standalone usage (without HAL drivers)
    - Full features coverage of the supported IP
  - **Init functions** (provided in *.c files)
    - compatible with Standard peripheral library

HAL library



HAL library



HAL library

Covered peripherals (1/2)

Peripherals (IPs)		STM32Cube Support	
		HAL	LL
System	Flash	Yes	No (some of the Flash features need to be handled in the MISC file to prevent dependency with HAL when using LL PWR driver)
	EXTI	Yes	Yes
	GPIO	Yes	Yes
	DMAs	Yes	Yes
	PWR	Yes	Yes
	RCC	Yes	Yes
	Cortex	Yes	No (some of the cortex features added: MPU, SYSTICK, CPUID, SLEELDEEP)
	SYSCFG	Yes	Yes
Analog	ADC	Yes	Yes
	SDADC	Yes	Yes
	DAC	Yes	Yes
	COMP	Yes	Yes
	DFSDM	Yes	No
	OPAMP	Yes	Yes
Timers	RTC	Yes	Yes
	TIM	Yes	Yes
	LPTIM	Yes	Yes
	HRTIM	Yes	Yes
	WWDG	Yes	Yes
	IWDG	Yes	Yes
Cryptography	CRC	Yes	Yes
	CRYP	Yes	No
	HASH	Yes	No
	RNG	Yes	Yes

HAL library

Covered peripherals (2/2)

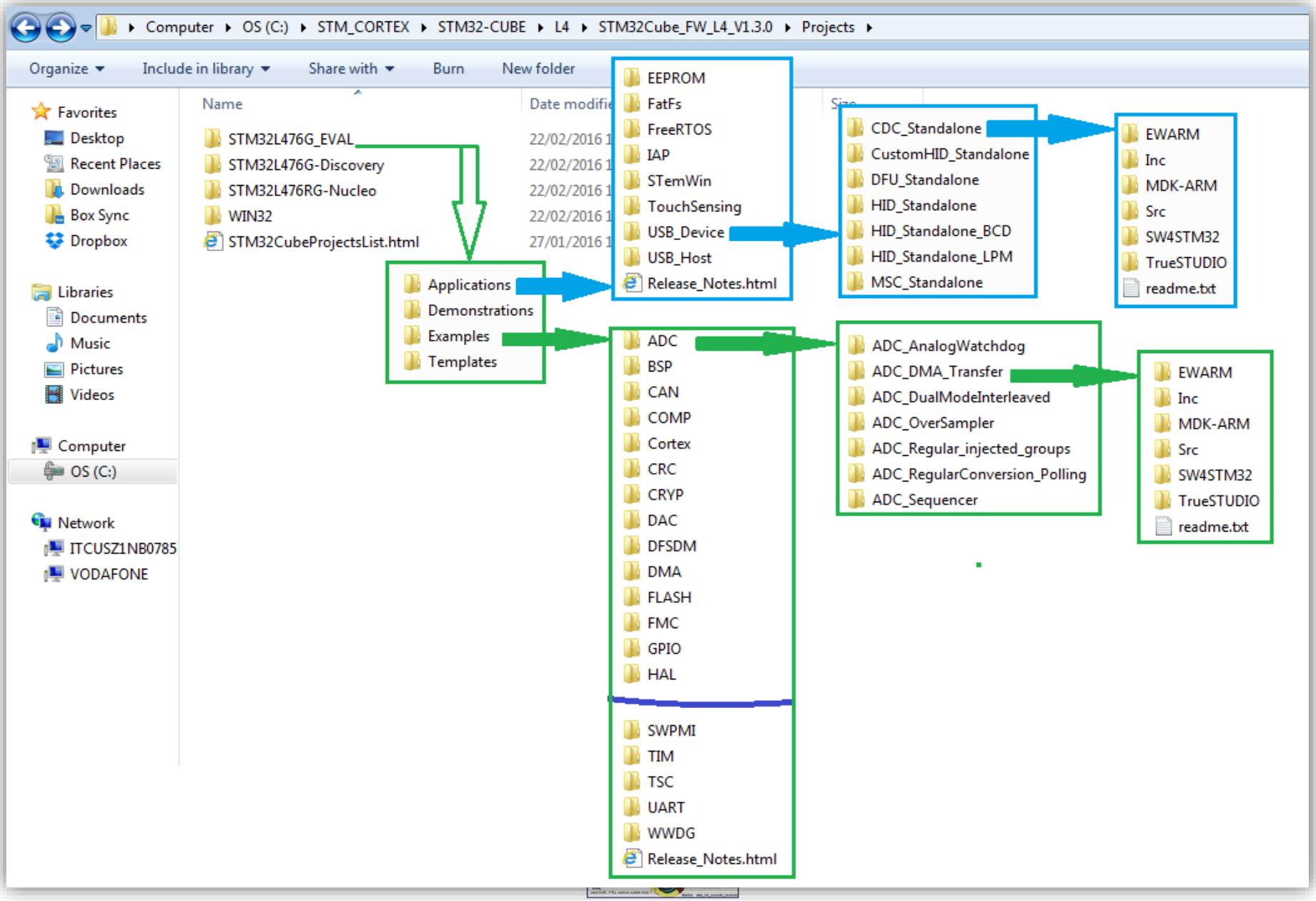
Peripherals (IPs)		STM32Cube Support	
		HAL	LL
Basic Connectivity	I2C/SMBUS	Yes	Yes
	UART/USART/LPUART	Yes	Yes
	SWPMI	Yes	Yes
	SPI/I2S	Yes	Yes
	SDMMC(SDIO)	Yes	No
	CAN	Yes	No
	CEC	Yes	No
Advanced Connectivity	USB-FS-Device	Yes	No
	USB-OTG-FS/HS	Yes	No
	Ethernet	Yes	No
	MDIOS	Yes	No
Interface	FSMC(FMC)	Yes	No
	LCD"Glass"	Yes	No
	LTDC	Yes	No
	DSI	Yes	No
	DMA2D	Yes	Yes
	JPEG	Yes	No
	DCMI	Yes	No
	QSPI	Yes	No
	SPDIF-IN	Yes	No
	SAI	Yes	No

HAL library

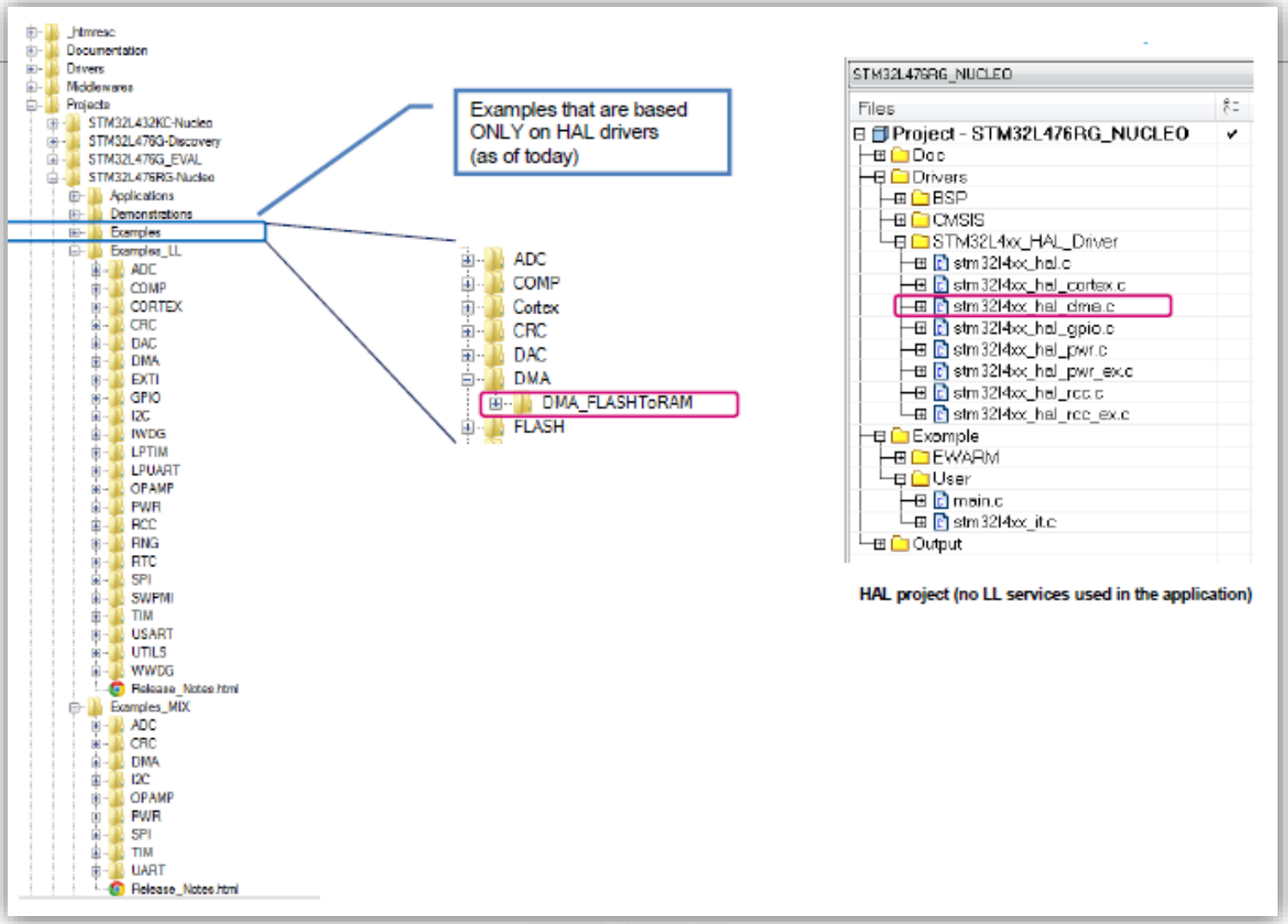
HAL vs. LL usage

- To cohabitate the HAL with the LL, user has to be aware about some HAL concepts.
- Main constraint is when the LL overwrites some registers that the content is mirorred in the HAL handles.
- The Low Layer drivers cannot be automatically used with the HAL for the same peripheral instance: mainly can't run concurrent process on the same IP using both APIs, however sequential use is allowed.
- The low layer drivers can be used without any constraint with all the HAL drivers that are not based on handle objects (RCC, Cortex, common HAL, flash and GPIO)
- The LL is intented to be used in expert mode (high knowledge on STM32 hardware aspect)

HAL library - Where to find examples ready to use ?

