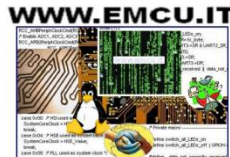




Presenter's name

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



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

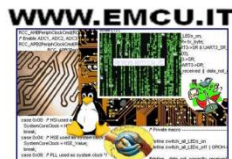
CUBE



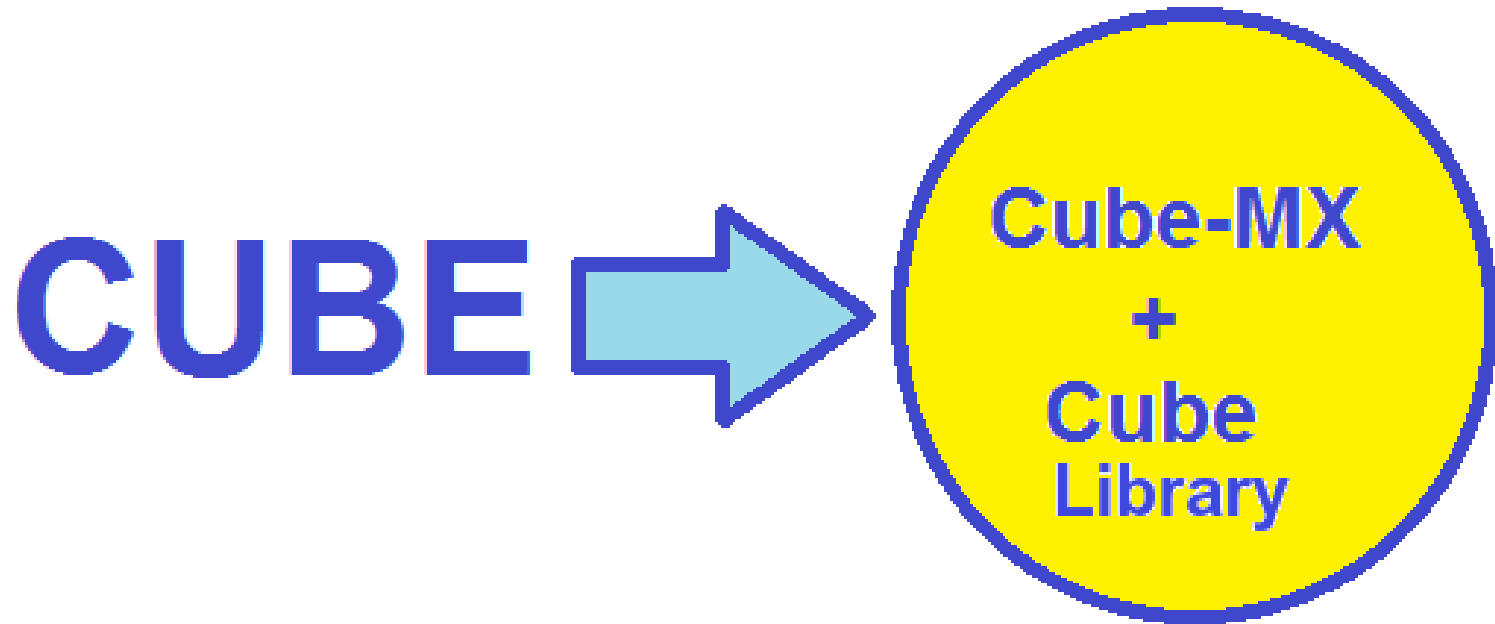
3



23 July 2016

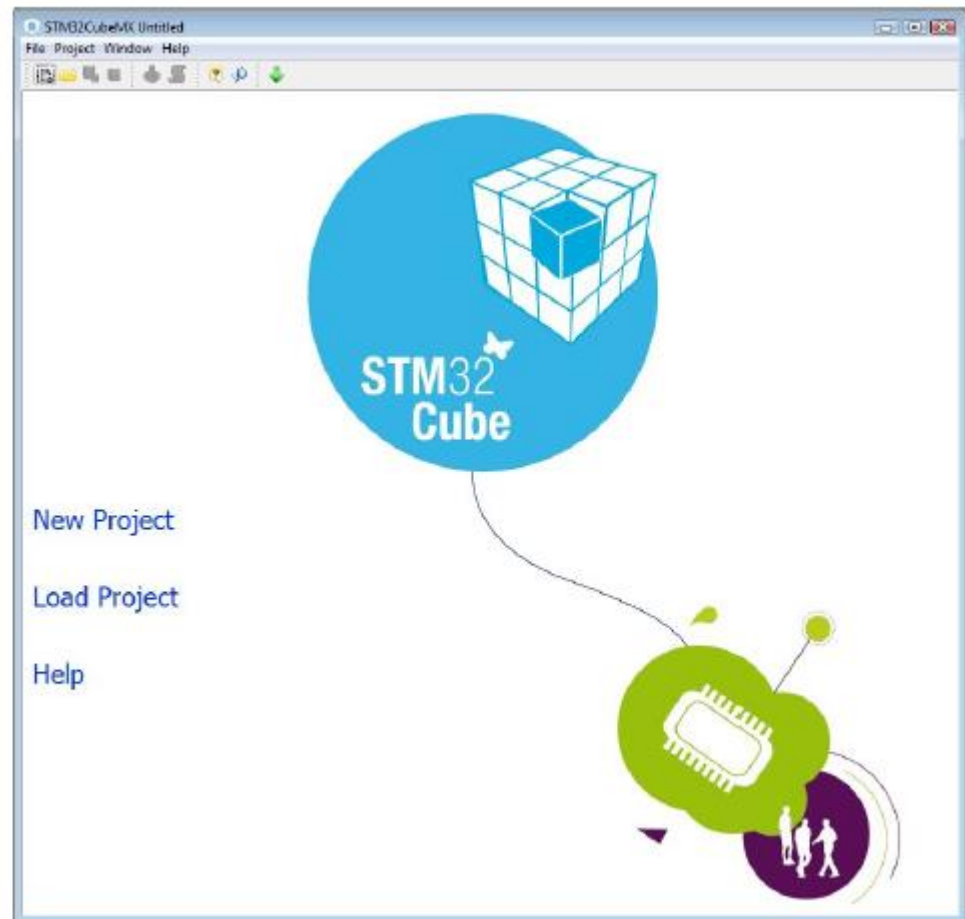


Introduction to CubeMX 1/3



Introduction to CubeMX 2/3

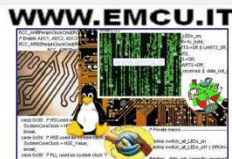
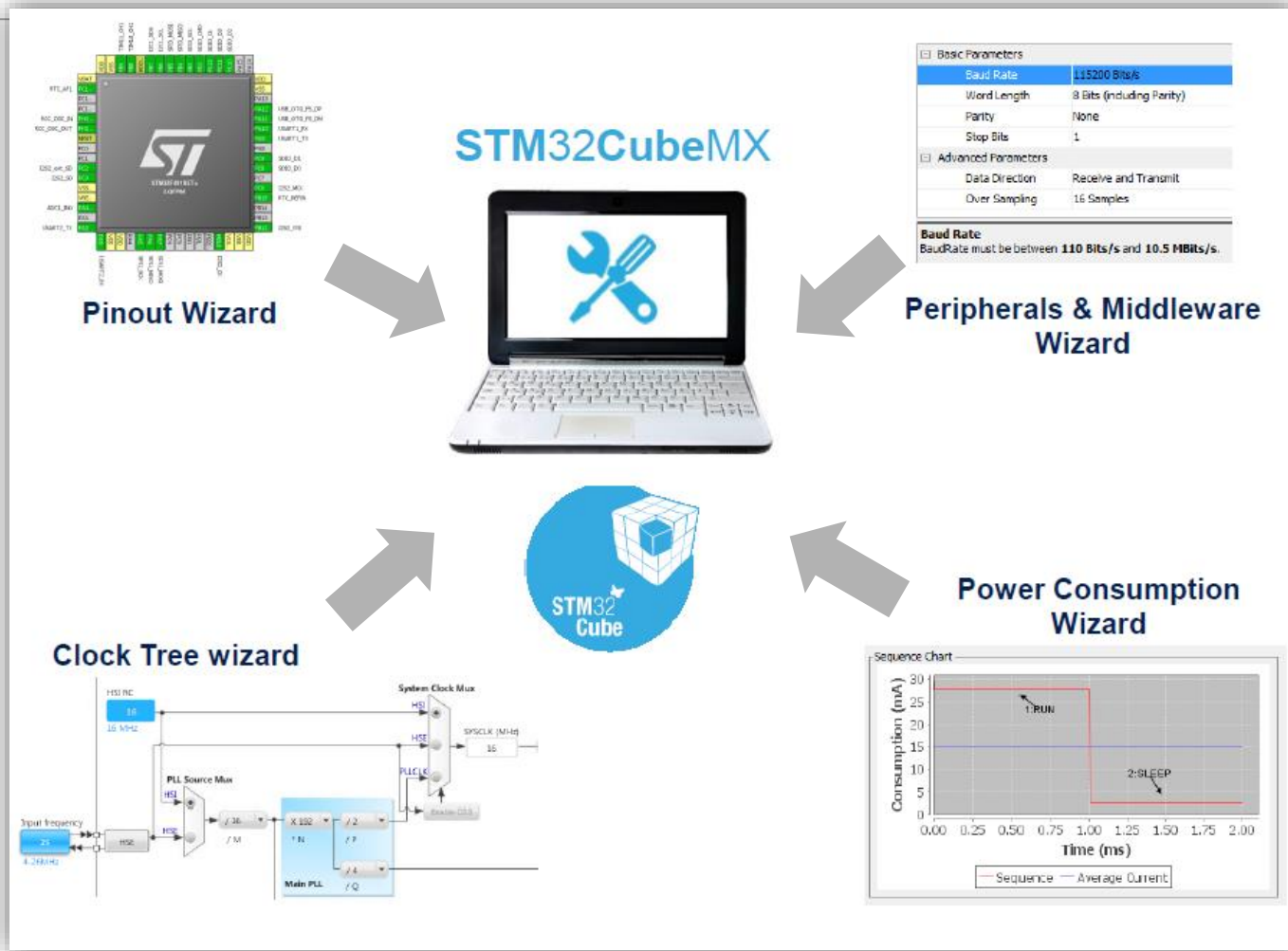
- MCU selector
- Pinout configuration
- Clock tree initialization
- Peripherals and middleware parameters
- Code generation
- Power consumption calculator



WWW.EMCU.IT

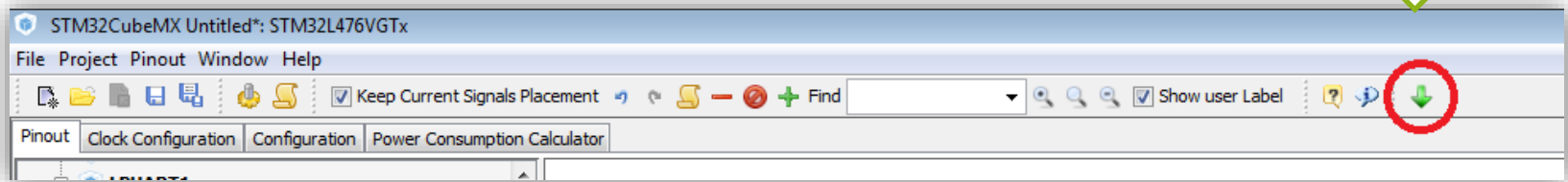


Introduction to CubeMX 3/3

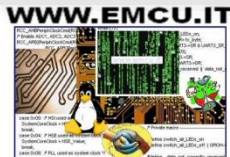
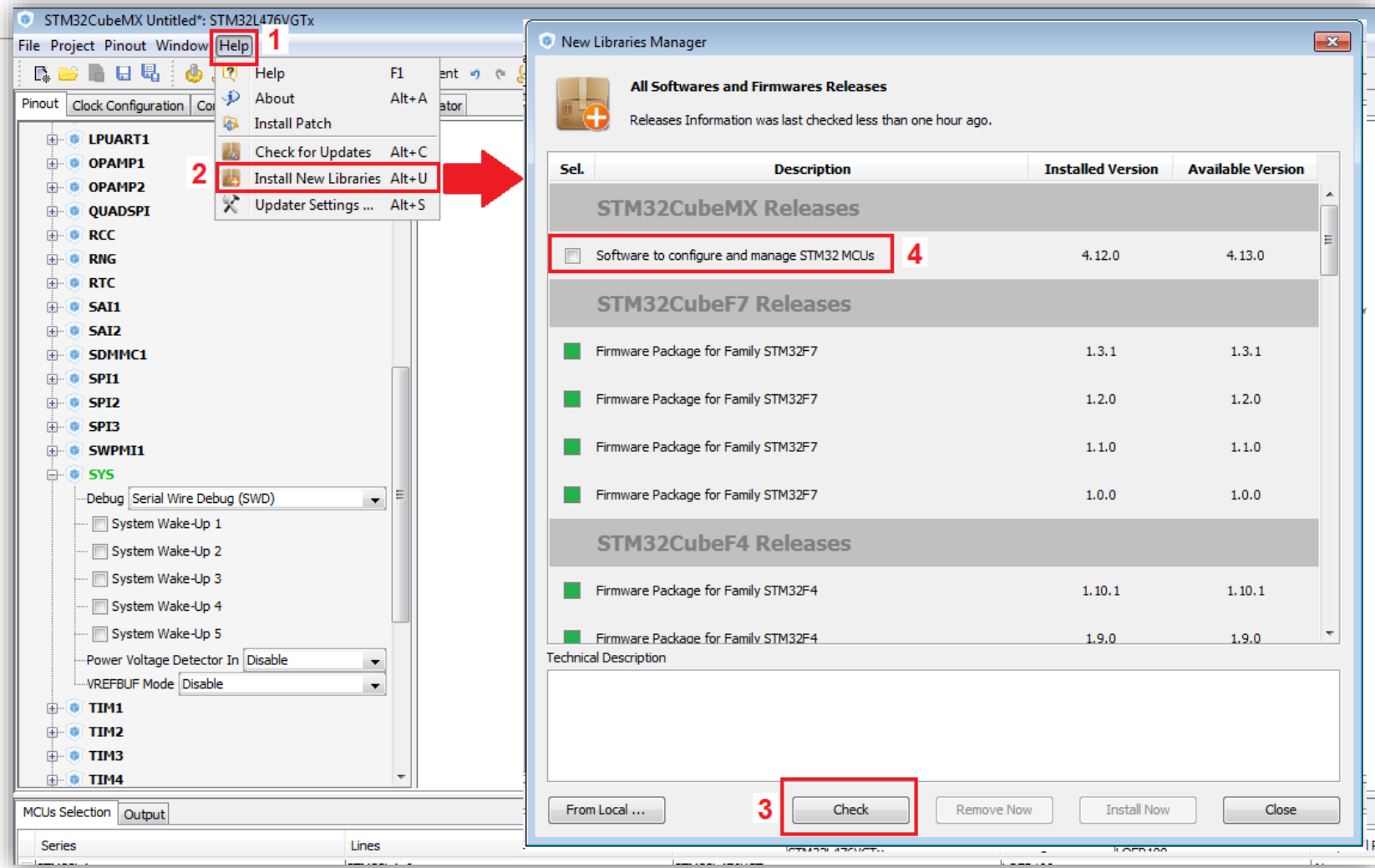