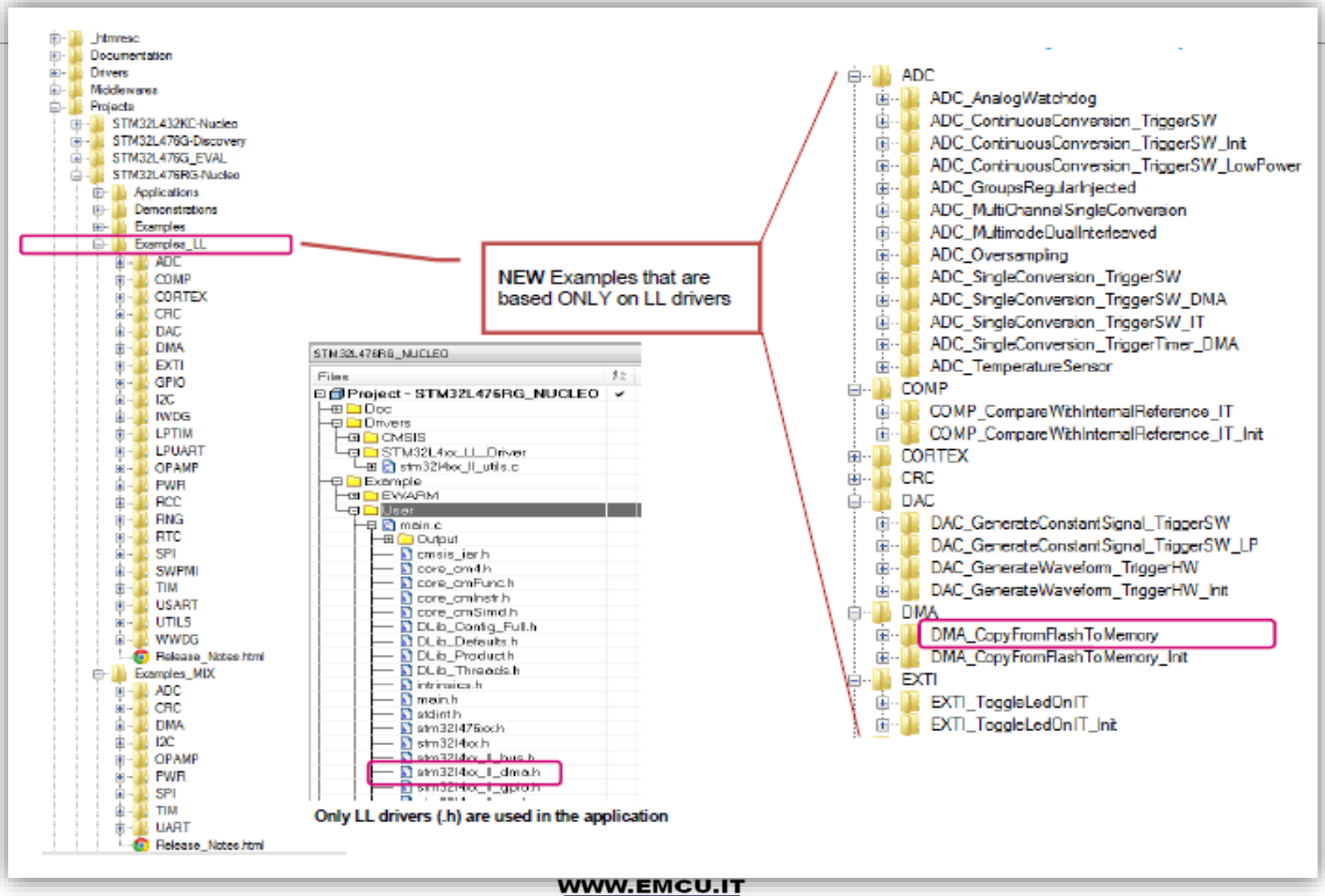


HAL library – HAL examples

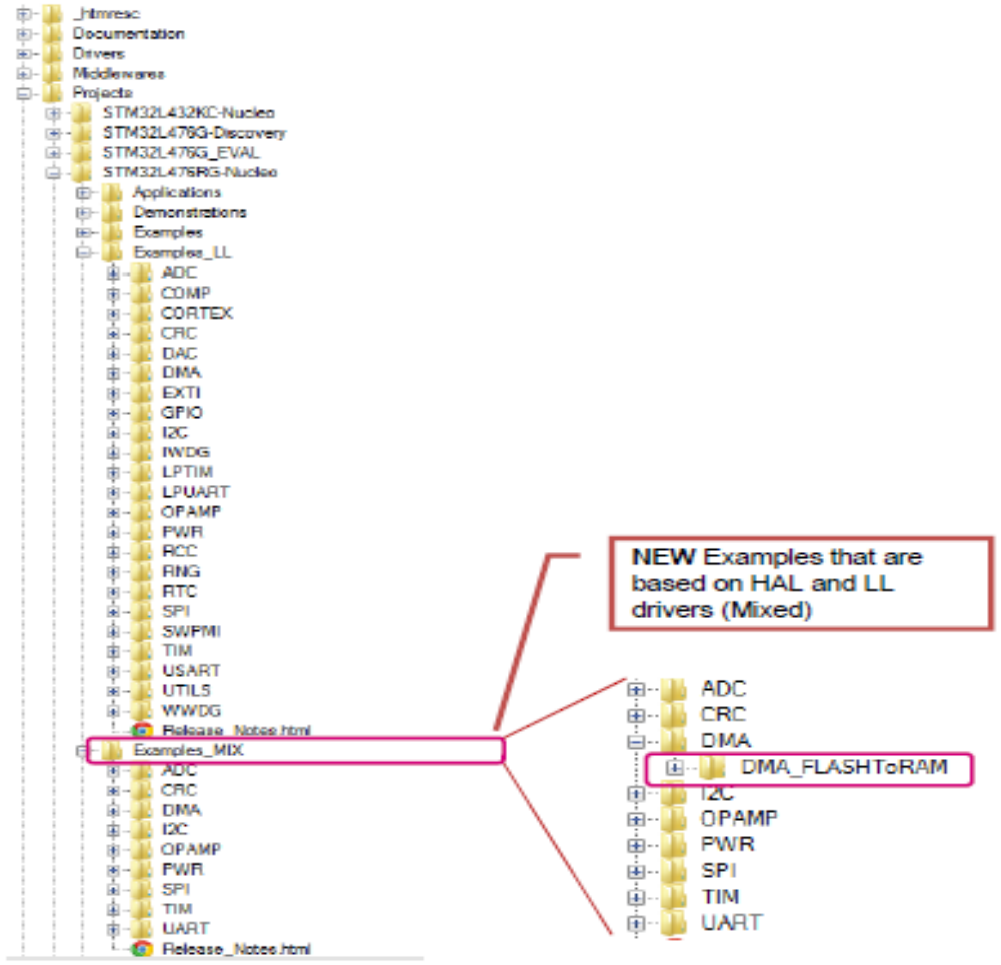


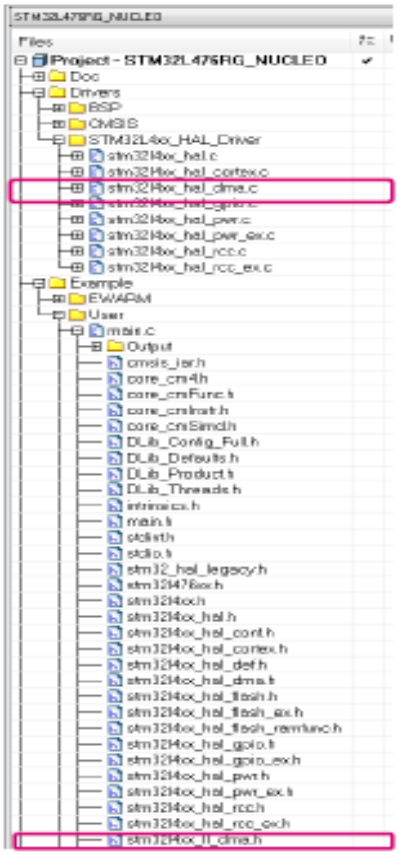
HAL project (no LL services used in the application)

HAL library – LL examples



HAL library - LL & HAL mix Example





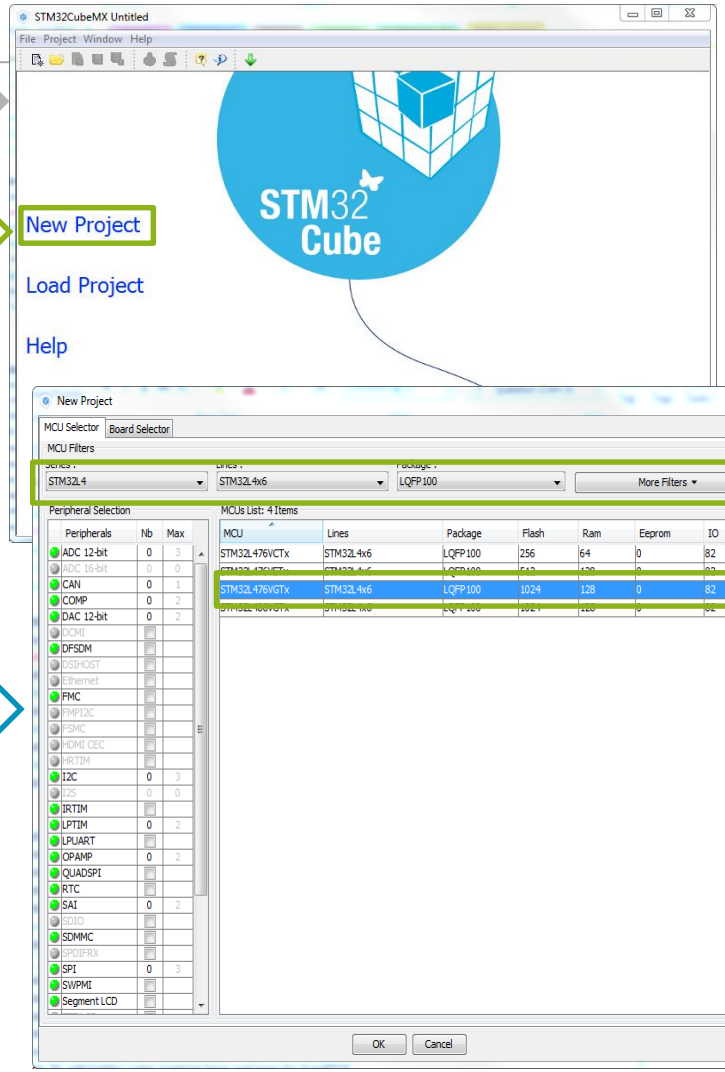
Mixed HAL and LL drivers used in the application

Start new project

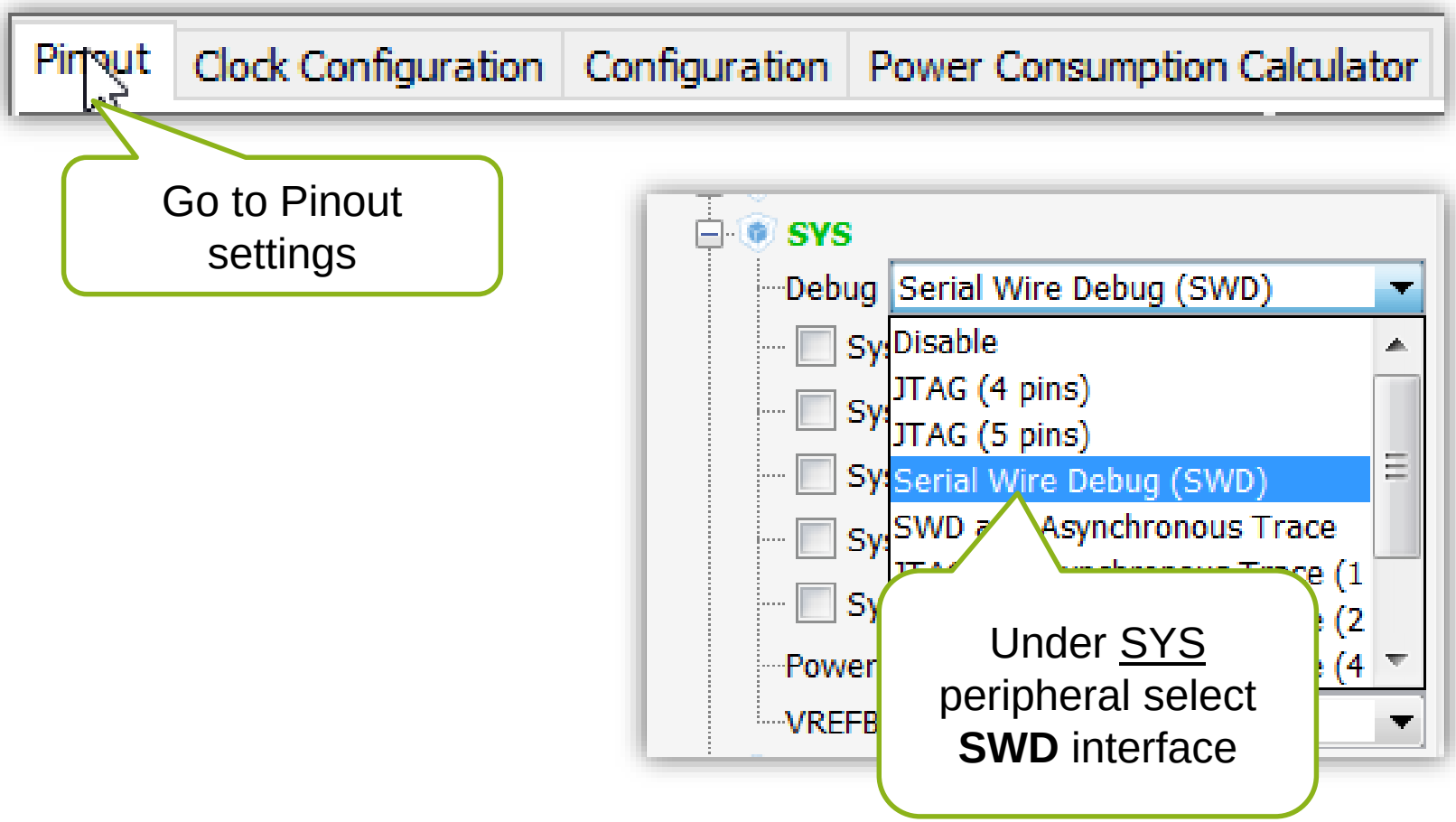
New Project

Create new project using CubeMX

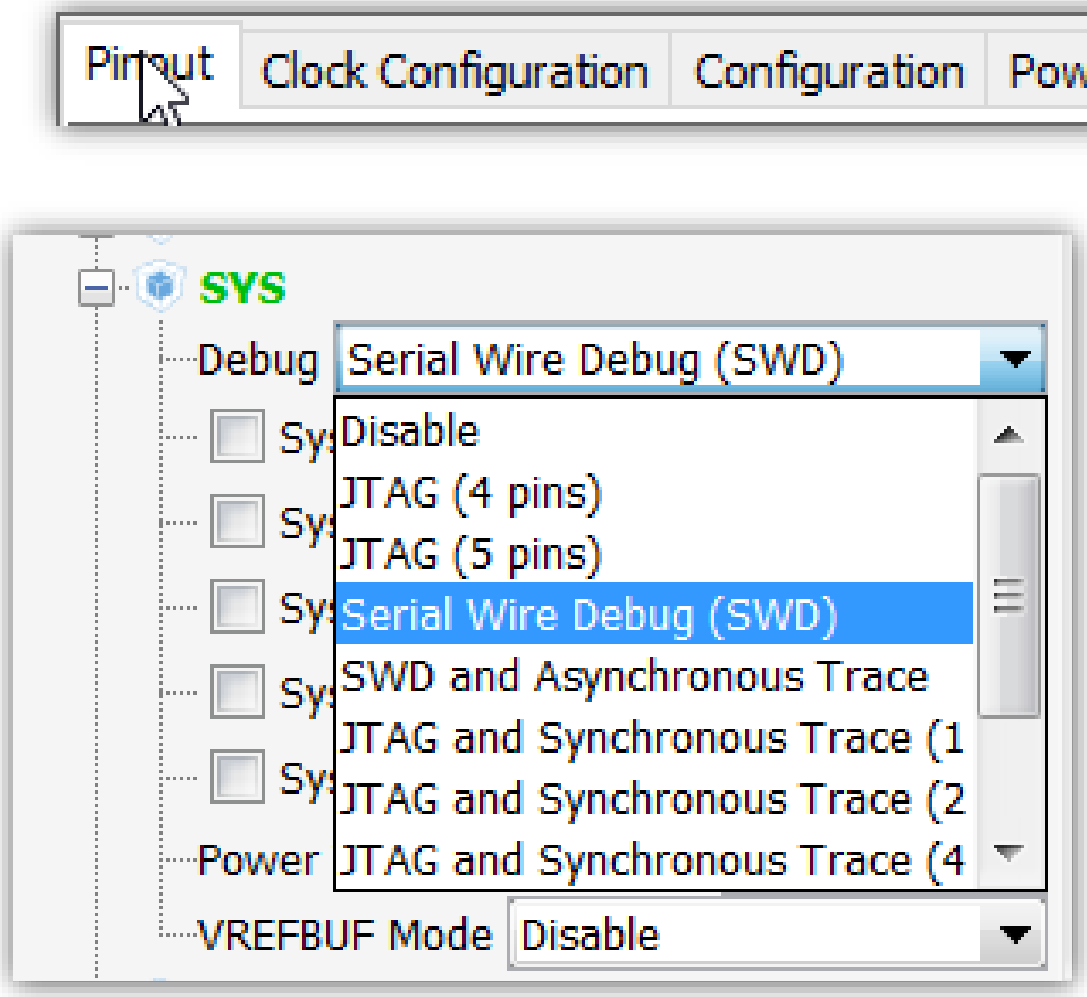
- Run CubeMX tool
- Start **new project**
  - Click “New Project” desktop shortcut, or
  - Go to “Menu->File->New Project”
- Filter:
  - Series: STM32L4
  - Line: STM32L4x6
  - Package: LQFP100
- Select: **STM32L476VGTx**

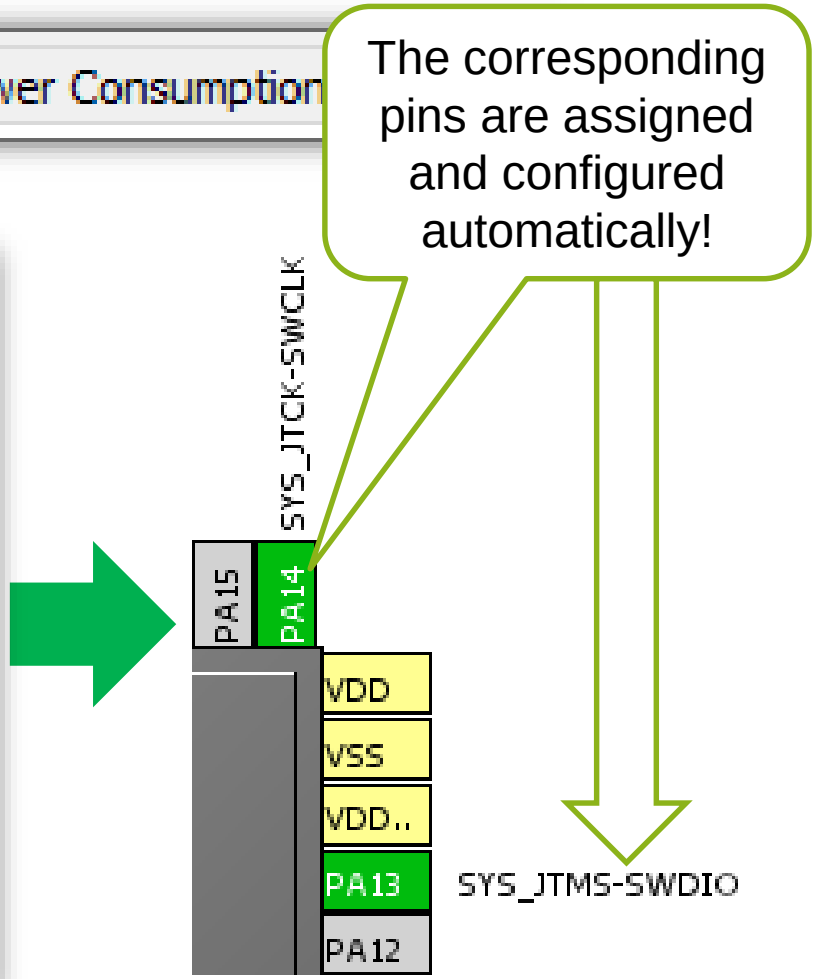


Configure debug interface 1/2

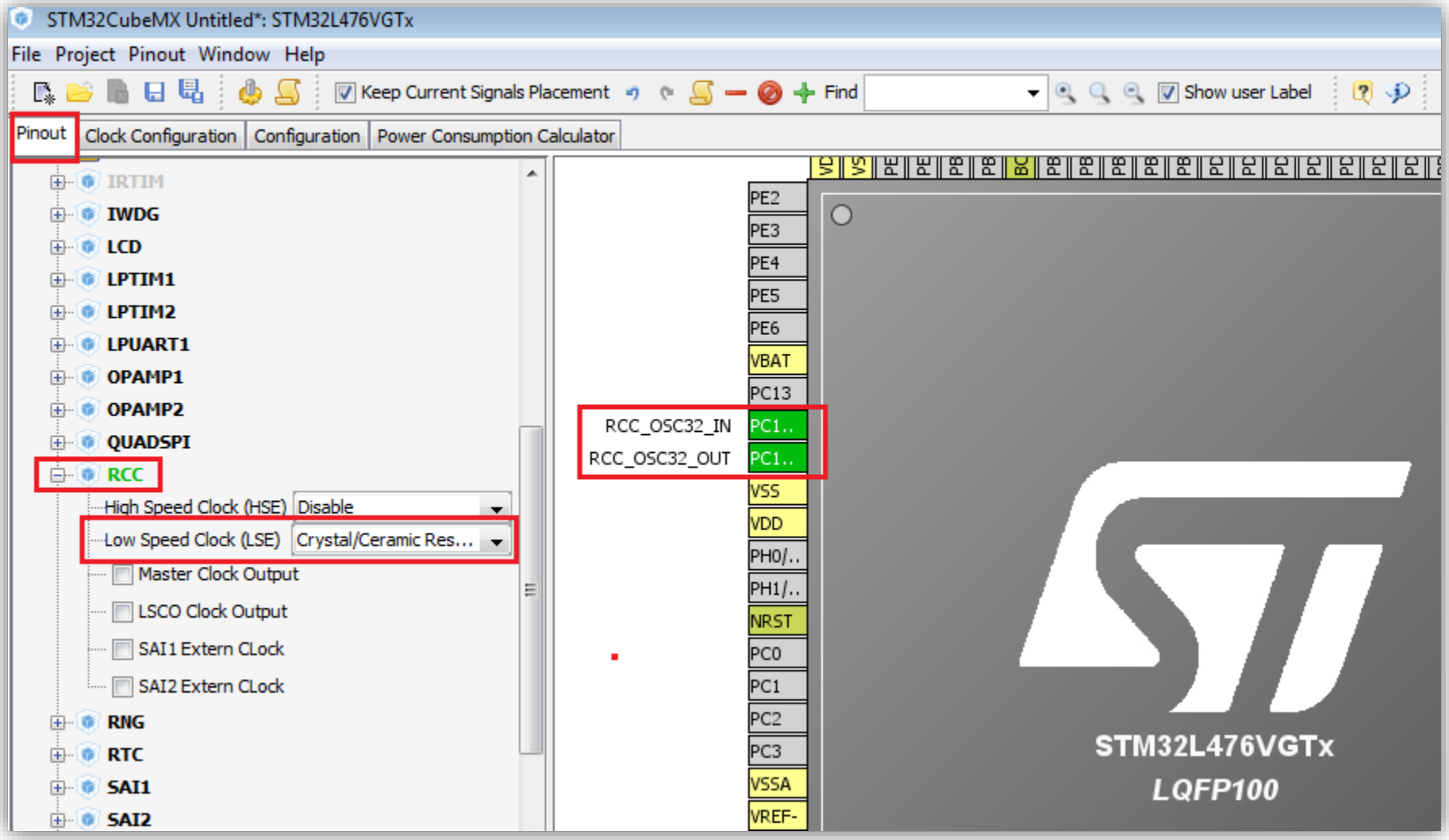


Configure debug interface 2/2



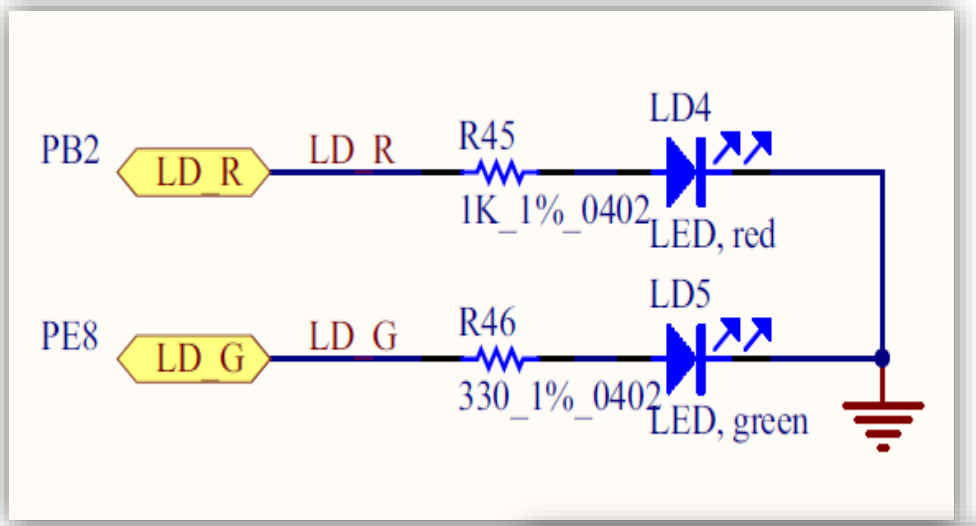


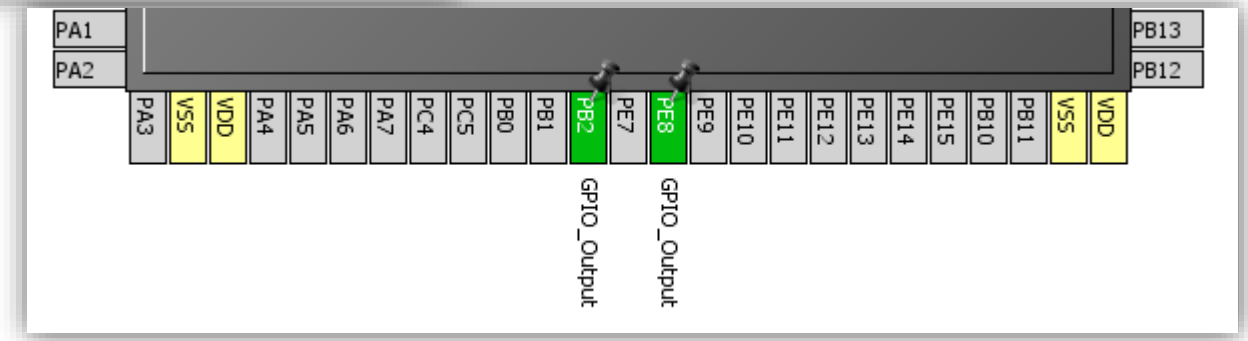
Configure LSE resonator (32,768 KHz)