CubeMX request update 1/2

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



CubeMX request update 2/2



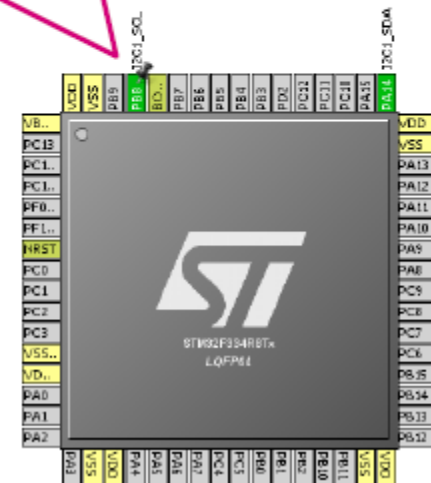
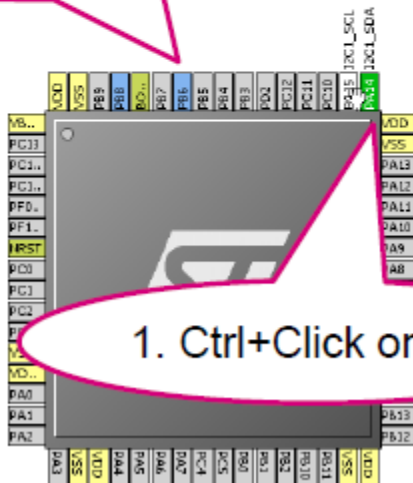
CubeMX: Pinout configuration

- 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)

2. Show alternative positions

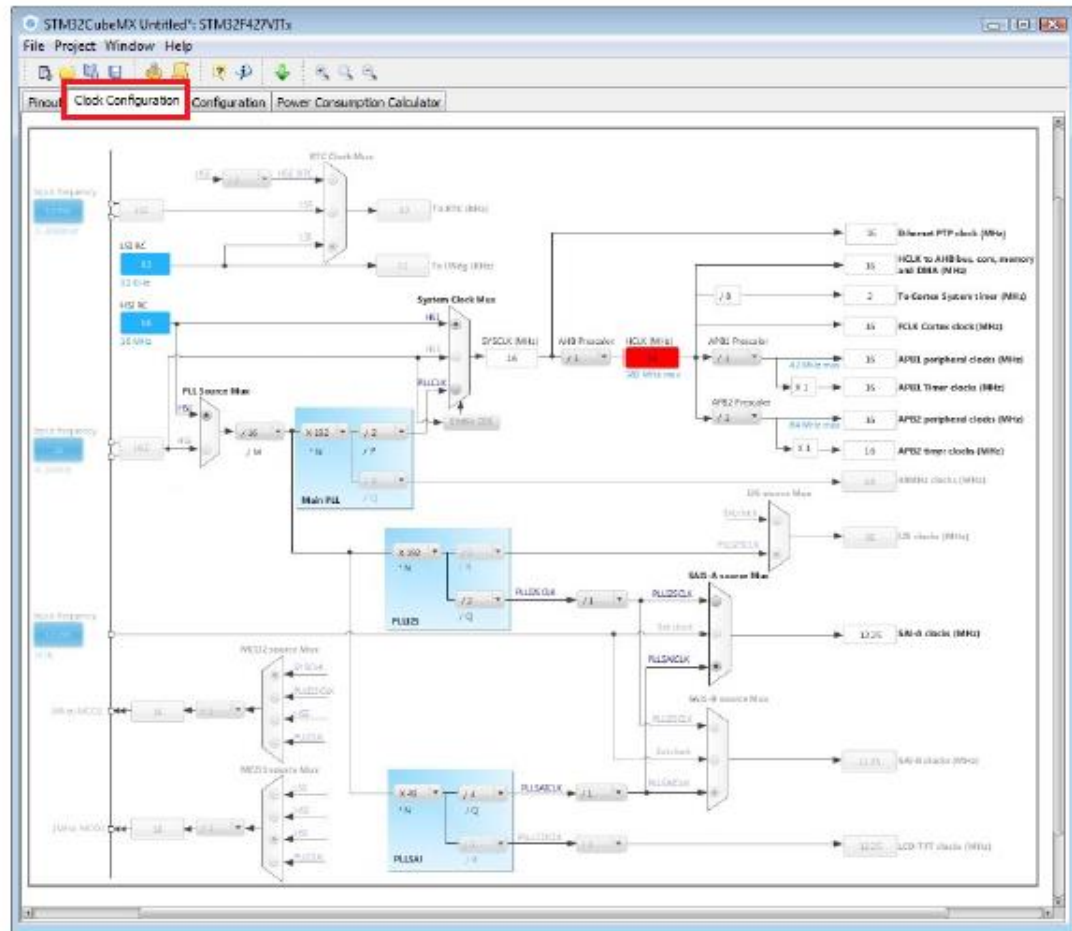
3. Move pin to new position

1. Ctrl+Click on pin



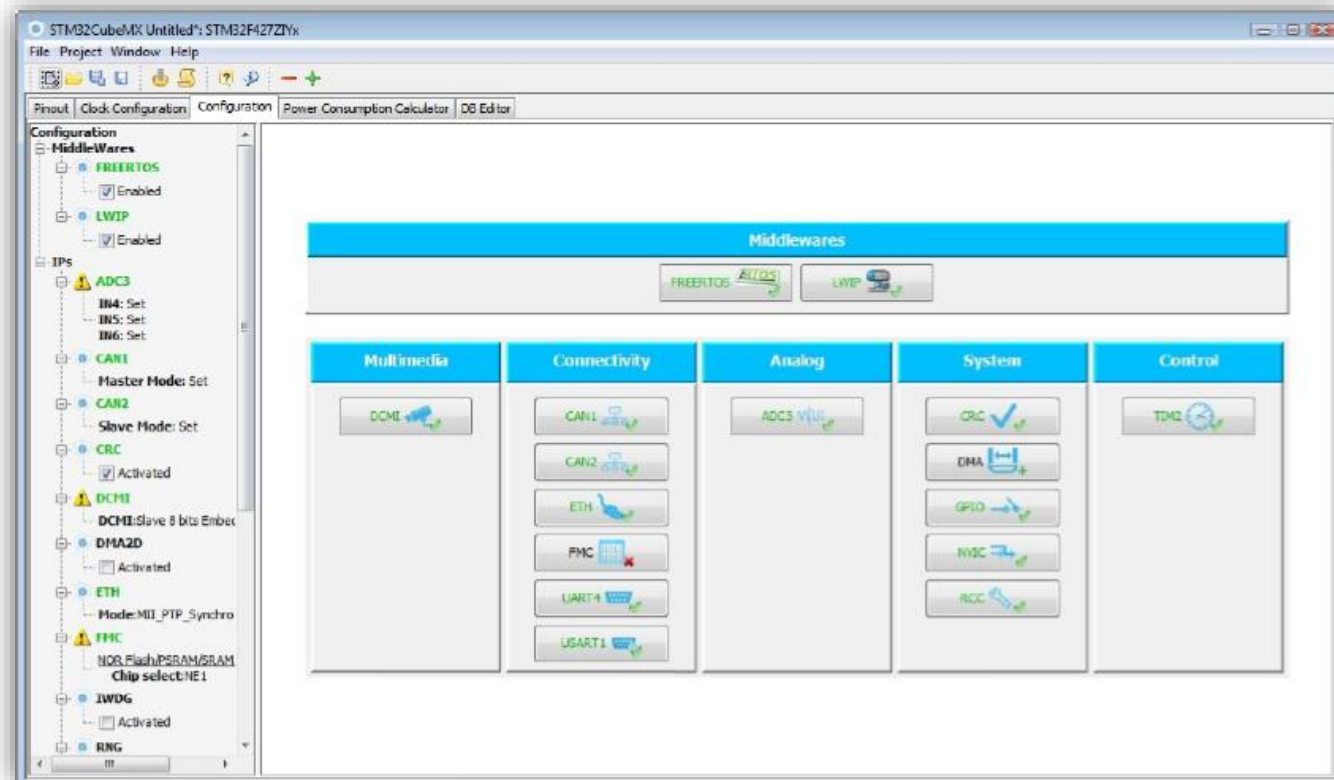
CubeMX: Clock tree

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



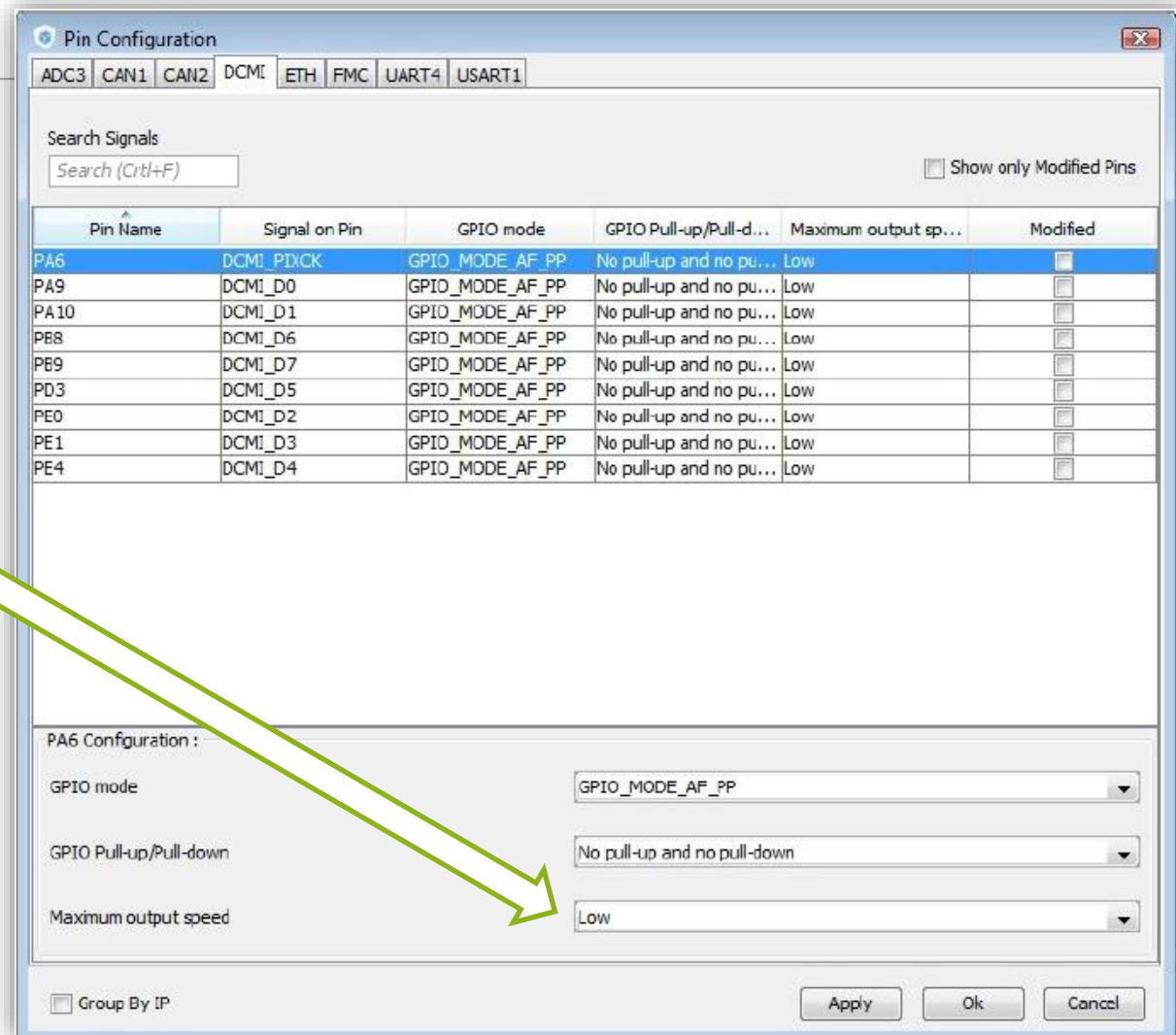
CubeMX: Peripheral and middleware configuration

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



CubeMX: GPIO Panel

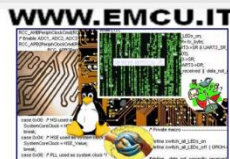
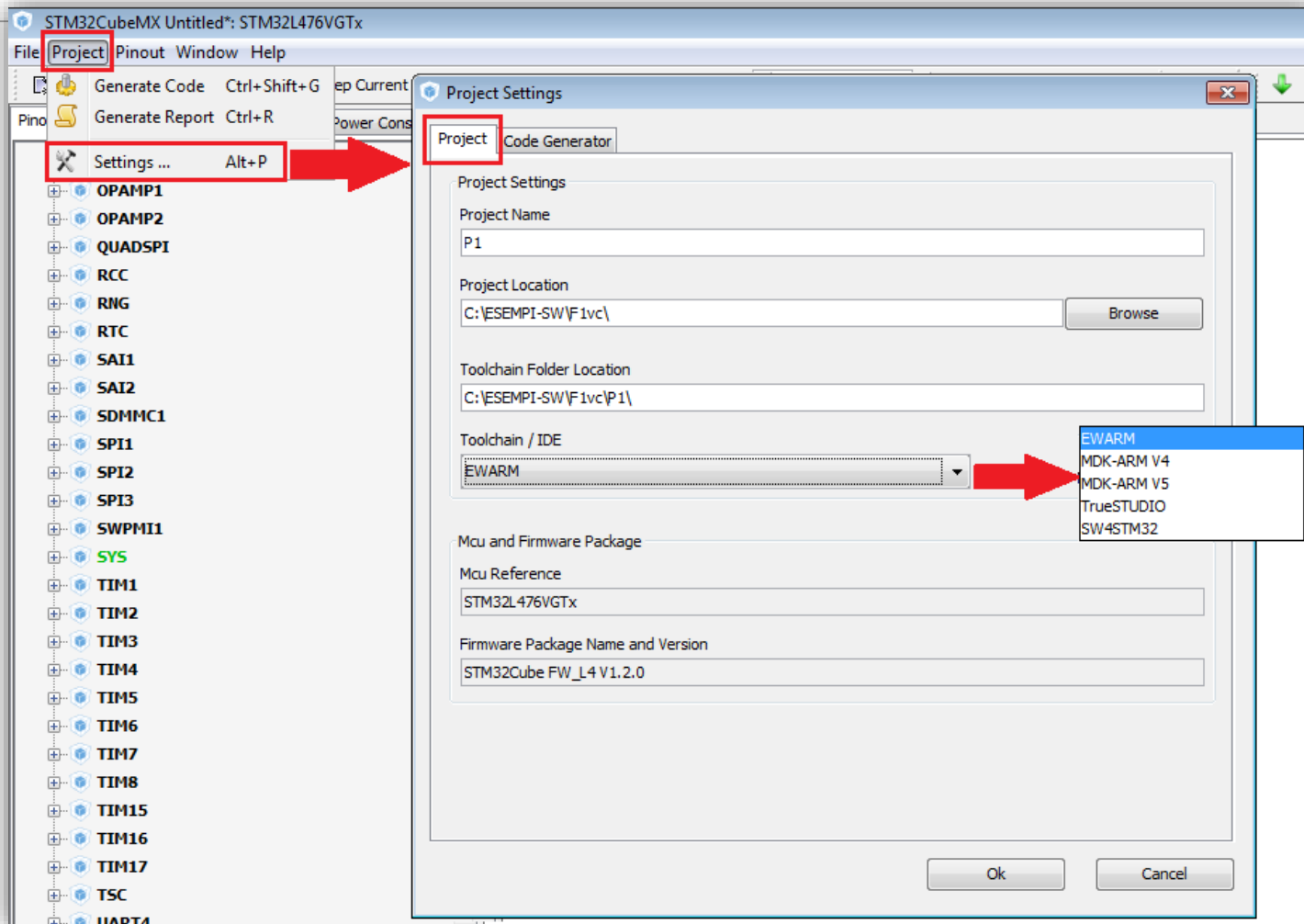
- Most of the GPIO parameters are set by default to the correct value
- You may want to change the maximum output speed
- You can select multiple pin at a time to set the same parameter



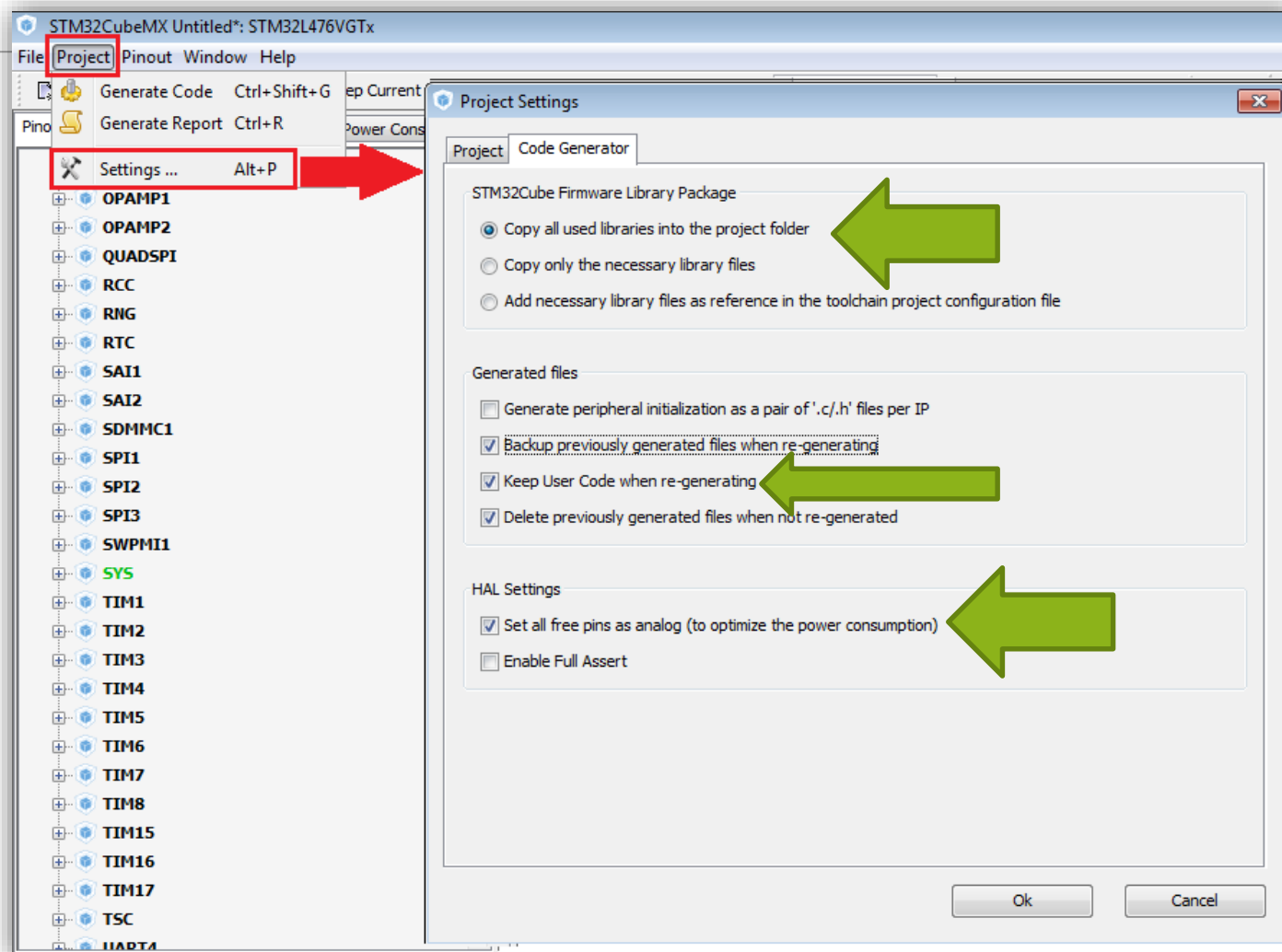
WWW.EMCU.IT



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

Generated files

- ☐ Generate peripheral initialization as a pair of '.c/.h' files per IP
- ☐ Backup previously generated files when re-generating
- ☒ Keep User Code when re-generating
- ☒ Delete previously generated files when not re-generated

```
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
38  /* Private function prototypes -----
39  static void SystemClock_Config(void);
40  static void StartThread(void const * argument);
41  static void MX_GPIO_Init(void);
42  static void MX_ADC1_Init(void);
43  static void MX_NVIC_Init(void);
44
45  int main(void)
46  {
47  /* USER CODE BEGIN 1 */
48
49  /* USER CODE END 1 */
50
51  /* MCU Configuration-----
52  /* Reset of all peripherals, Initializes the Flash interface
53  HAL_Init();
54  /* Configure the system clock */
```

WWW.EMCU.IT



CubeMX: Power consumption calculator

STM32CubeMX Fiorentini-STM32L053C6.ioc: STM32L053C6Tx

File Project Power Window Help

Pinout Clock Configuration Configuration Power Consumption Calculator

Microcontroller Selected

Series: STM32L0
Line: STM32L0x3
MCU: STM32L053C6Tx
[Datasheet:](#) 025844_Rev4

Parameter Selection

Ambient Temperature (°C): 25
Vdd Power Supply (V): 3.0

Battery Selection

Select

Battery: LI-SOCL2(A3400)
In Series: 1
In Parallel: 1
Capacity: 3400.0 mAh
Self Discharge: 0.08 %/month
Nominal Voltage: 3.6 V
Max Cont Current: 100.0 mA
Max Pulse Current: 200.0 mA

Information Notes

Help

Sequence

Load Save Delete Compare

Transitions checker
☐ Enabled Show log

Sequence Table

Step	Mode	Vdd	Range/Scale	Memory	CPU/Bus Freq	Clock Config	Src Freq	Peripherals	Add. Current	Step Current	Duration	DMIPS	Volta...	Ta ...	C...
1	RUN	3.0	Range2-Medium	RAM	4.0 MHz	HSEBYP_4MHz PLL_OFF	4.0 MHz	GPIOA GPI...	0 mA	615 µA	3 ms	3.812	Battery	85.0	Da...
2	RUN	3.0	Range1+High	FLASH	8.0 MHz	HSEBYP_8MHz PLL_OFF	8.0 MHz	GPIOA GPI...	0 mA	1.77 mA	1 ms	7.624	Battery	85.0	Da...
3	STOP	3.0	NoRange	n/a	0 Hz	LSE RTC_ON IWDG_O...	32.768 kHz	GPIOA GPI...	0 mA	4 µA	100 ms	0.0	Battery	85.0	Da...

Step

Add Delete Duplicate Up Down Undo Redo

Display
Plot: All Steps Ext. Display

Results Charts

Consumption Profile by Step

Consumption (mA)

Time (ms)

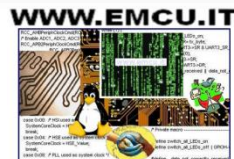
1: RUN 2: RUN 3: STOP

Idd by Step Average Current

Results Summary

Sequence Time / Ta Max 104 ms / 85.0 °C
Battery Life Estimation 9 years, 1 month, 27 days & 11 hours

Average Consumption 38.61 µA
Average DMIPS 4.75 DMIPS



HAL library

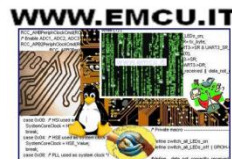
HAL library



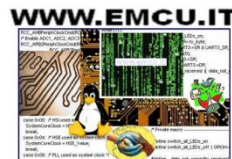
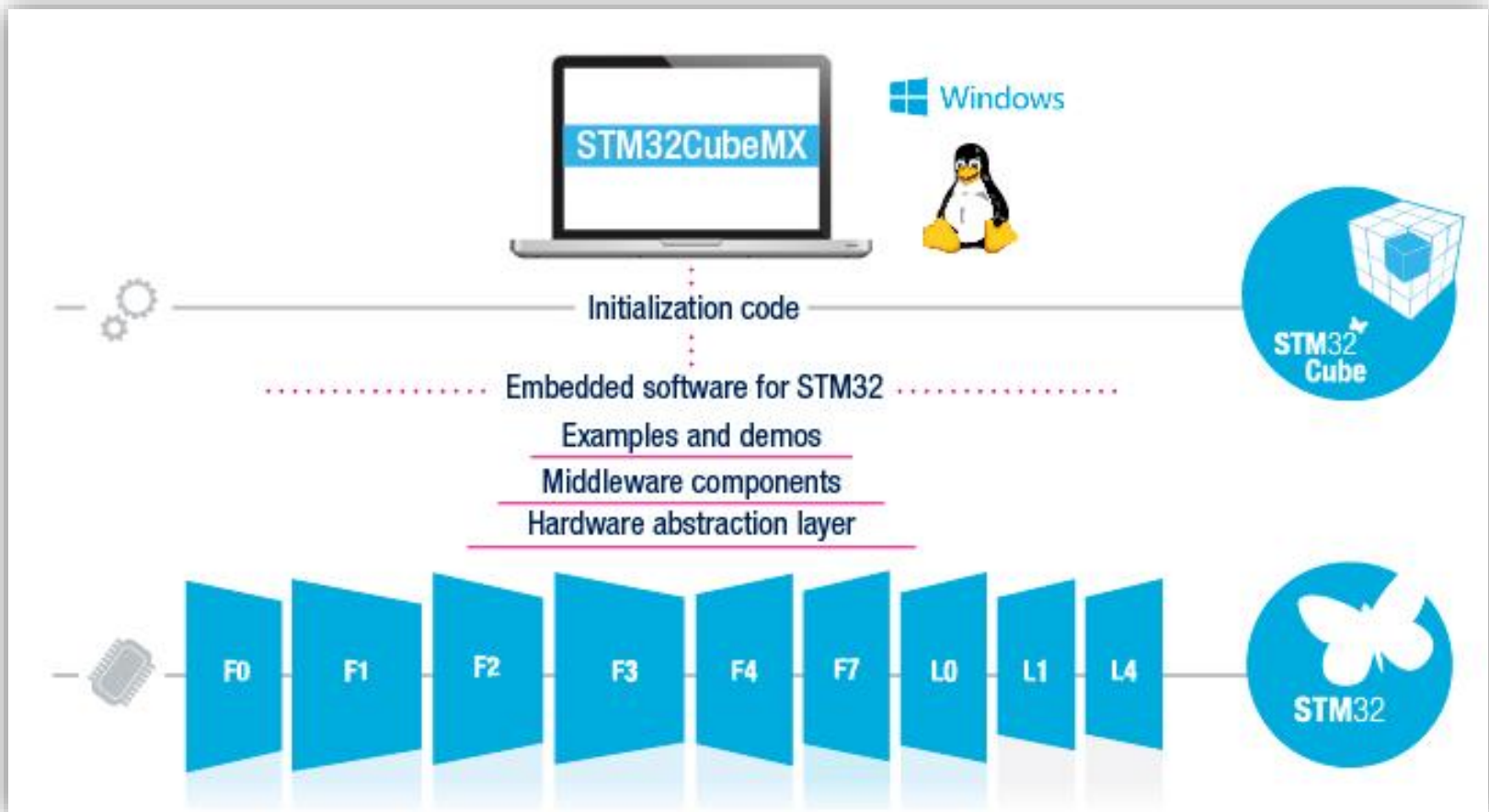
17