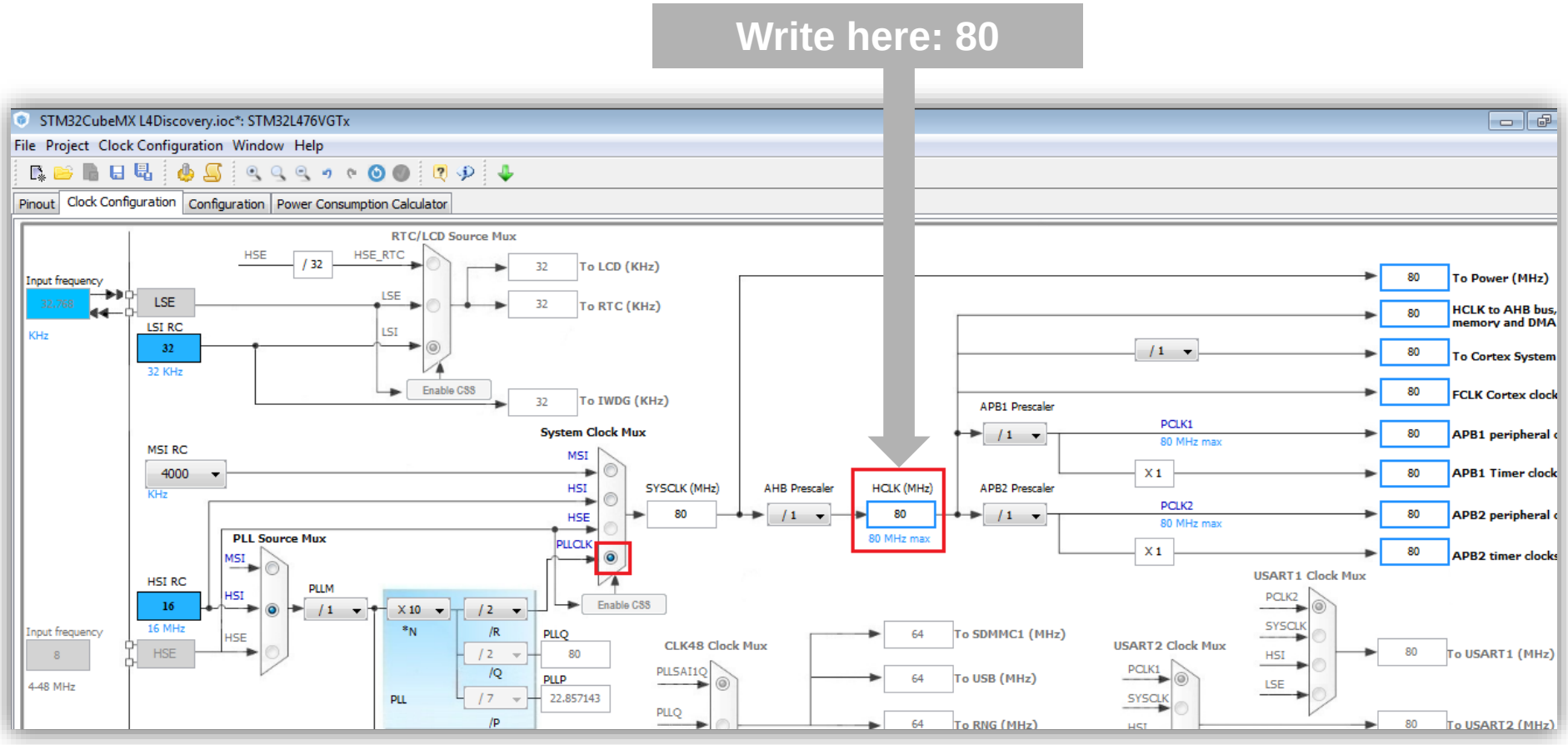
Configure GPIO for LED toggling

Configure LED pin as GPIO_Output

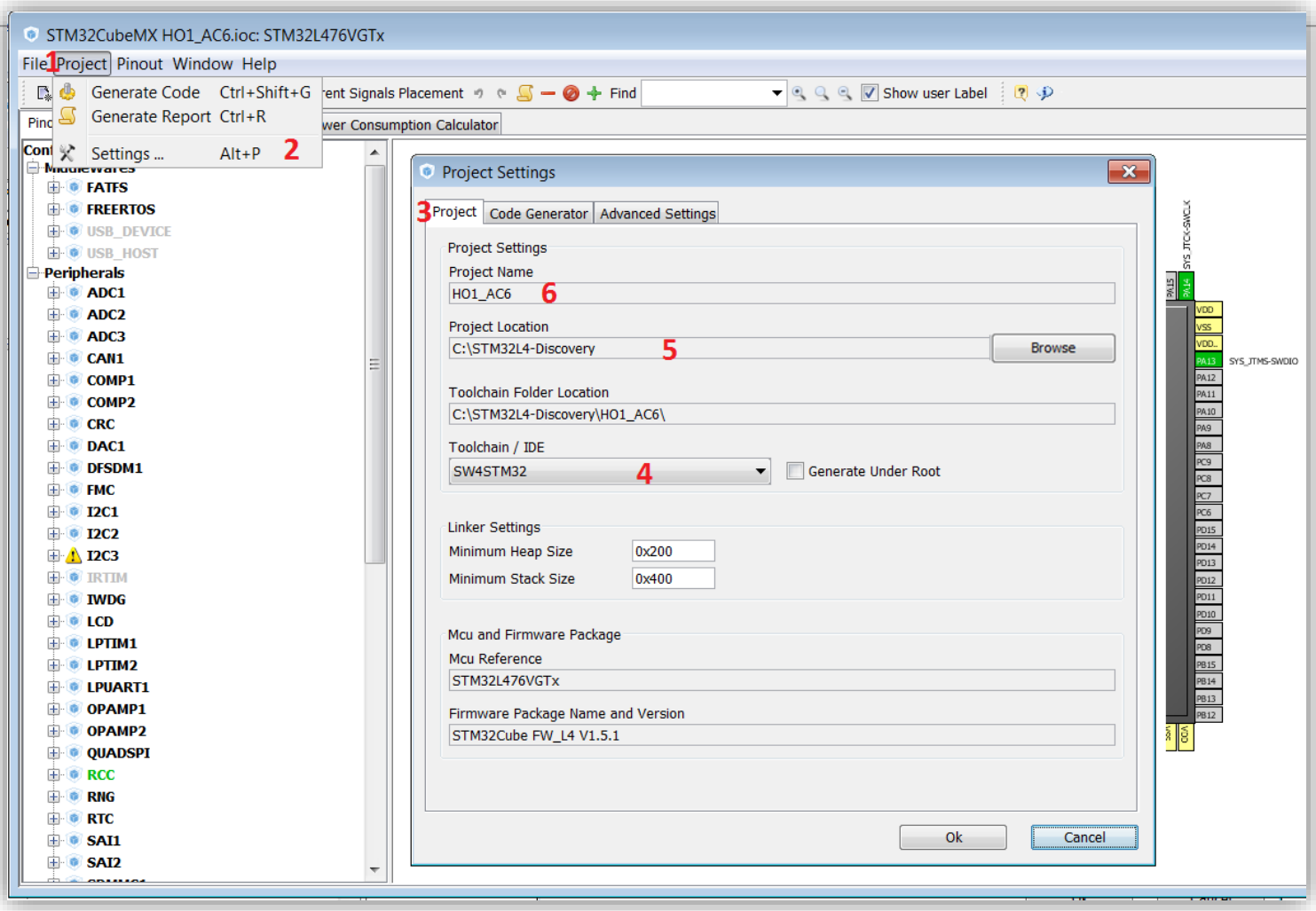




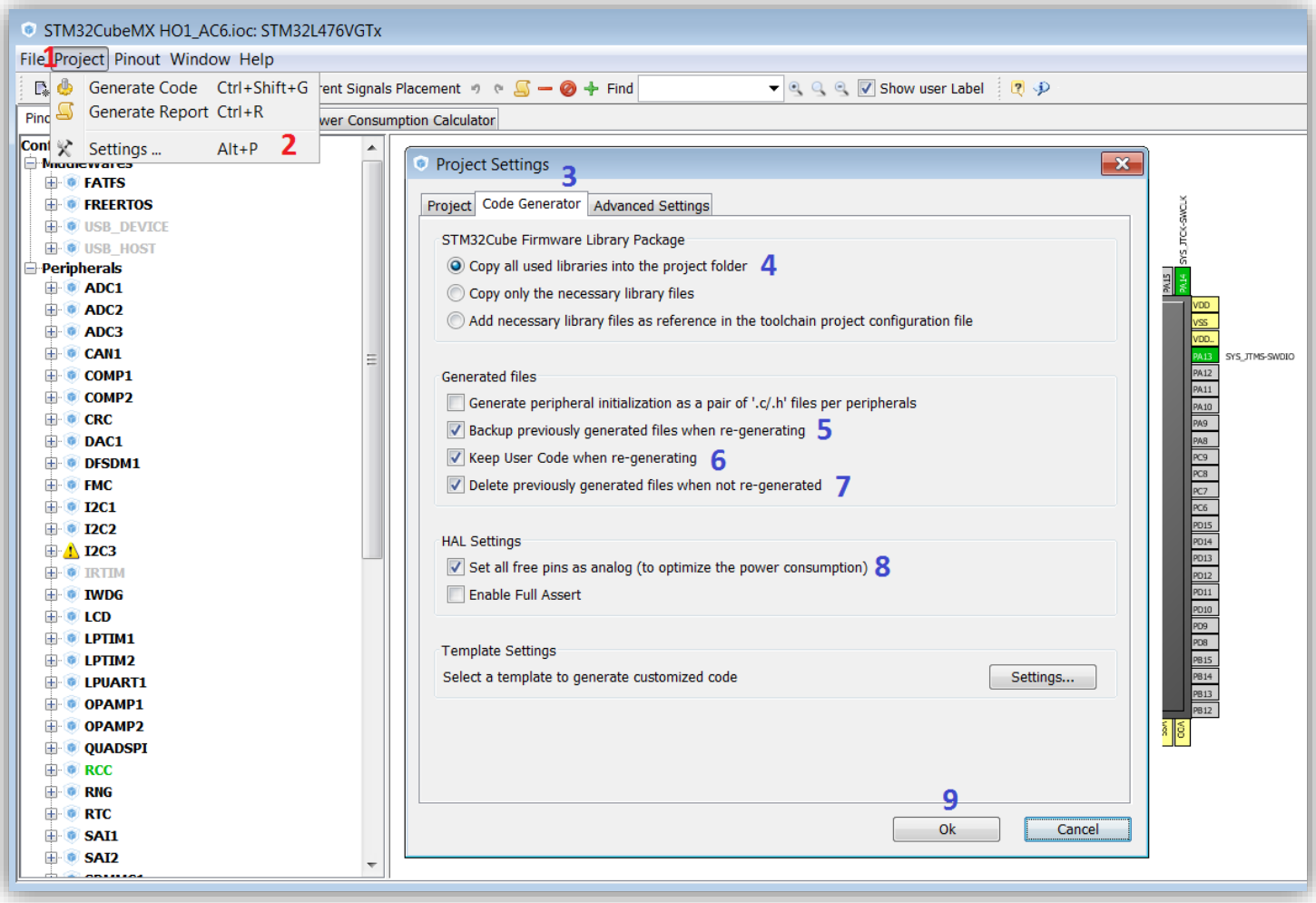
Clock configuration



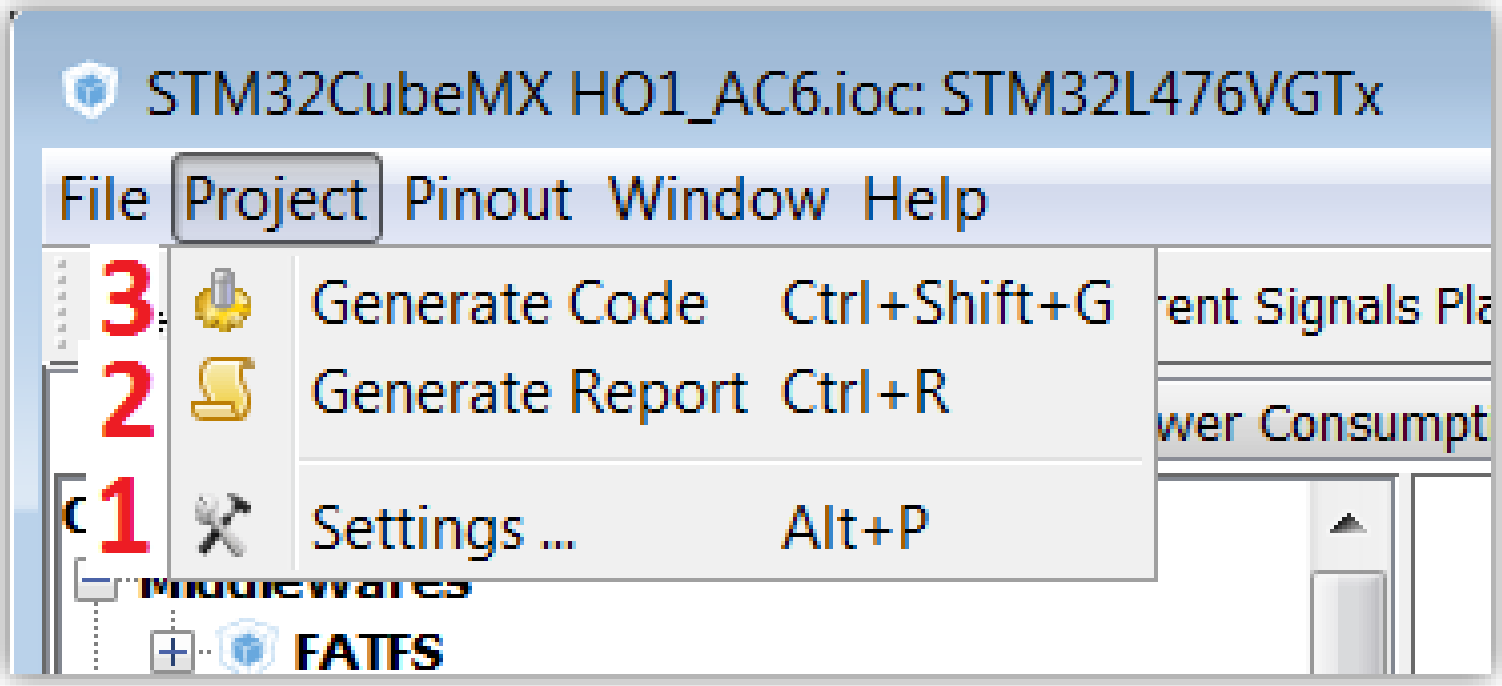
CubeMX generate the code for some GUI 1/3