23 July 2016

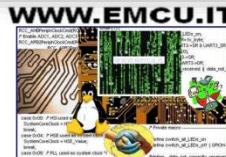
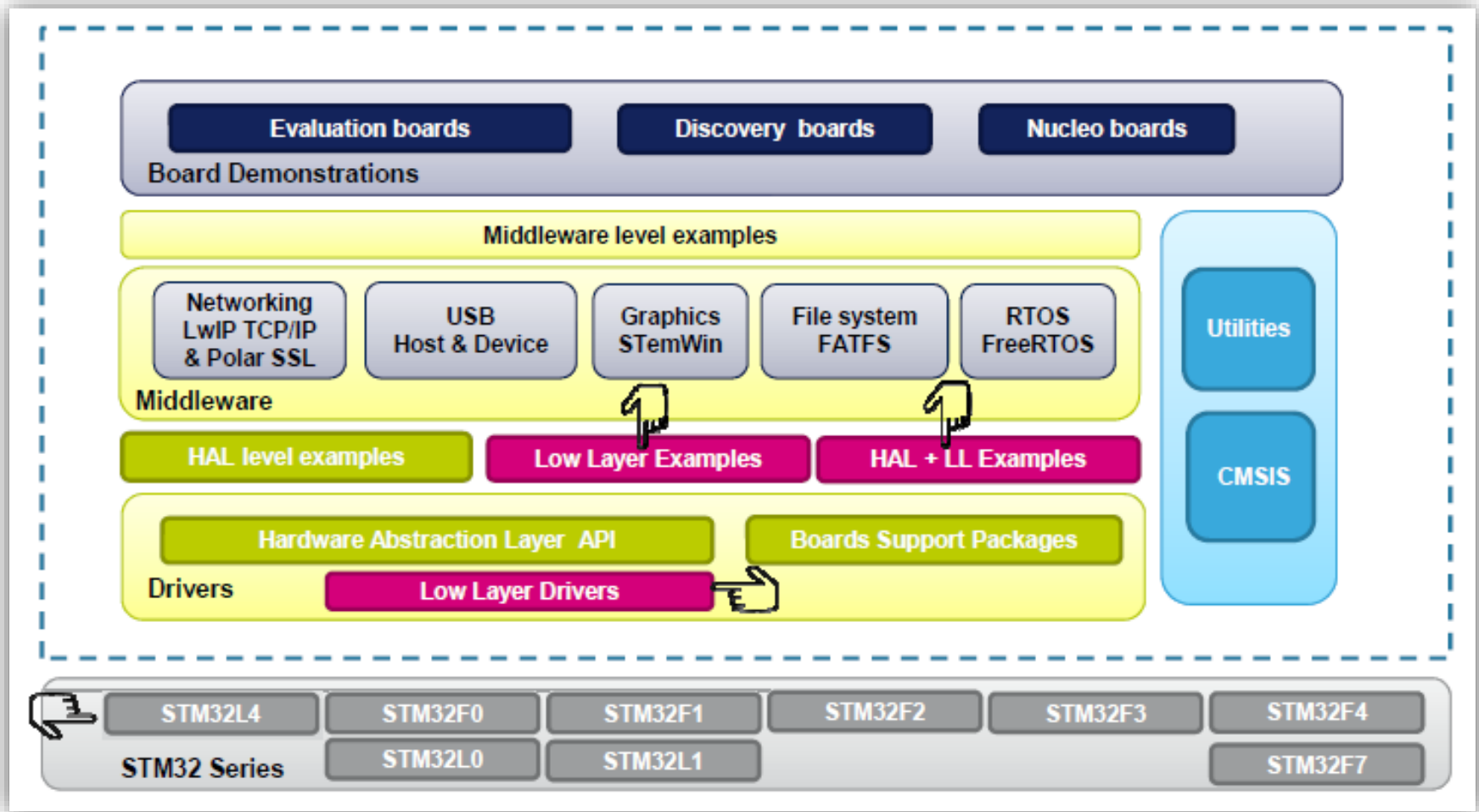


HAL library – HAL == hardware abstraction layer



HAL library

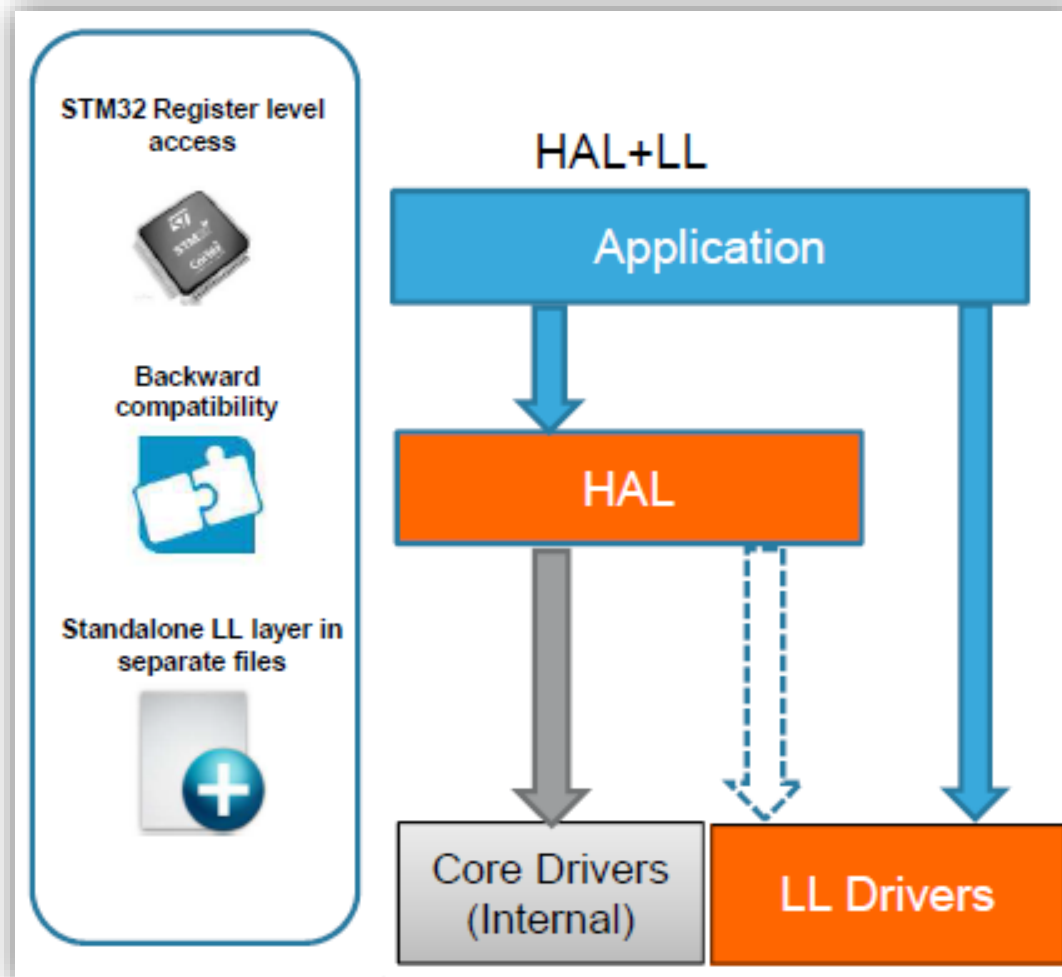
The HAL library are [here](#).



HAL library

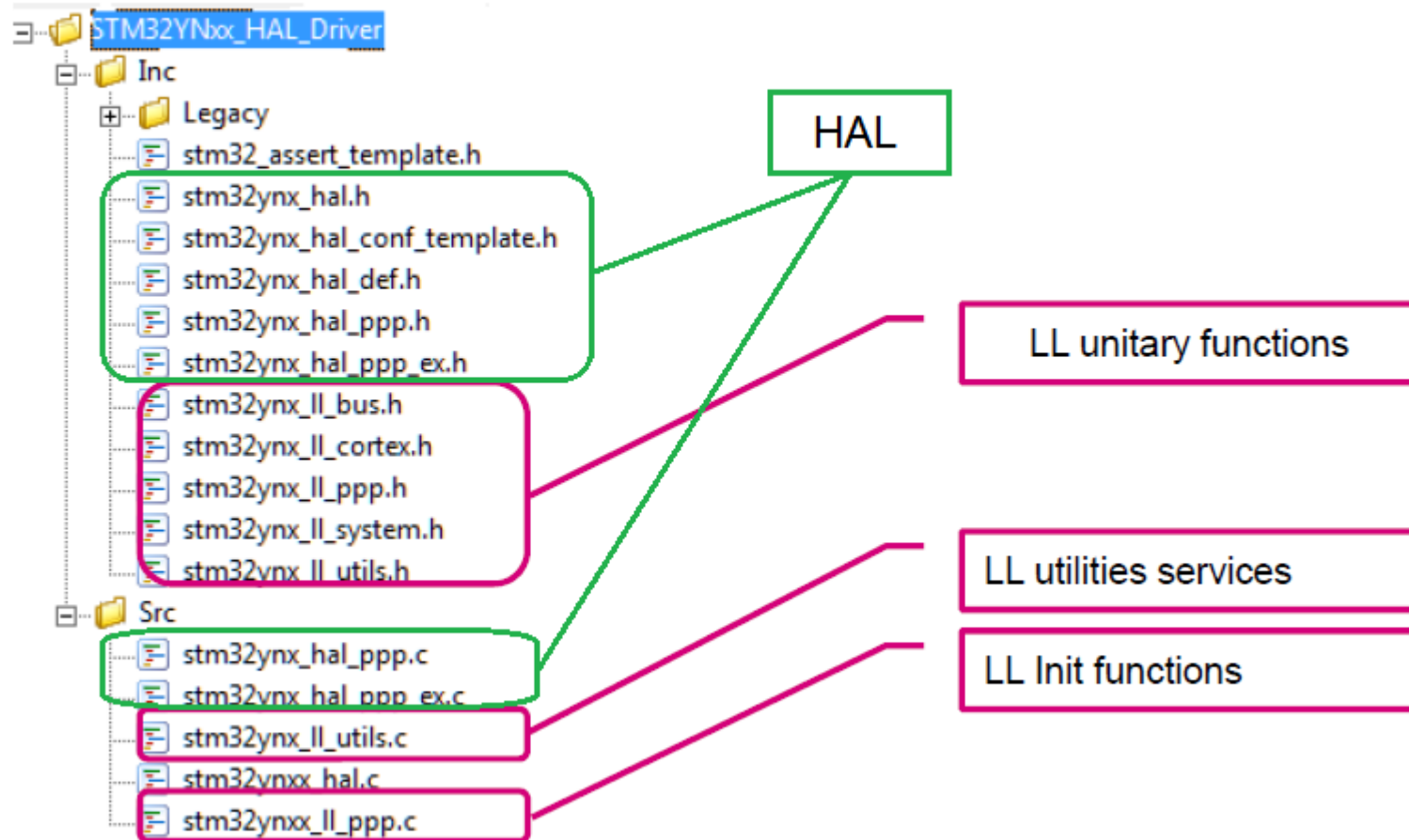
- **STM32Cube HAL & LL** are complementary and covers a wide range of applications requirements:
 - **HAL** offers **high level** and **functionalities oriented APIs**, with **high portability level** and **hide product/IPs complexity** to end user
 - **LL** offers **low level APIs** at **registers level**, 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

LL drivers are located in the Src/Inc HAL Driver folders



HAL library

Covered peripherals (1/2)

Peripherals (IPs)		STM32Cube Support	
System	Flash	HAL Yes	LL 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
	DMA	Yes	Yes
	PWR	Yes	Yes
	RCC	Yes	Yes
	Cortex	Yes	No (some of the cortex features added: MPU, SYSTICK, CPUID, SLEEPDEEP)
Analog	SYSCFG	Yes	Yes
	ADC	Yes	Yes
	SDADC	Yes	Yes
	DAC	Yes	Yes
	COMP	Yes	Yes
	DFSDM	Yes	No
Timers	OPAMP	Yes	Yes
	RTC	Yes	Yes
	TIM	Yes	Yes
	LPTIM	Yes	Yes
	HRTIM	Yes	Yes
	WWDG	Yes	Yes
Cryptography	IWDG	Yes	Yes
	CRC	Yes	Yes
	CRYP	Yes	No
	HASH	Yes	No
	RNG	Yes	Yes

WWW.EMCU.IT



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

WWW.EMCU.IT

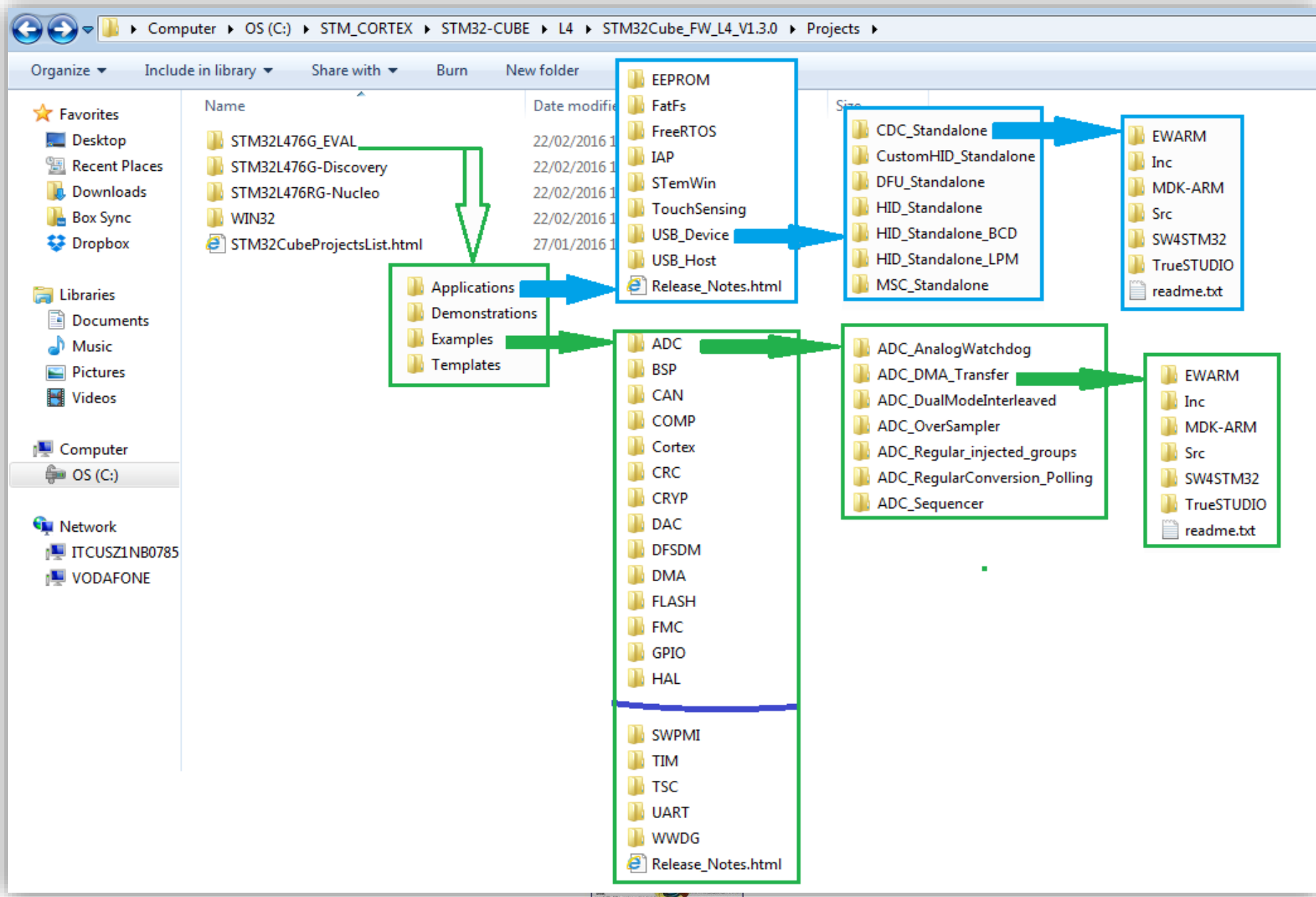


HAL library

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 mirrored 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 intended 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

The image displays two screenshots from the STM32CubeIDE software. The left screenshot shows the 'Project Explorer' with the 'Examples' folder expanded. A callout box points to the 'Examples' folder, stating: 'Examples that are based ONLY on HAL drivers (as of today)'. Below this, a list of examples is shown, with 'DMA_FLASHToRAM' and 'FLASH' highlighted by a pink box. The right screenshot shows the 'Files' view of a project named 'STM32L476RG_NUCLEO'. The project structure includes folders for 'Doc', 'Drivers', 'BSP', 'CMSIS', 'STM32L4xx_HAL_Driver', 'Example', 'EWARM', 'User', and 'Output'. The file 'stm32l4xx_hal_dma.c' is highlighted with a pink box. Below the screenshots, the text 'HAL project (no LL services used in the application)' is displayed.

Examples that are based ONLY on HAL drivers (as of today)

ADC
COMP
Cortex
CRC
DAC
DMA
DMA_FLASHToRAM
FLASH

Files

Project - STM32L476RG_NUCLEO ✓

Doc

Drivers

BSP

CMSIS

STM32L4xx_HAL_Driver

stm32l4xx_hal.c

stm32l4xx_hal_cortex.c

stm32l4xx_hal_dma.c

stm32l4xx_hal_gpio.c

stm32l4xx_hal_pwr.c

stm32l4xx_hal_pwr_ex.c

stm32l4xx_hal_rcc.c

stm32l4xx_hal_rcc_ex.c

Example

EWARM

User

main.c

stm32l4xx_it.c

Output

HAL project (no LL services used in the application)



HAL library – LL examples

NEW Examples that are based ONLY on LL drivers

Only LL drivers (.h) are used in the application

Examples_LL

- ADC
- COMP
- CORTEX
- CRC
- DAC
- DMA
- EXTI
- GPIO
- I2C
- INVDG
- LPTIM
- LPUART
- OPAMP
- PWR
- RCC
- RNG
- RTC
- SPI
- SWPMI
- TIM
- USART
- UTILS
- WWDG

Examples_MIX

- ADC
- CRC
- DMA
- I2C
- OPAMP
- PWR
- SPI
- TIM
- UART

Release_Notes.html

Project - STM32L476RG_NUCLEO

- Doc
- Drivers
- CMSIS
- STM32L4xx_LL_Driver
- stm32l4xx_utils.c
- Example
- EWARM
- User
- main.c
- Output
- cmsis_lsr.h
- core_cm4.h
- core_cmFunc.h
- core_cmInstr.h
- core_cmSimd.h
- DLlib_Config_Full.h
- DLlib_Defaults.h
- DLlib_Product.h
- DLlib_Threads.h
- intrinsics.h
- main.h
- stdint.h
- stm32l4xx.h
- stm32l4xx_ll.h
- stm32l4xx_ll_dma.h
- stm32l4xx_ll_gpio.h

ADC

- ADC_AnalogWatchdog
- ADC_ContinuousConversion_TriggerSW
- ADC_ContinuousConversion_TriggerSW_Init
- ADC_ContinuousConversion_TriggerSW_LowPower
- ADC_GroupsRegularInjected
- ADC_MultiChannelSingleConversion
- ADC_MultimodeDualInterleaved
- ADC_Oversampling
- ADC_SingleConversion_TriggerSW
- ADC_SingleConversion_TriggerSW_DMA
- ADC_SingleConversion_TriggerSW_IT
- ADC_SingleConversion_TriggerTimer_DMA
- ADC_TemperatureSensor

COMP

- COMP_CompareWithInternalReference_IT
- COMP_CompareWithInternalReference_IT_Init

CORTEX

CRC

DAC

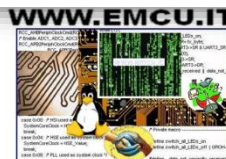
- DAC_GenerateConstantSignal_TriggerSW
- DAC_GenerateConstantSignal_TriggerSW_LP
- DAC_GenerateWaveform_TriggerHW
- DAC_GenerateWaveform_TriggerHW_Init

DMA

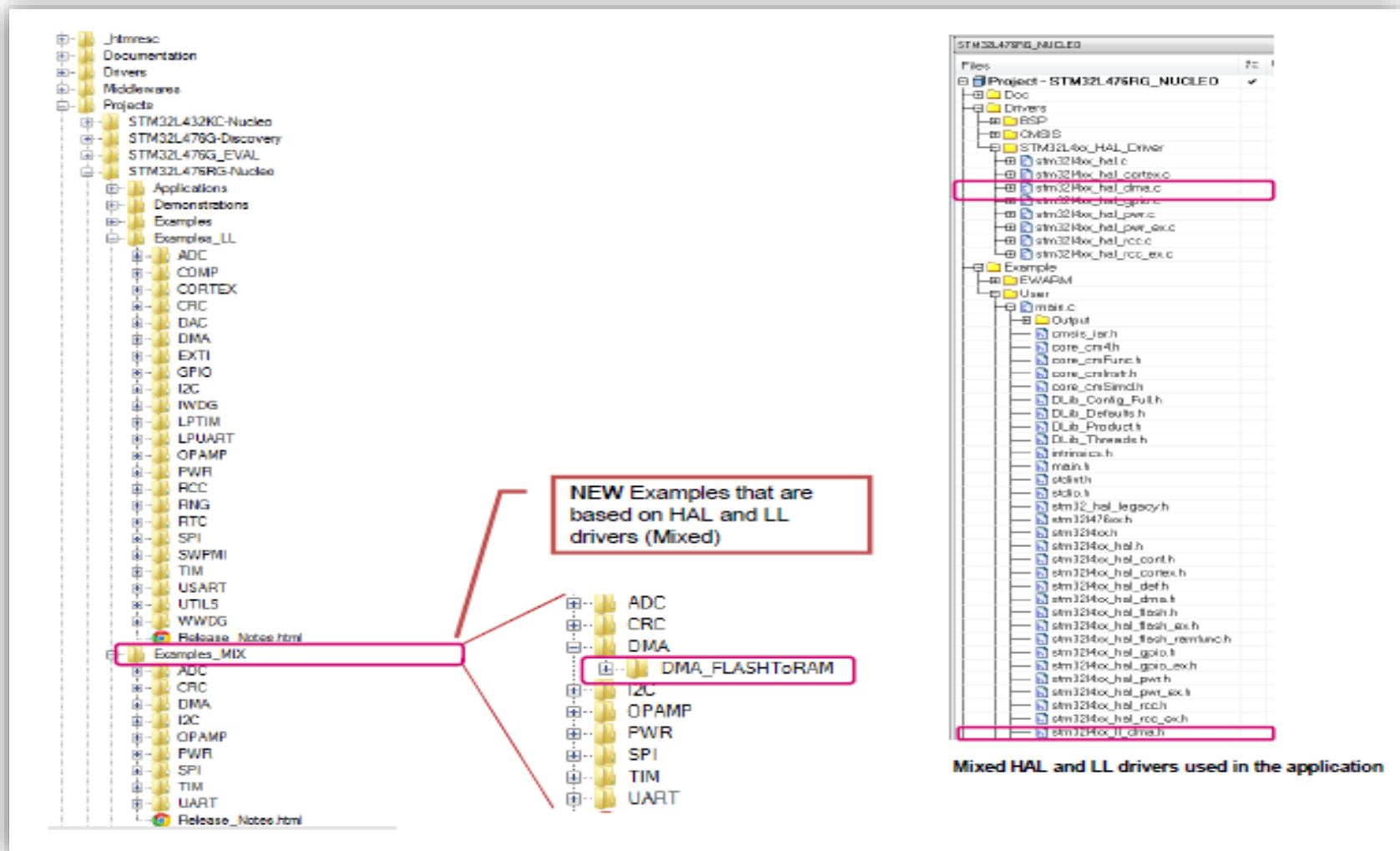
- DMA_CopyFromFlashToMemory
- DMA_CopyFromFlashToMemory_Init

EXTI

- EXTI_ToggleLedOnIT
- EXTI_ToggleLedOnIT_Init



HAL library - LL & HAL mix Example



Start new project

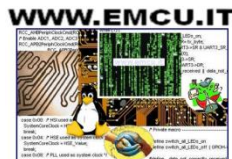
New Project



30

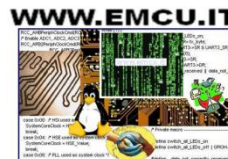
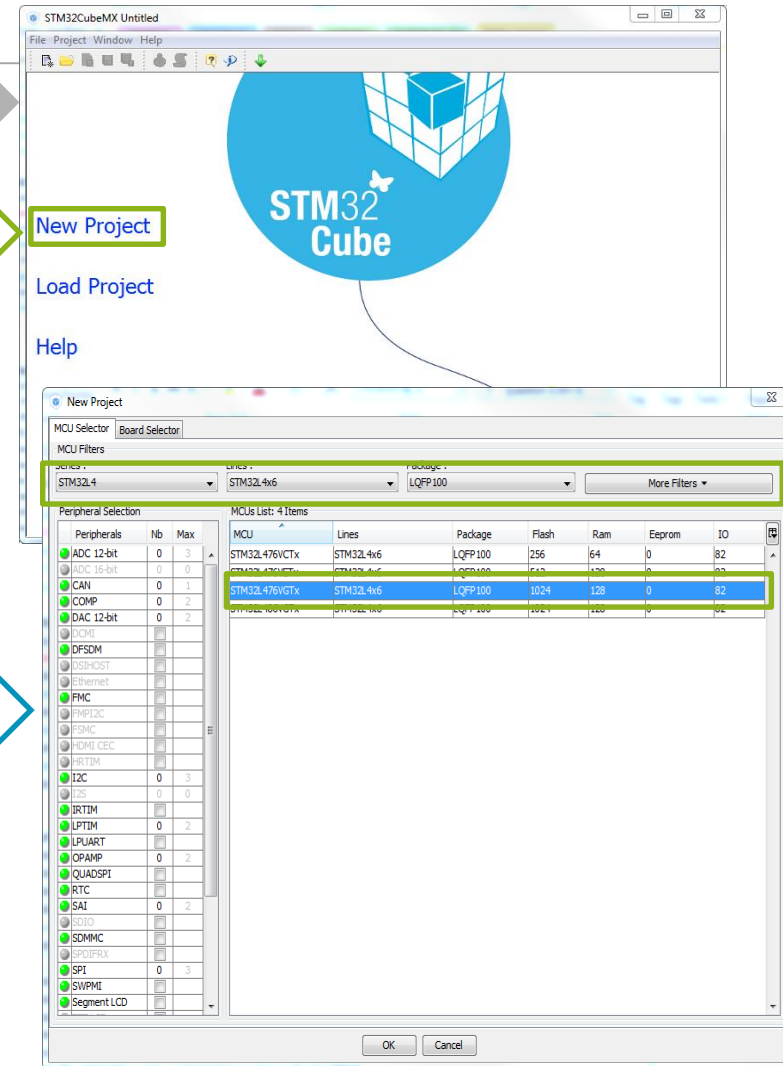