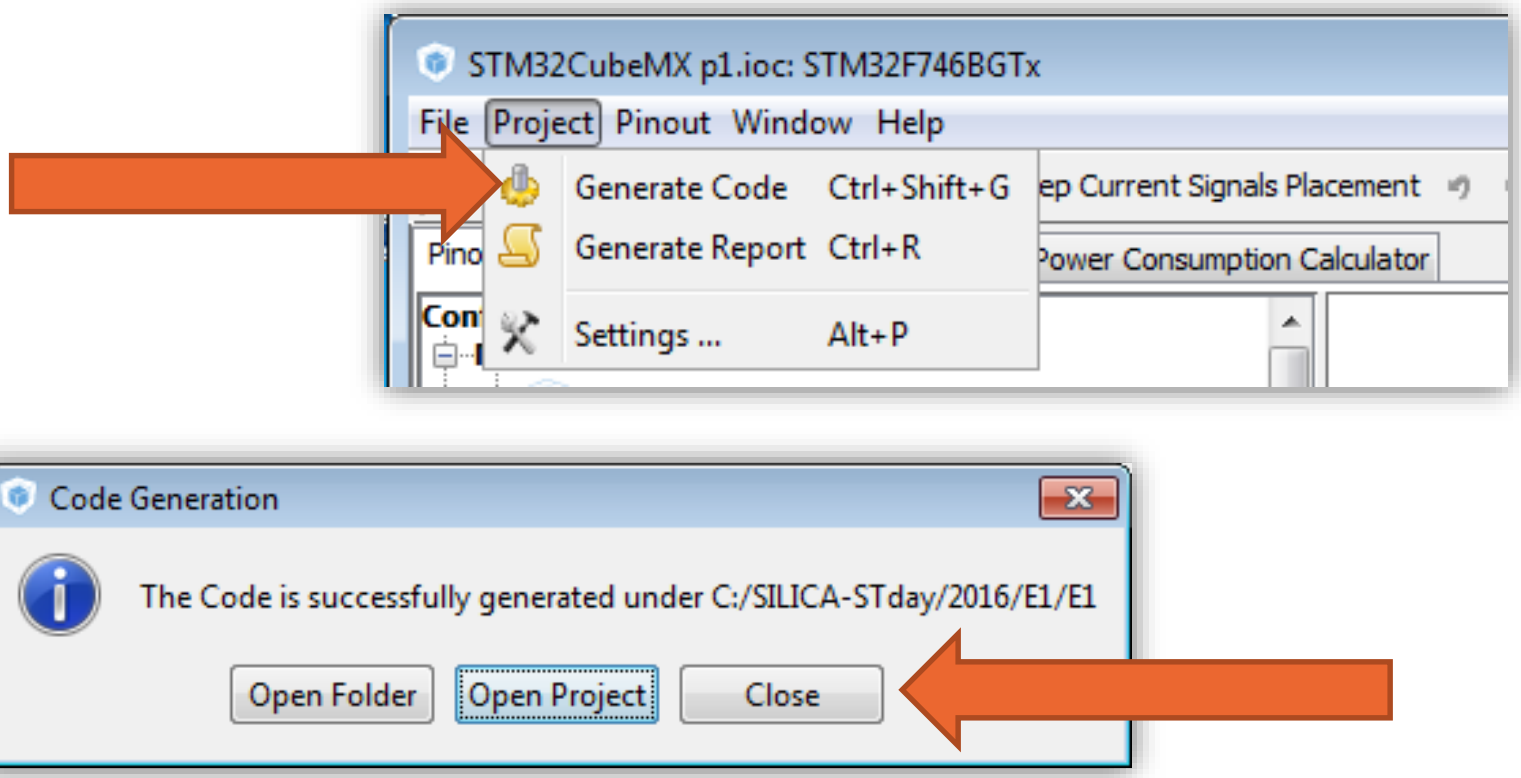
CubeMX generate the code for some GUI 2/3



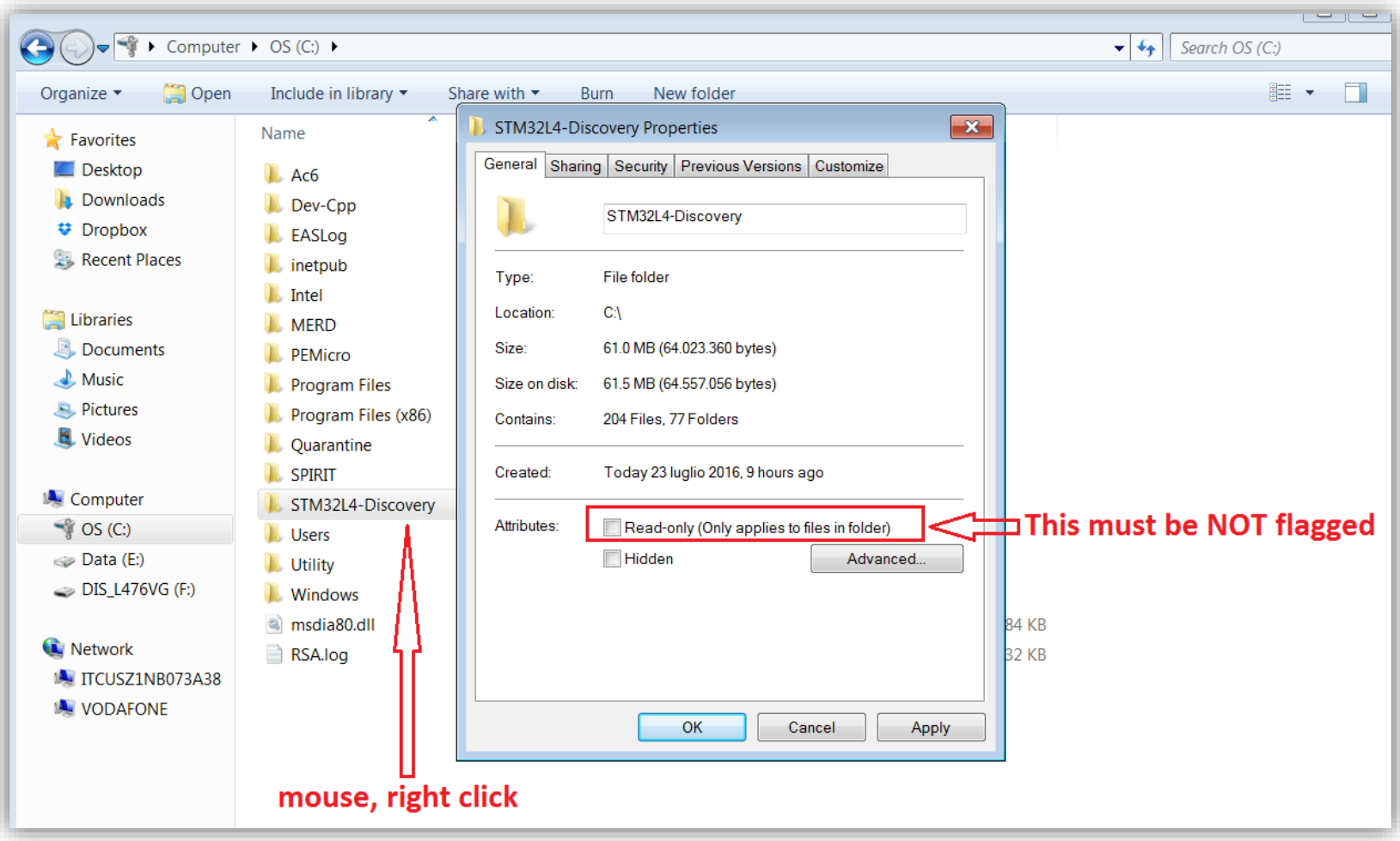
CubeMX generate the code for some GUI 3/3



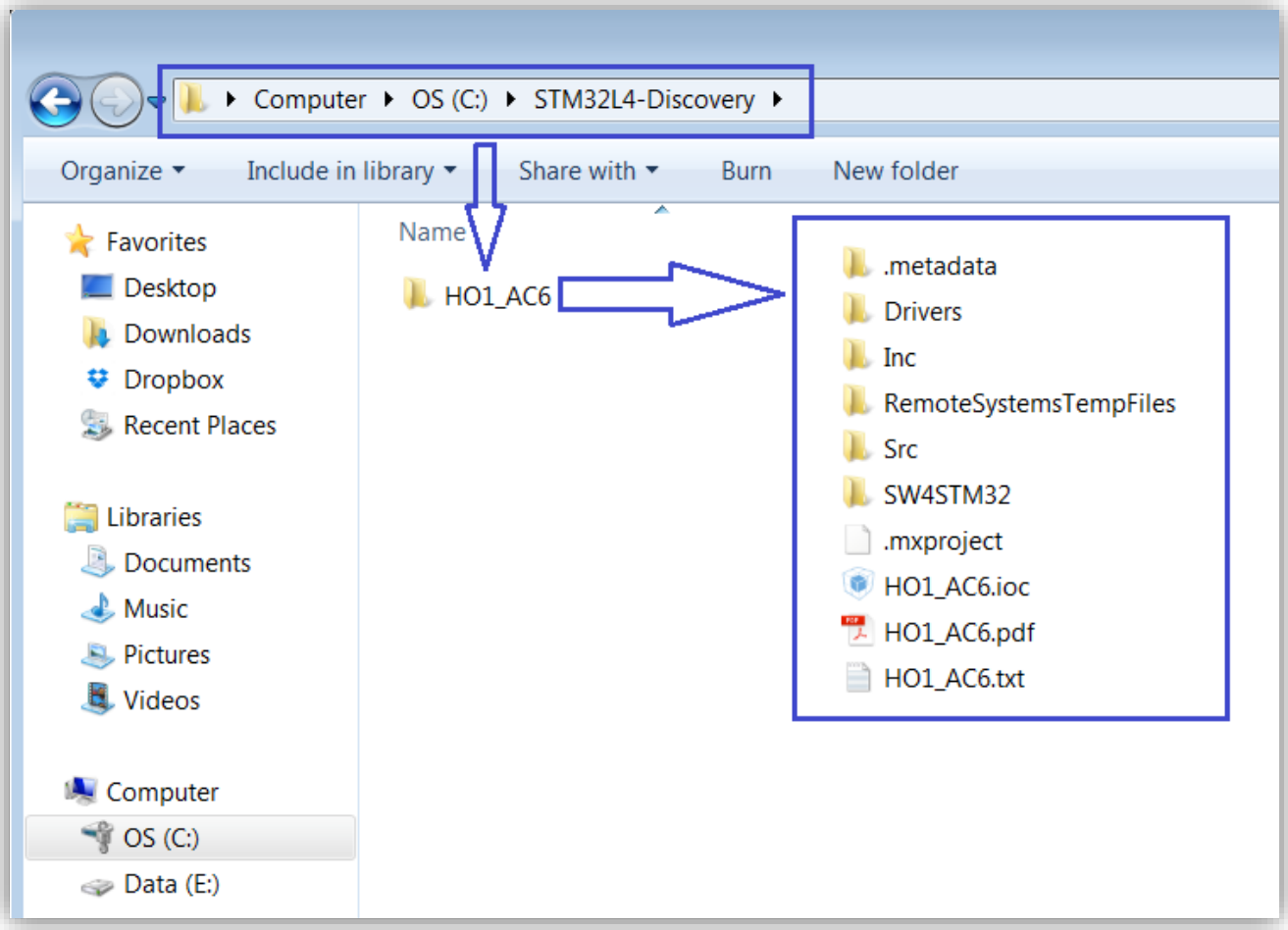
CubeMX generate the code



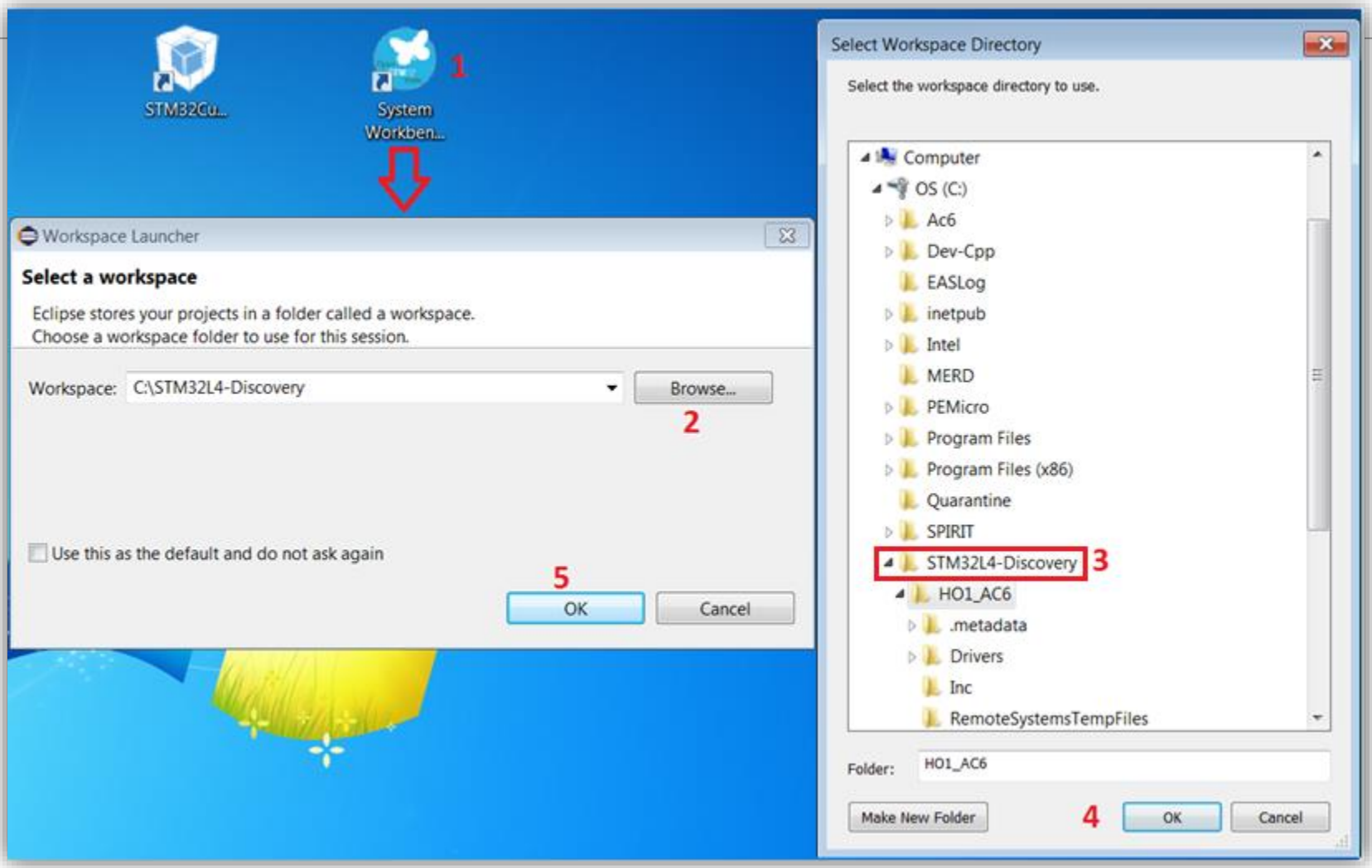
Control the proprietary of your working directory