23 July 2016

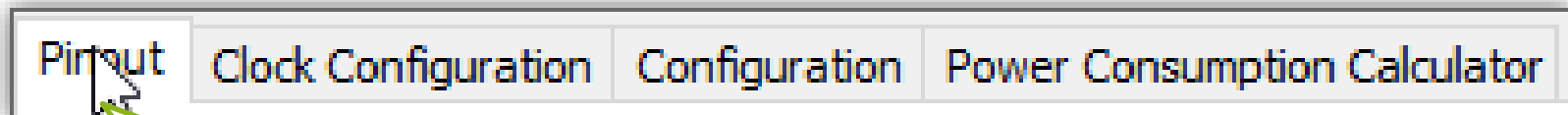


Create new project using CubeMX

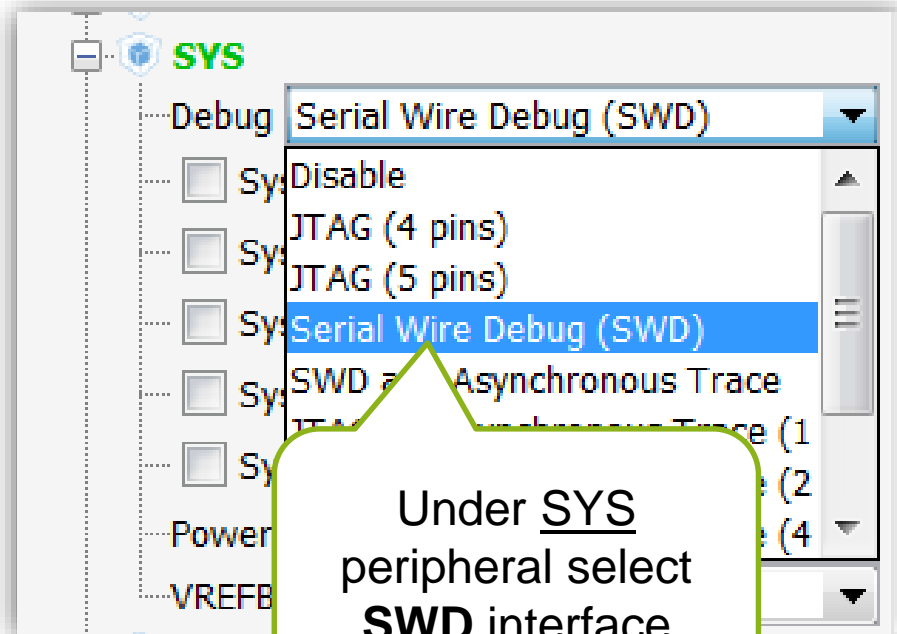
- Run CubeMX tool
- Start **new project**
 - Click “New Project” desktop shortcut, or
 - Go to “Menu->File->New Project”
- Filter:
 - Series: STM32L4
 - Line: STM32L4x6
 - Package: LQFP100
- Select: **STM32L476VGTx**



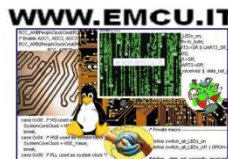
Configure debug interface 1/2



Go to Pinout settings



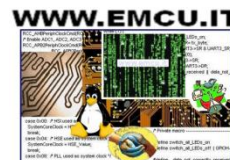
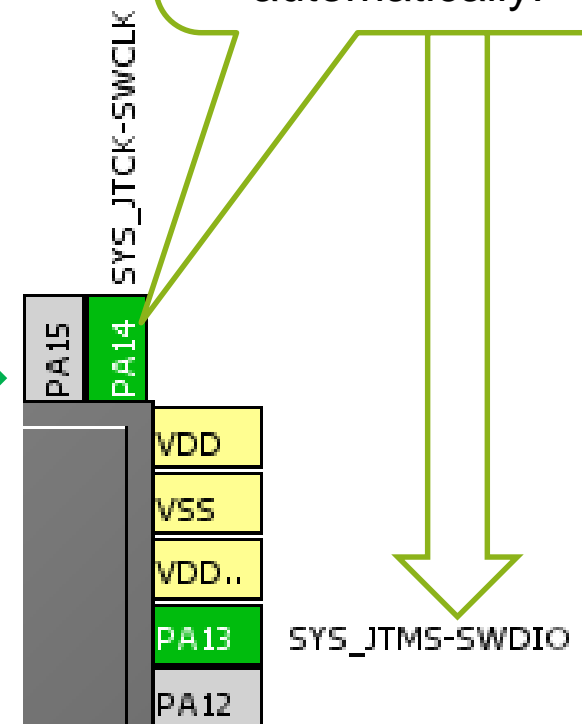
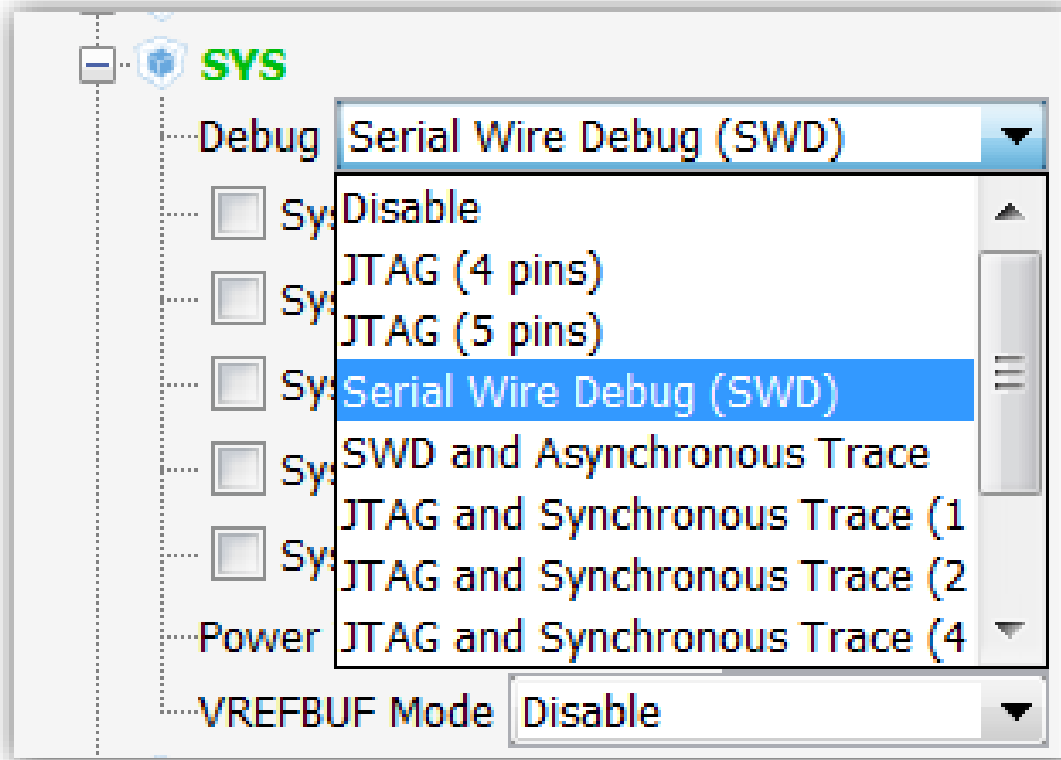
Under SYS
peripheral select
SWD interface



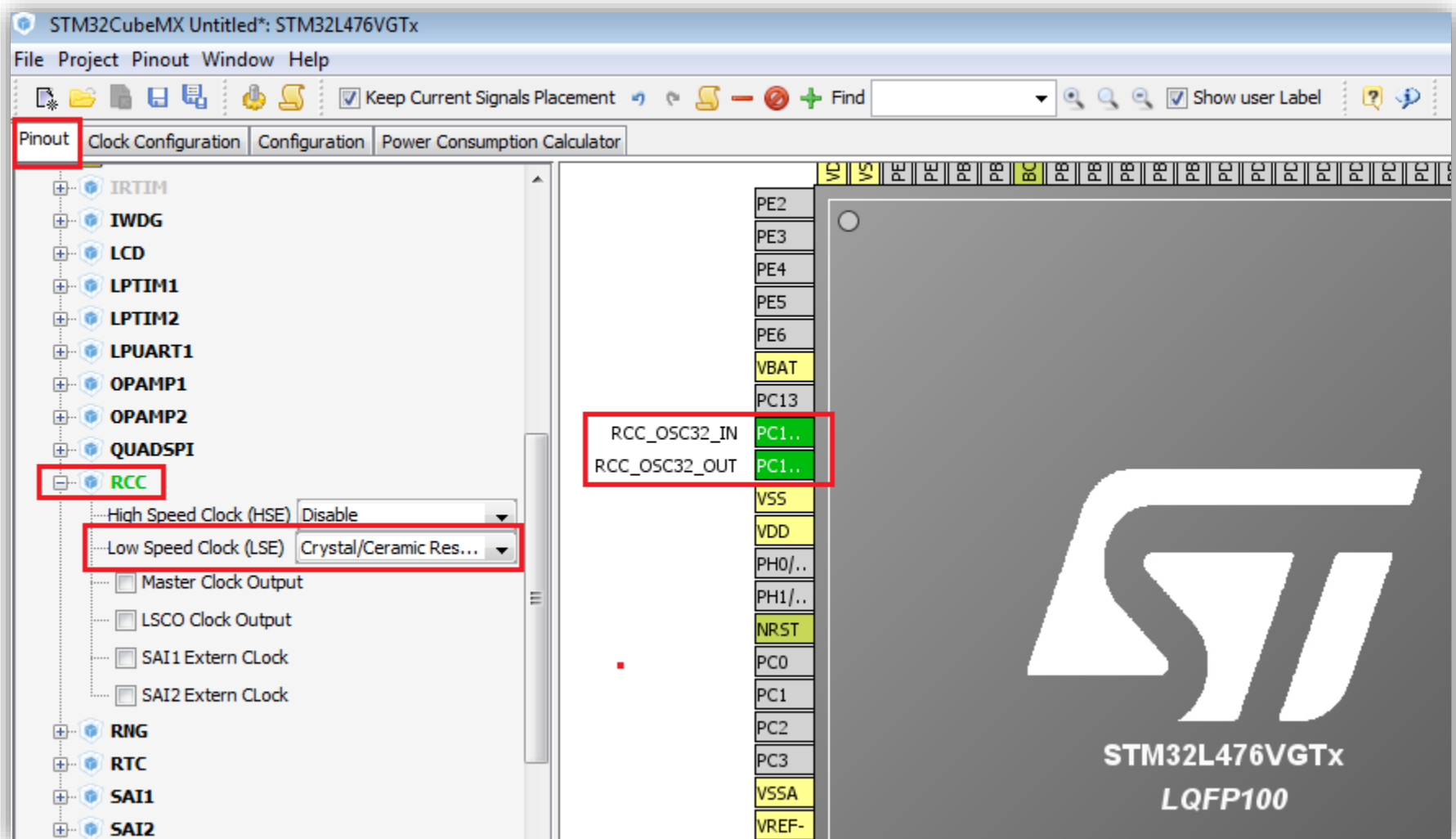
Configure debug interface 2/2



The corresponding pins are assigned and configured automatically!