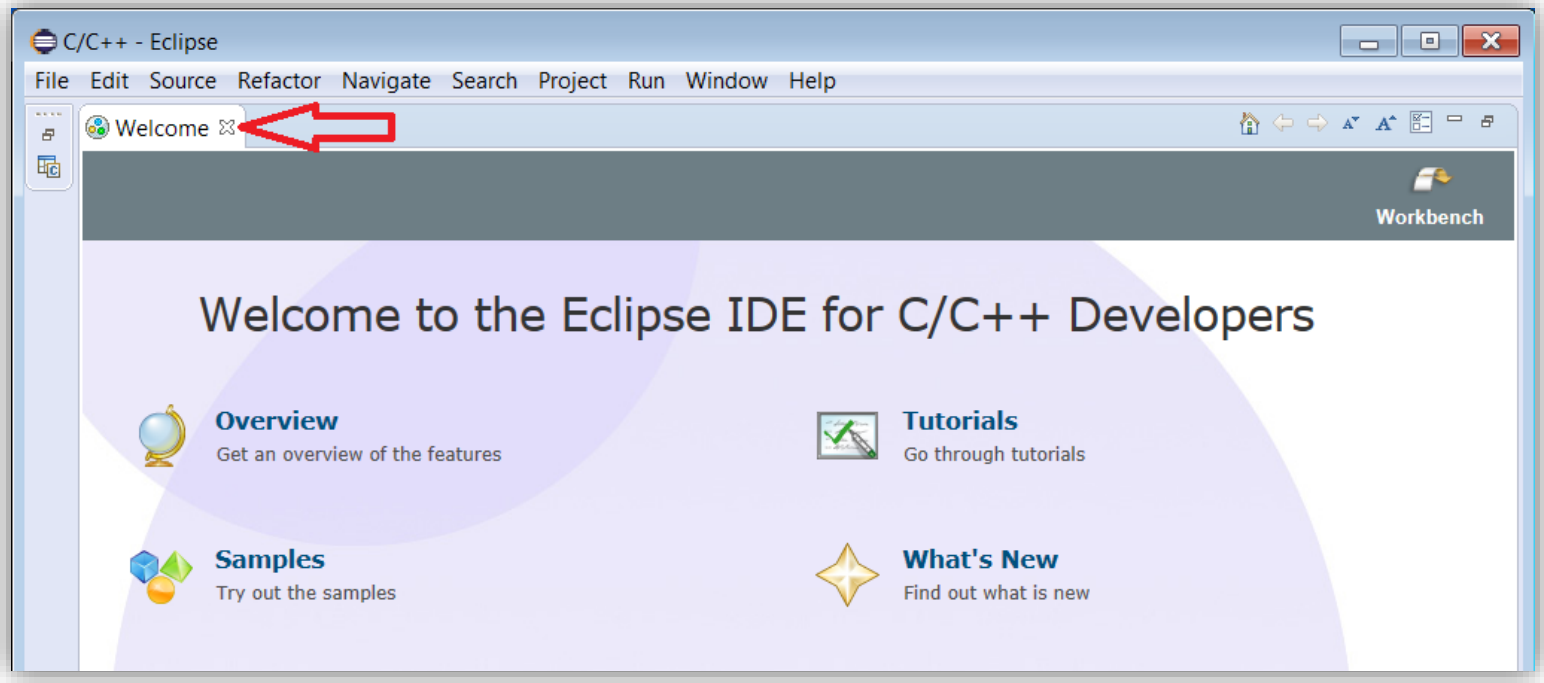
Contents of your working directory



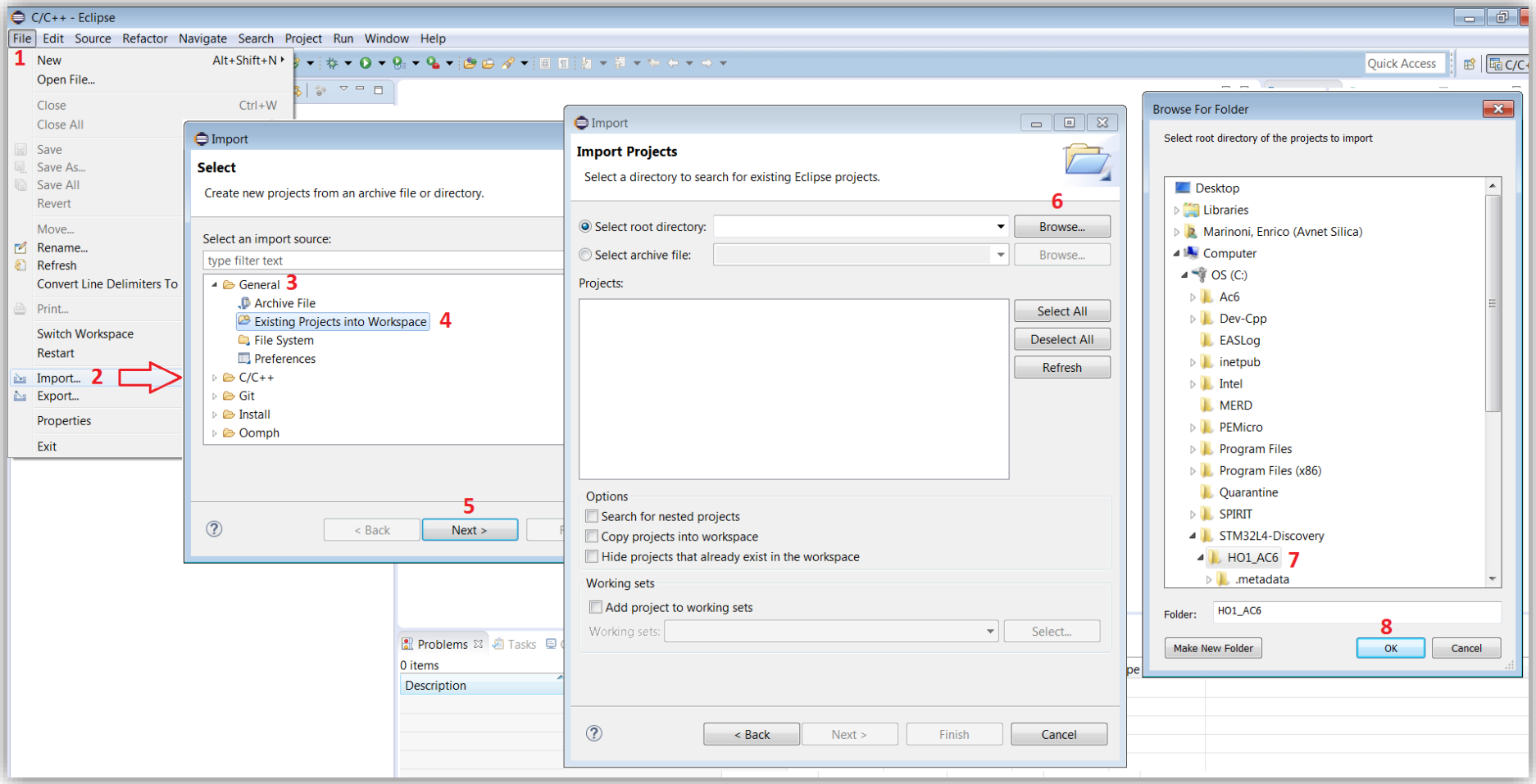
Run AC6 - (SW4STM32 - System Workbench)



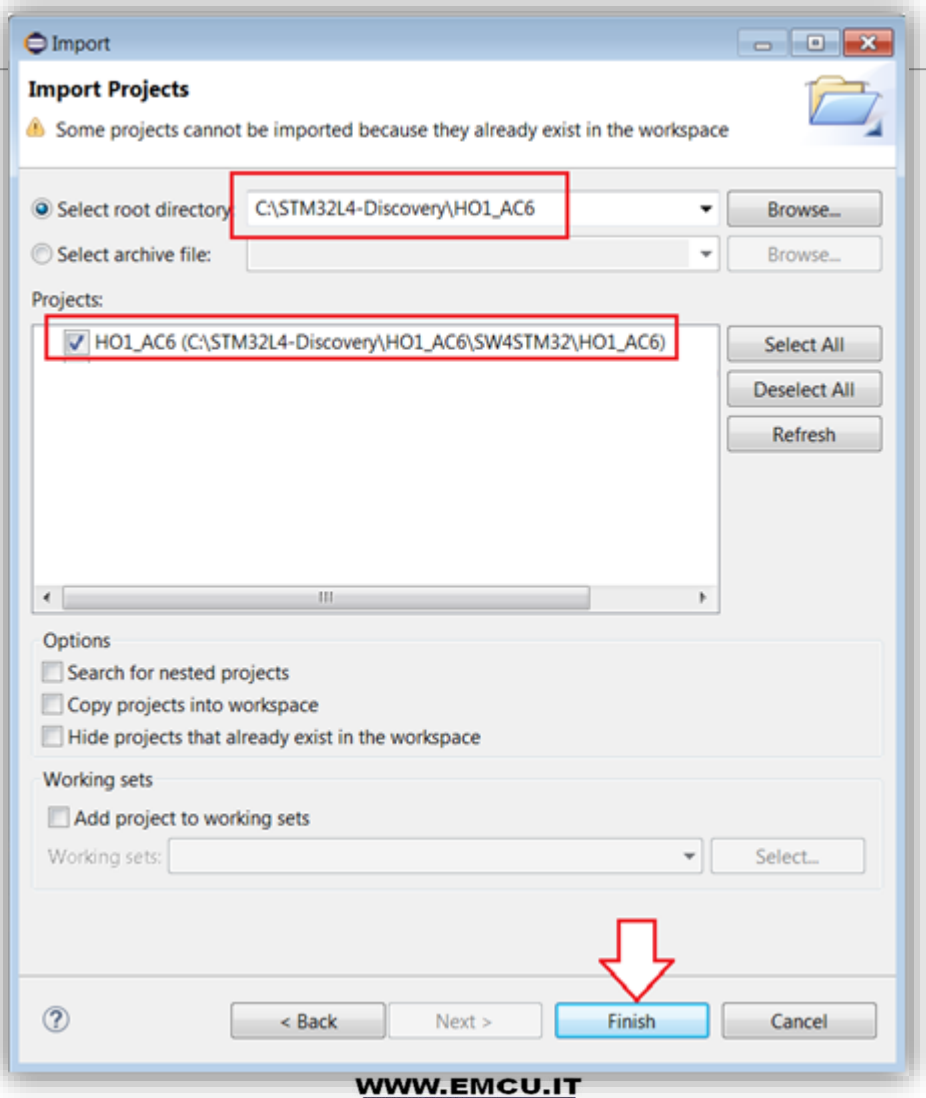
AC6 - (SW4STM32 - System Workbench)



AC6 - (SW4STM32 - System Workbench)

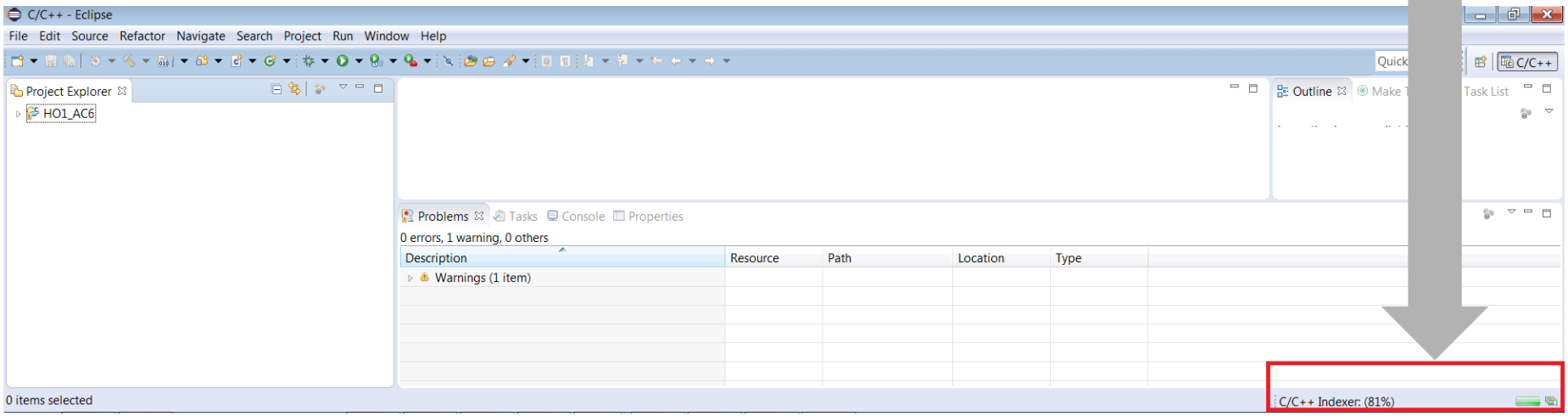


AC6 - (SW4STM32 - System Workbench)

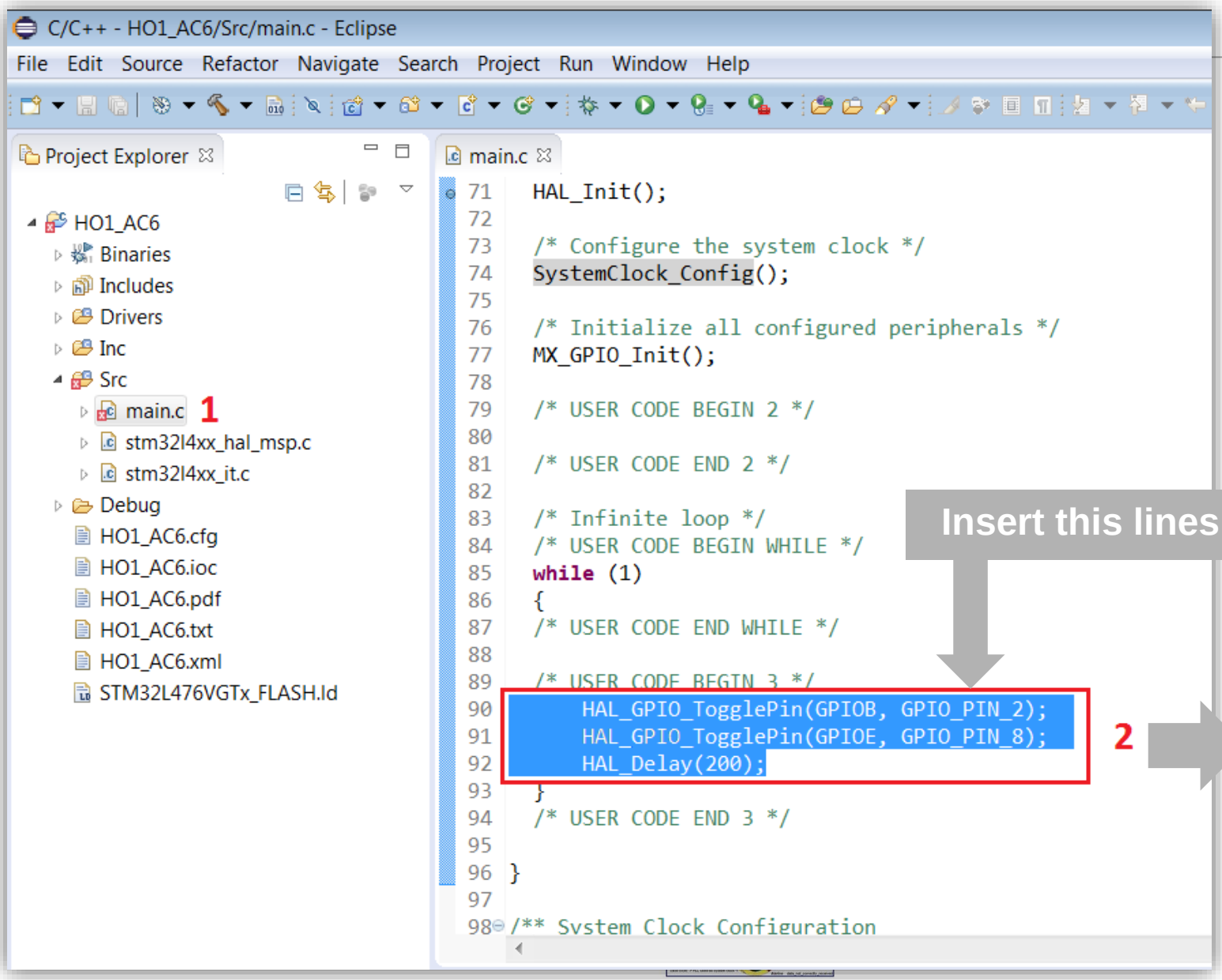


AC6 - (SW4STM32 - System Workbench)

Wait the import of the project



AC6 - (SW4STM32 - System Workbench)



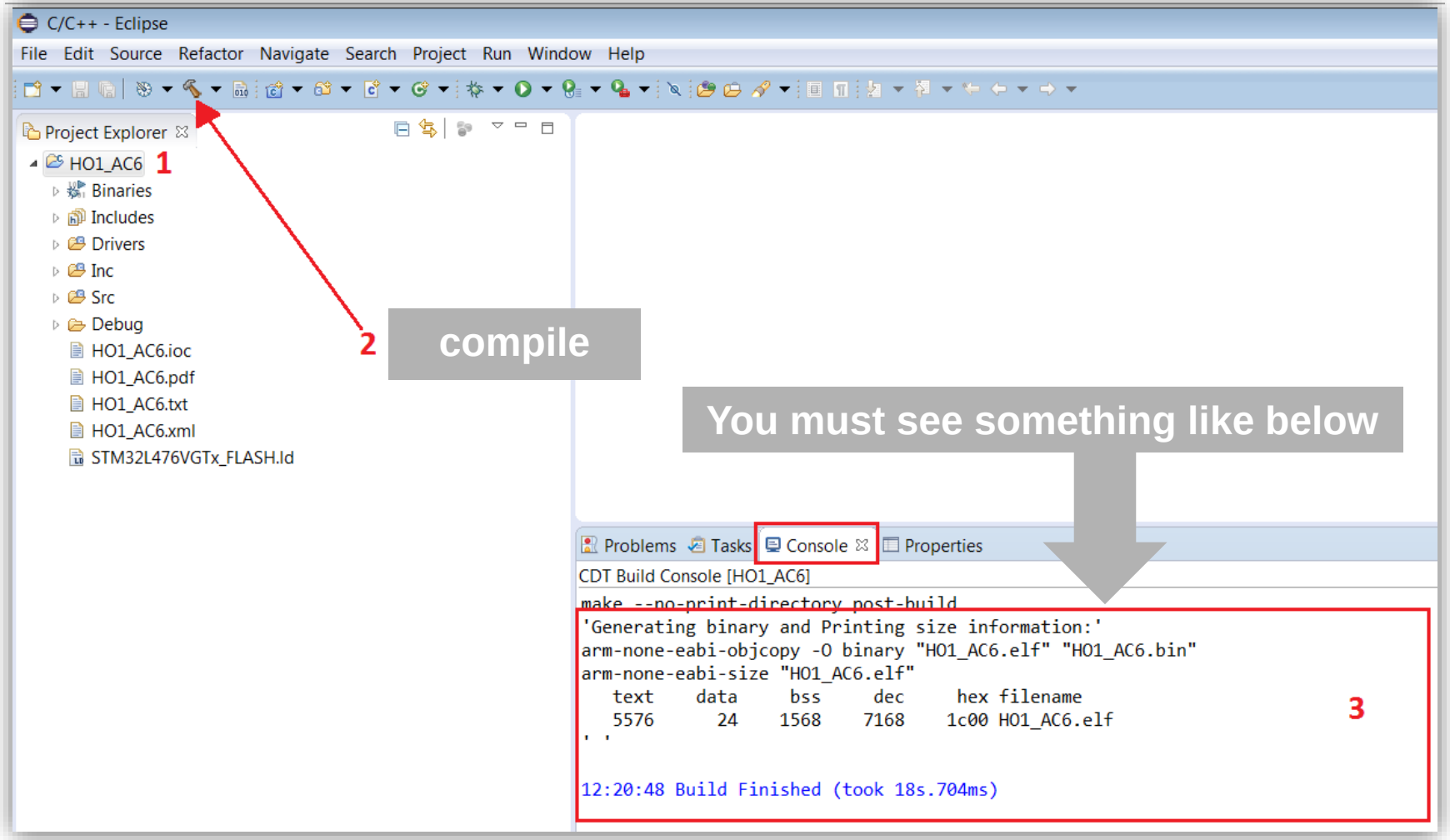
Insert this lines in main.c

2

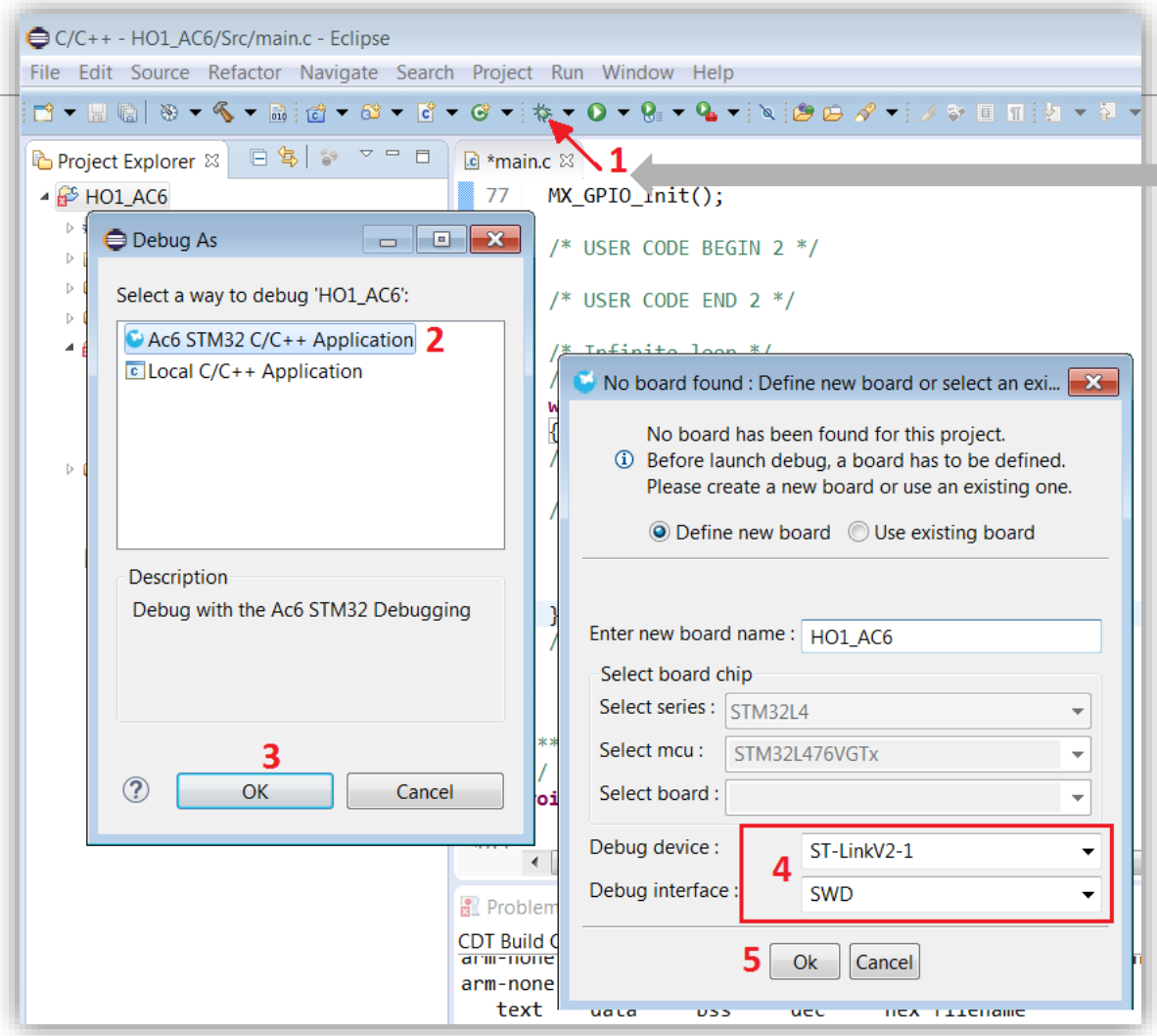
See the:
UM1884
Description of STM32L4 HAL and LL drivers

SILICA
An Avnet Company

AC6 - (SW4STM32 - System Workbench)

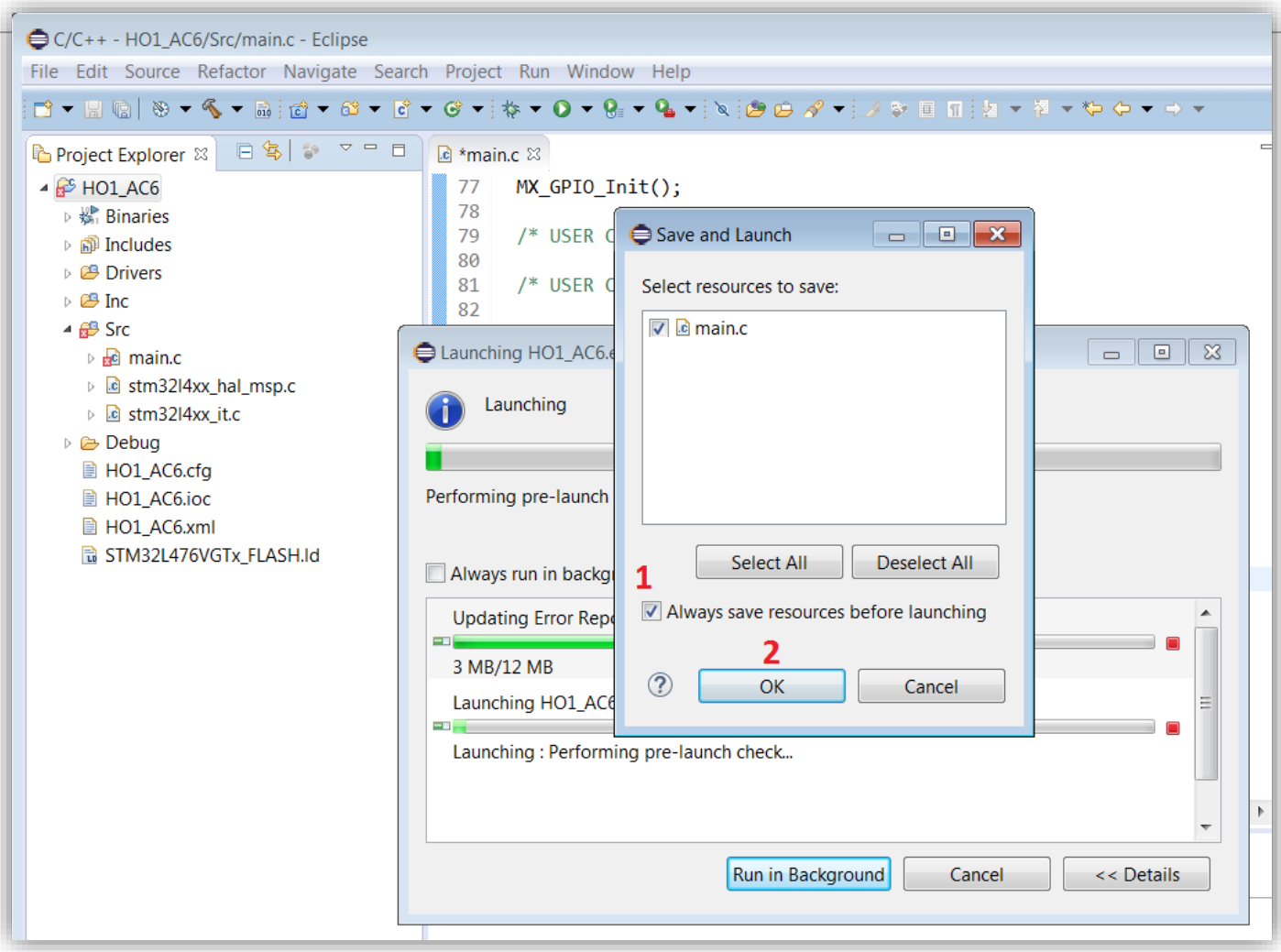


AC6 - (SW4STM32 - System Workbench)