Configure LSE resonator (32,768 KHz)



WWW.EMCU.IT



34

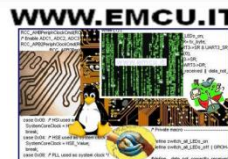
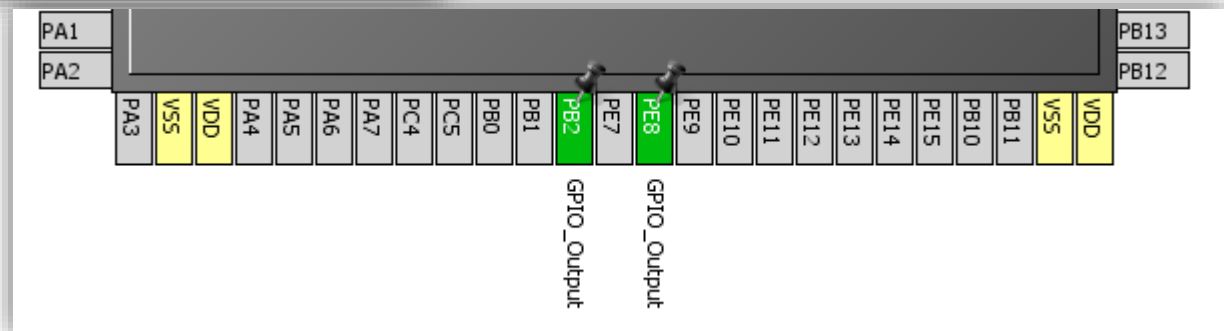
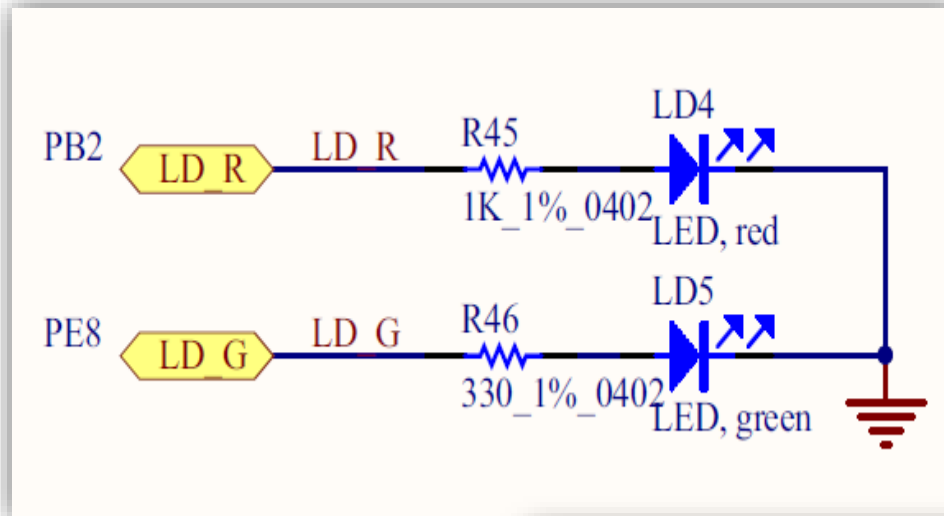


23 July 2016



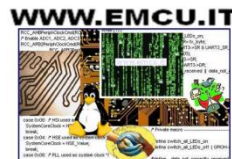
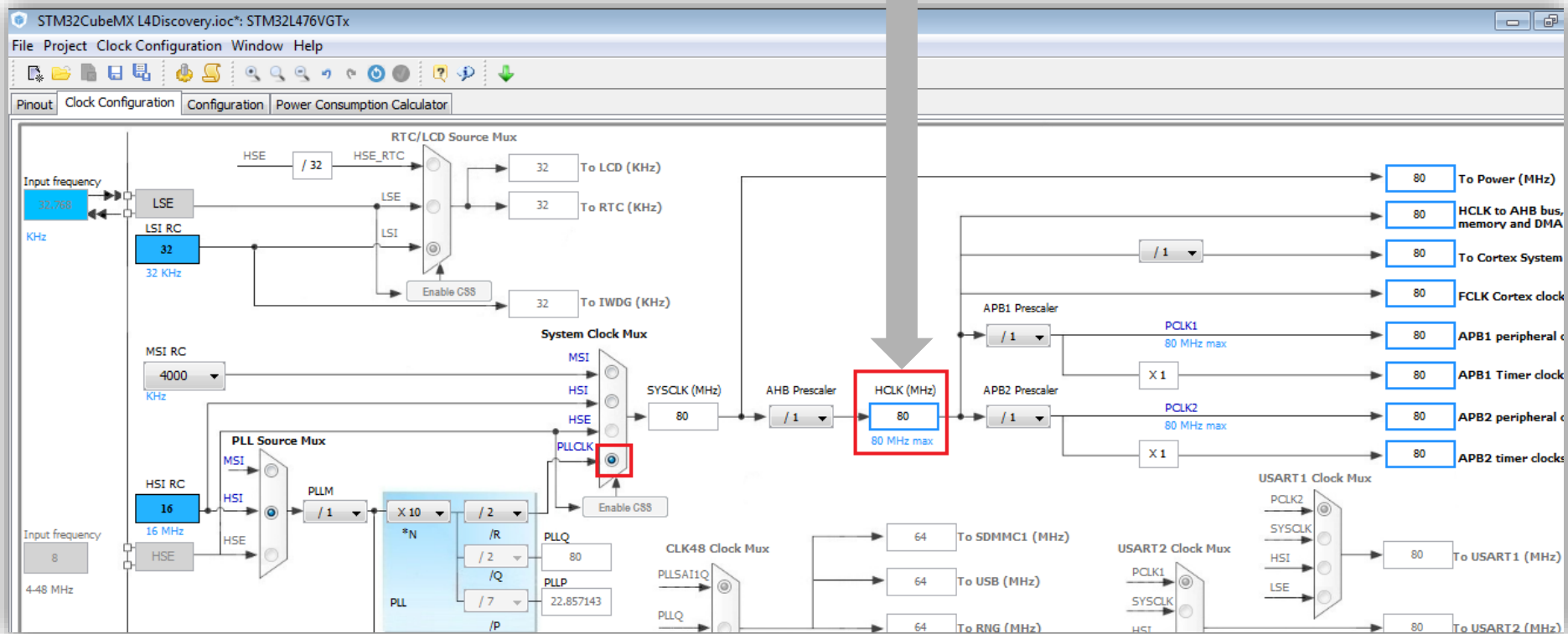
Configure GPIO for LED toggling

Configure LED pin as GPIO_Output

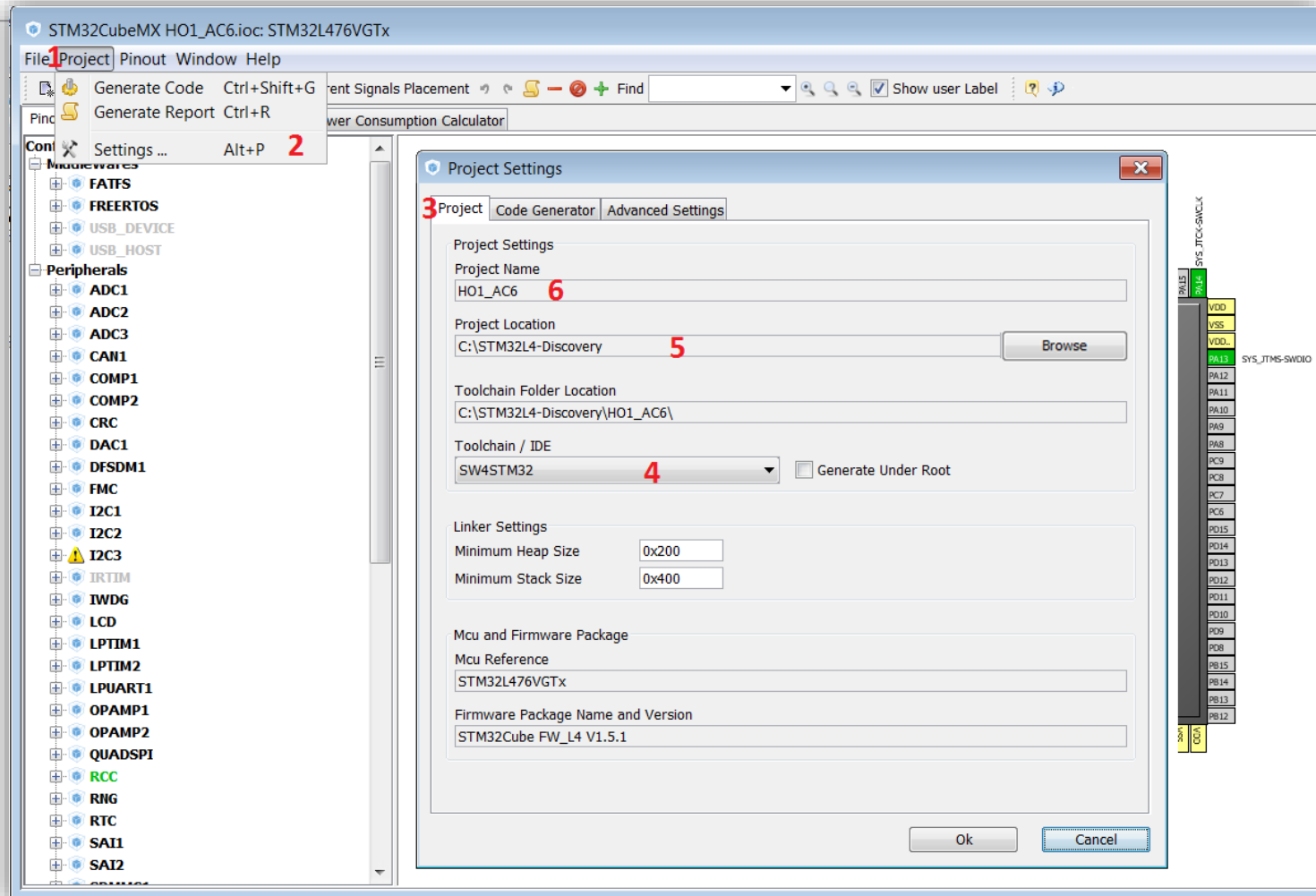


Clock configuration

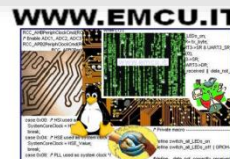
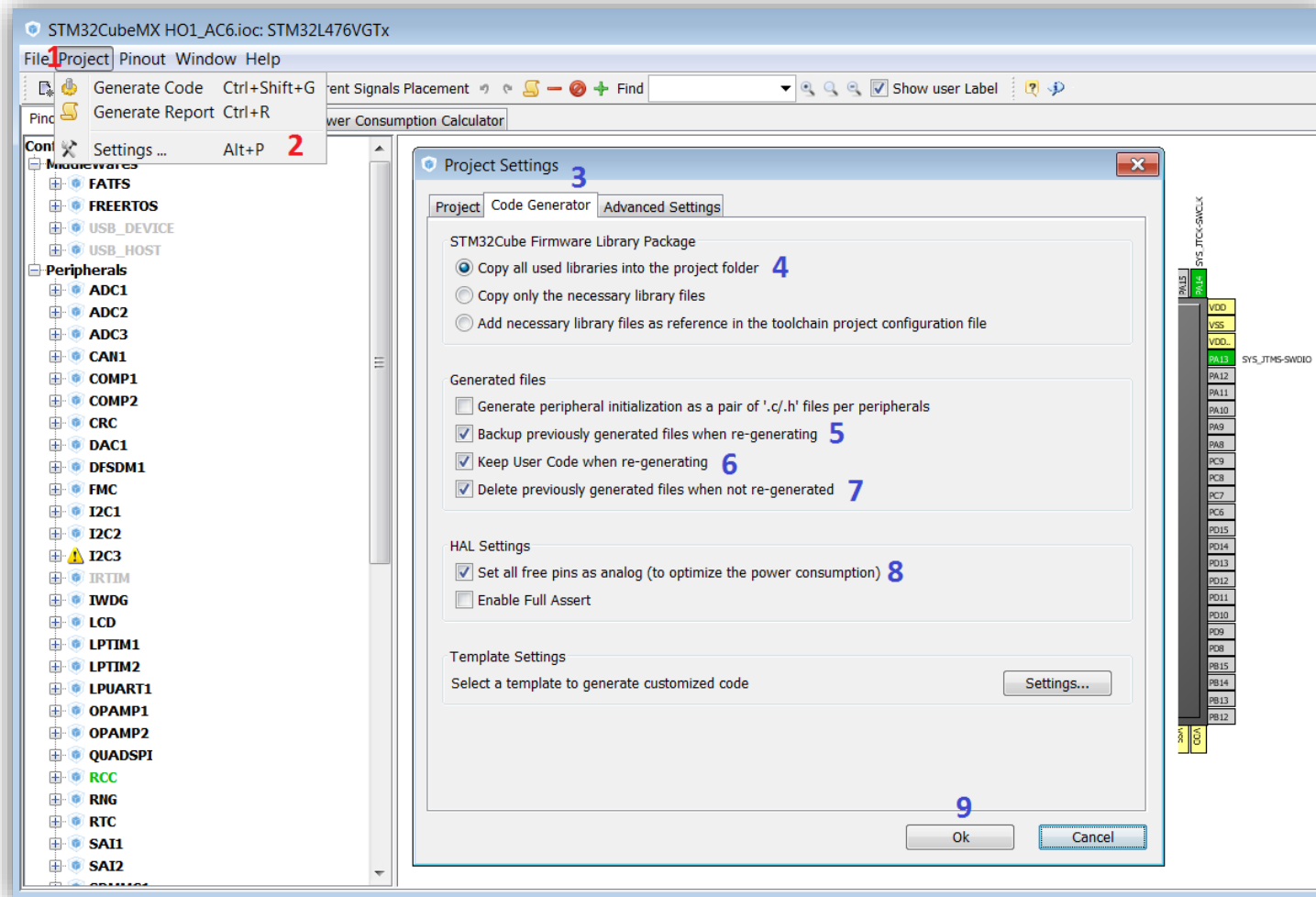
Write here: 80