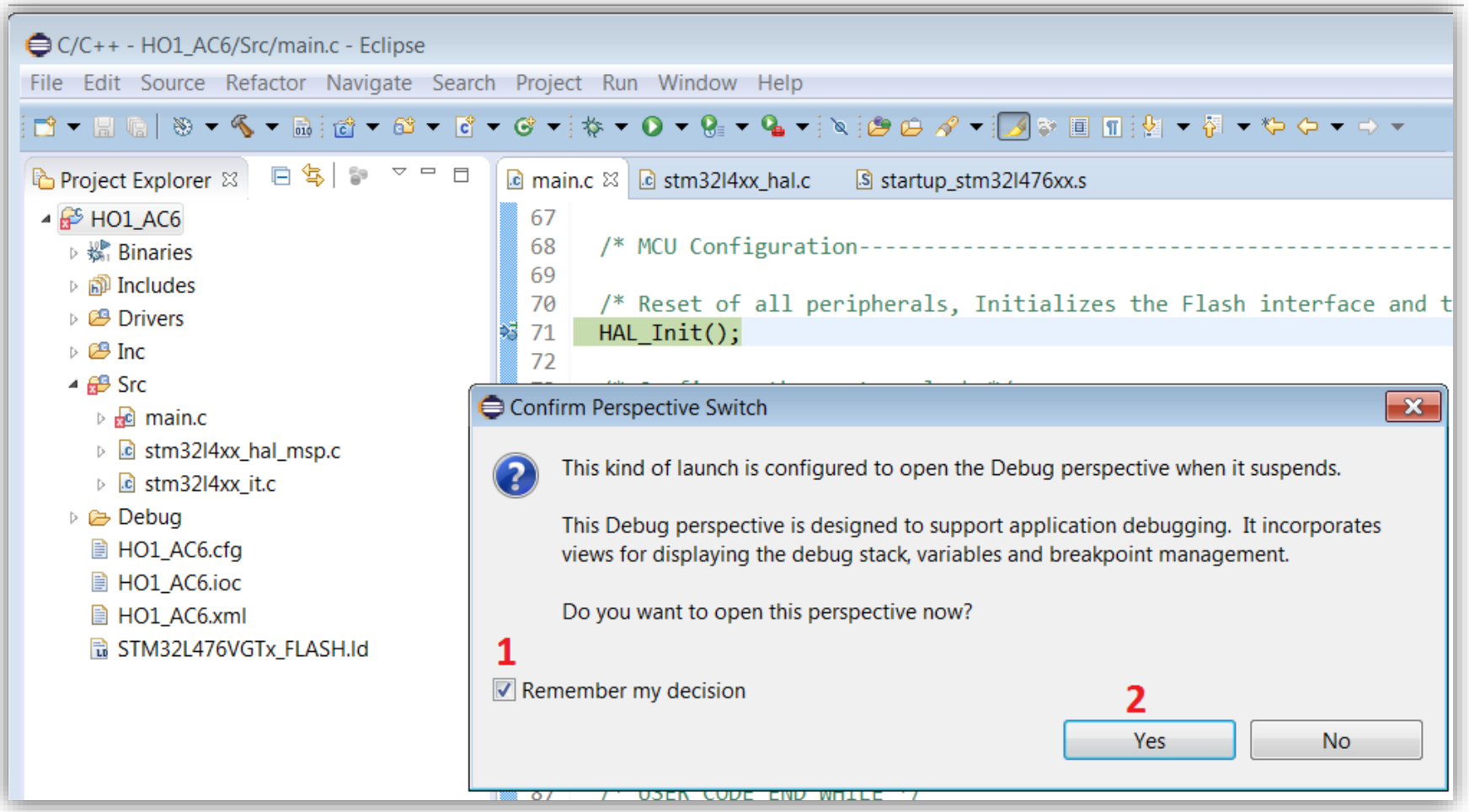


Enter in debug

AC6 - (SW4STM32 - System Workbench)

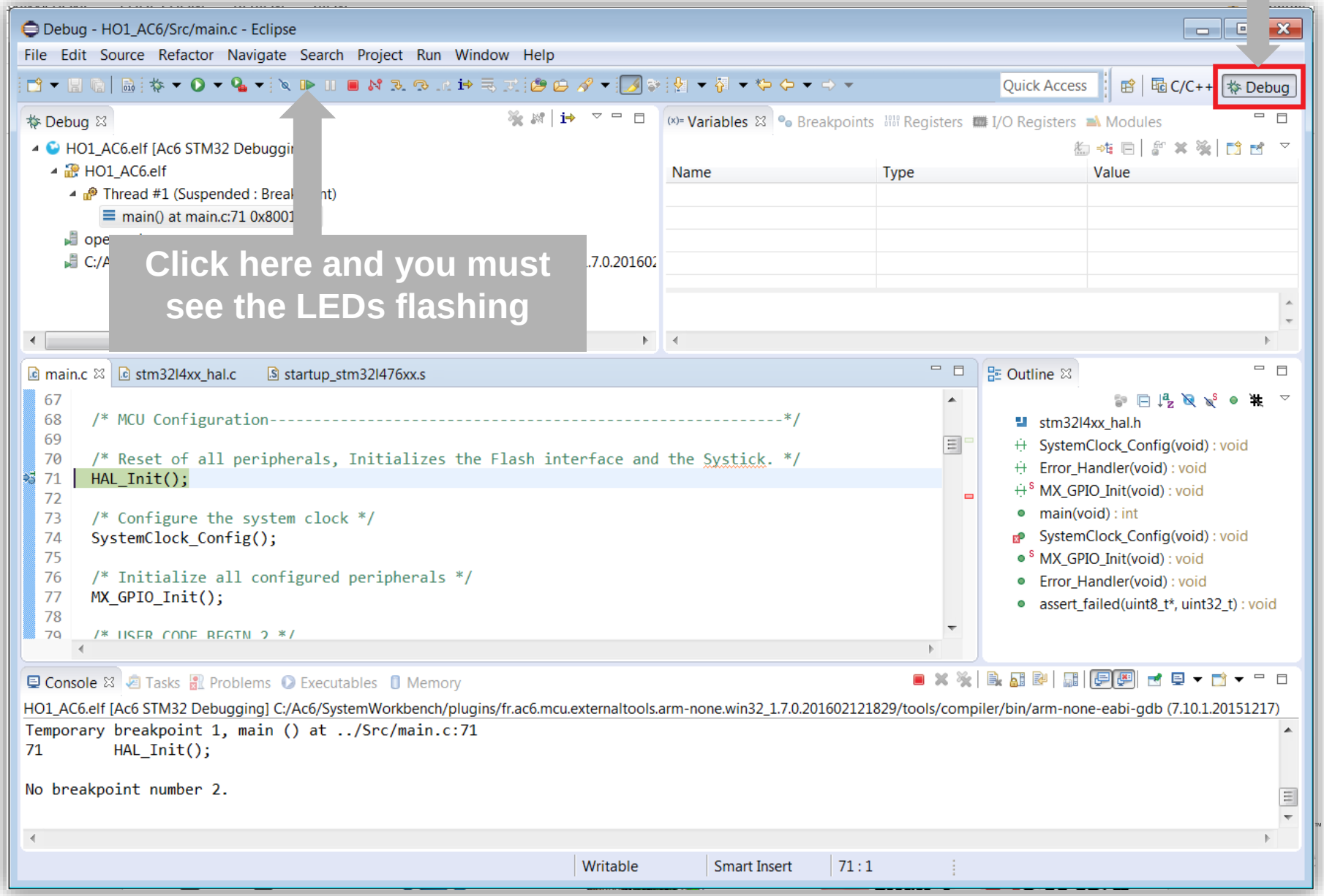


AC6 - (SW4STM32 - System Workbench)

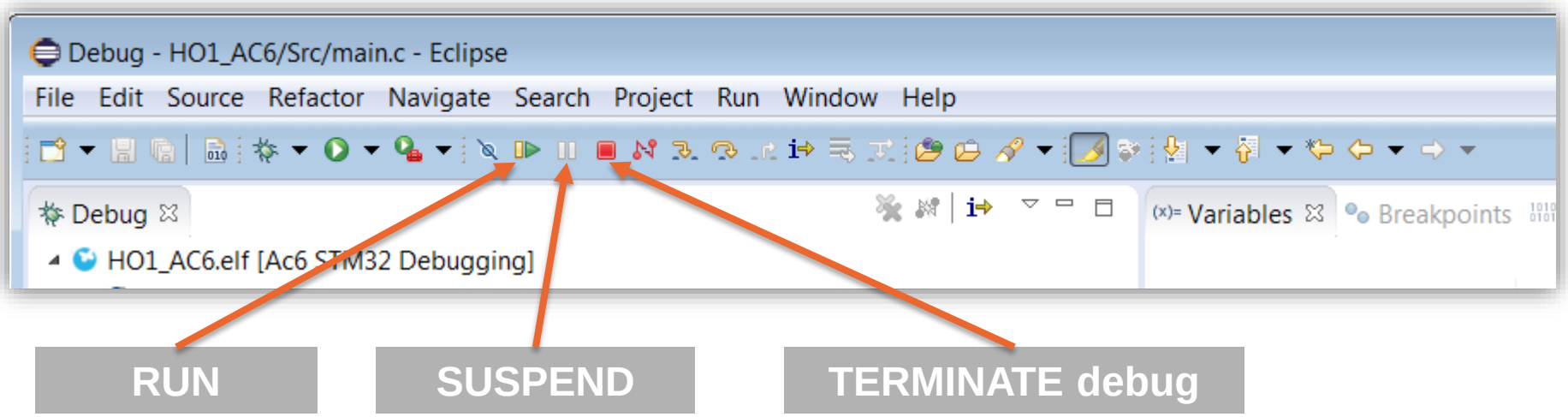


AC6 - (SW4STM32 - System Workbench)

Now you are in debug

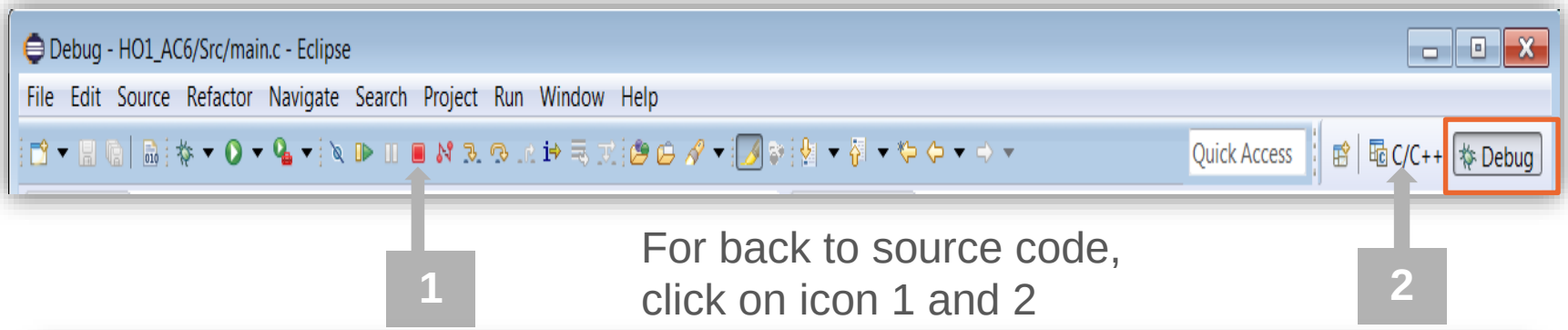


AC6 - (SW4STM32 - System Workbench)



For more information regarding **AC6** syntax, tutorial, etc, see **here**

AC6 - (SW4STM32 - System Workbench)



For back to source code,
click on icon 1 and 2