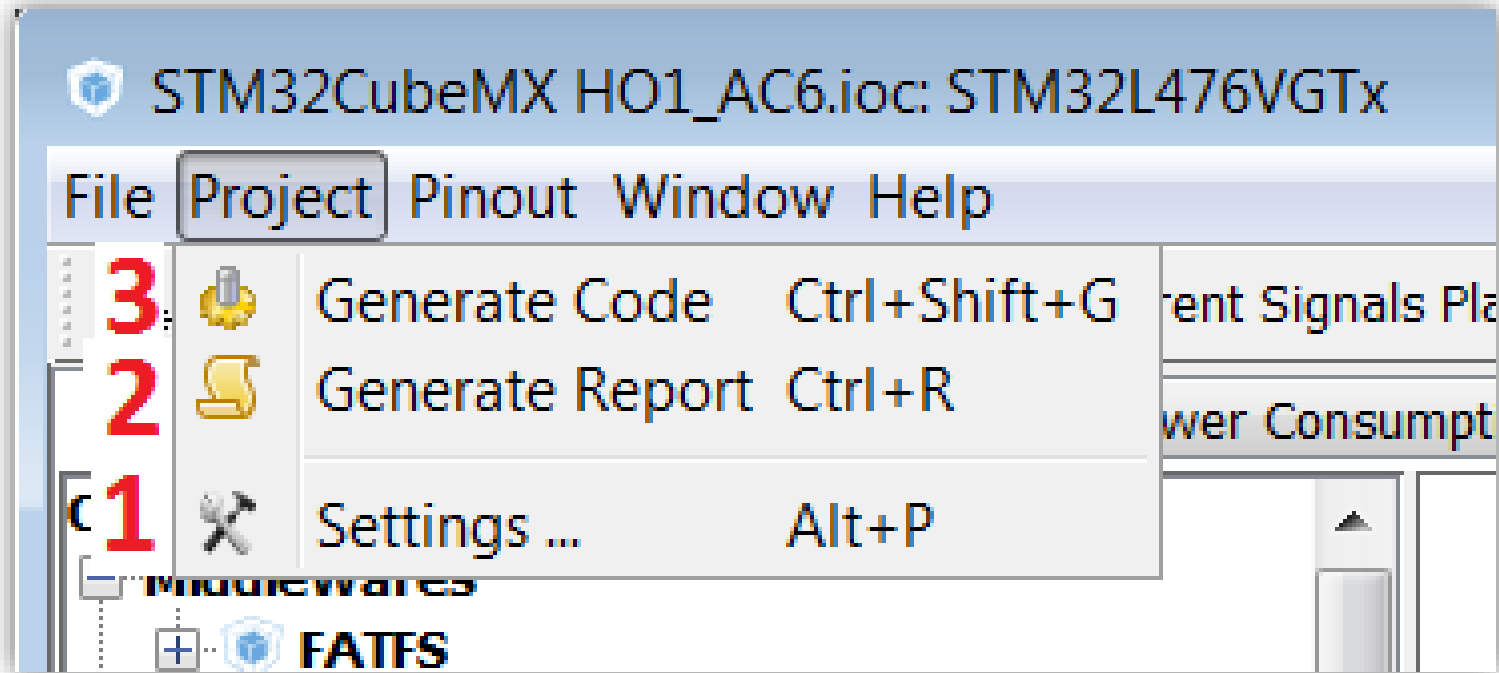
CubeMX generate the code for some GUI 1/3



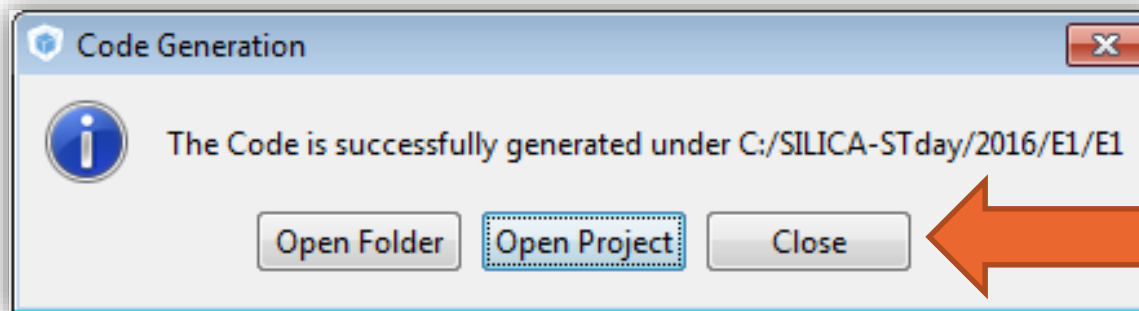
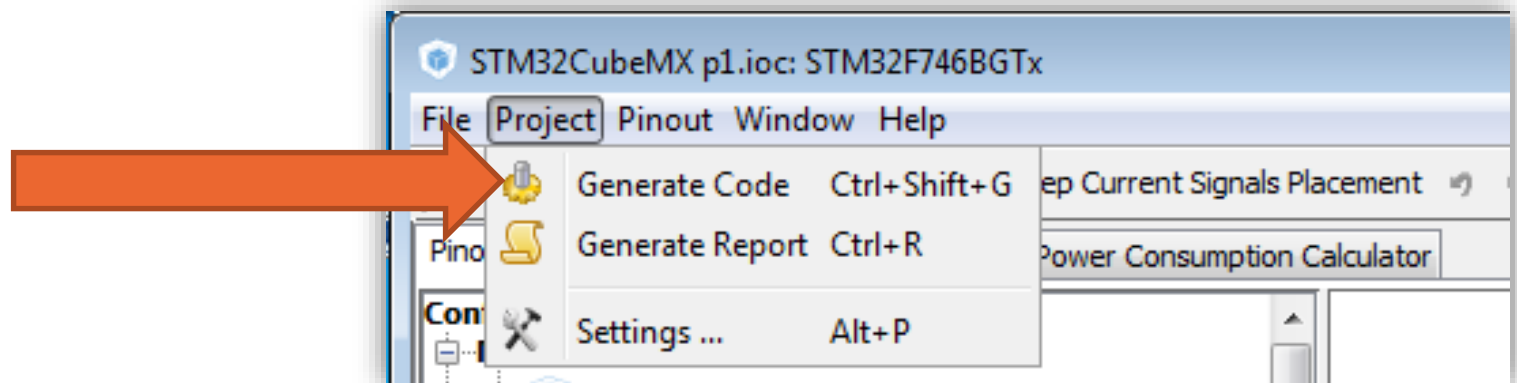
CubeMX generate the code for some GUI 2/3



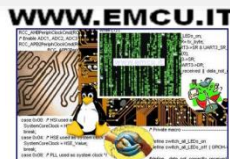
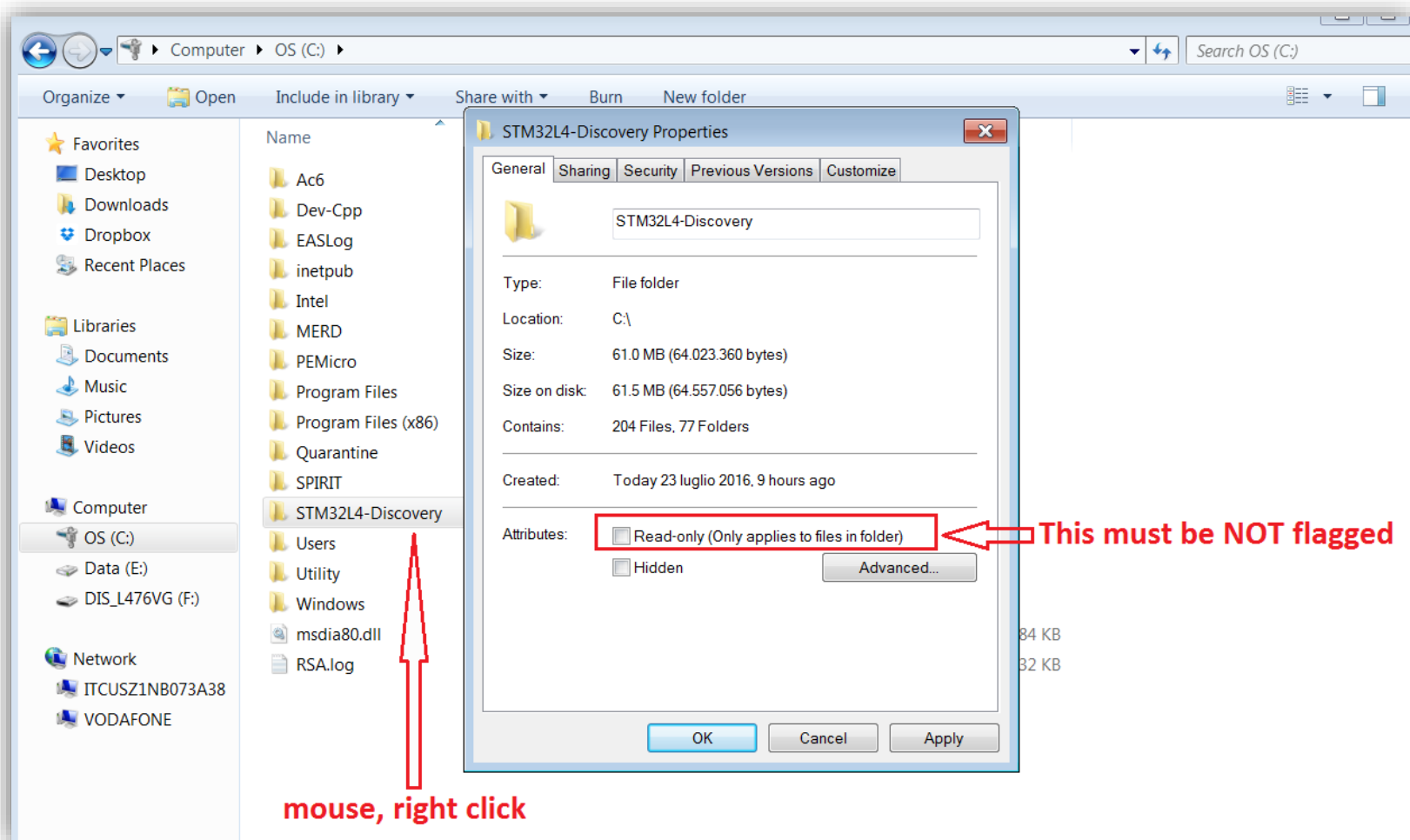
CubeMX generate the code for some GUI 3/3



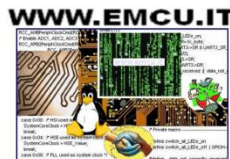
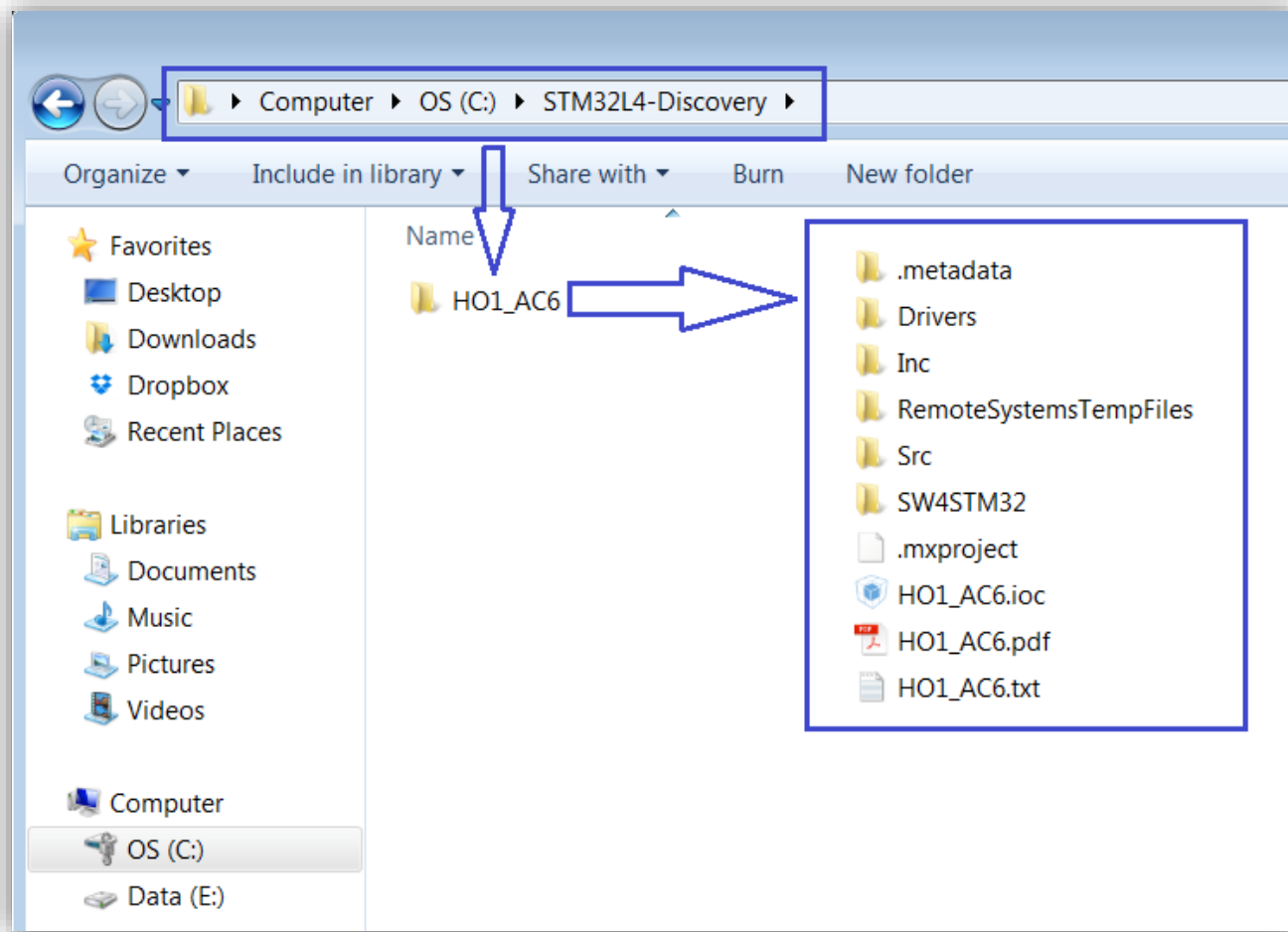
CubeMX generate the code



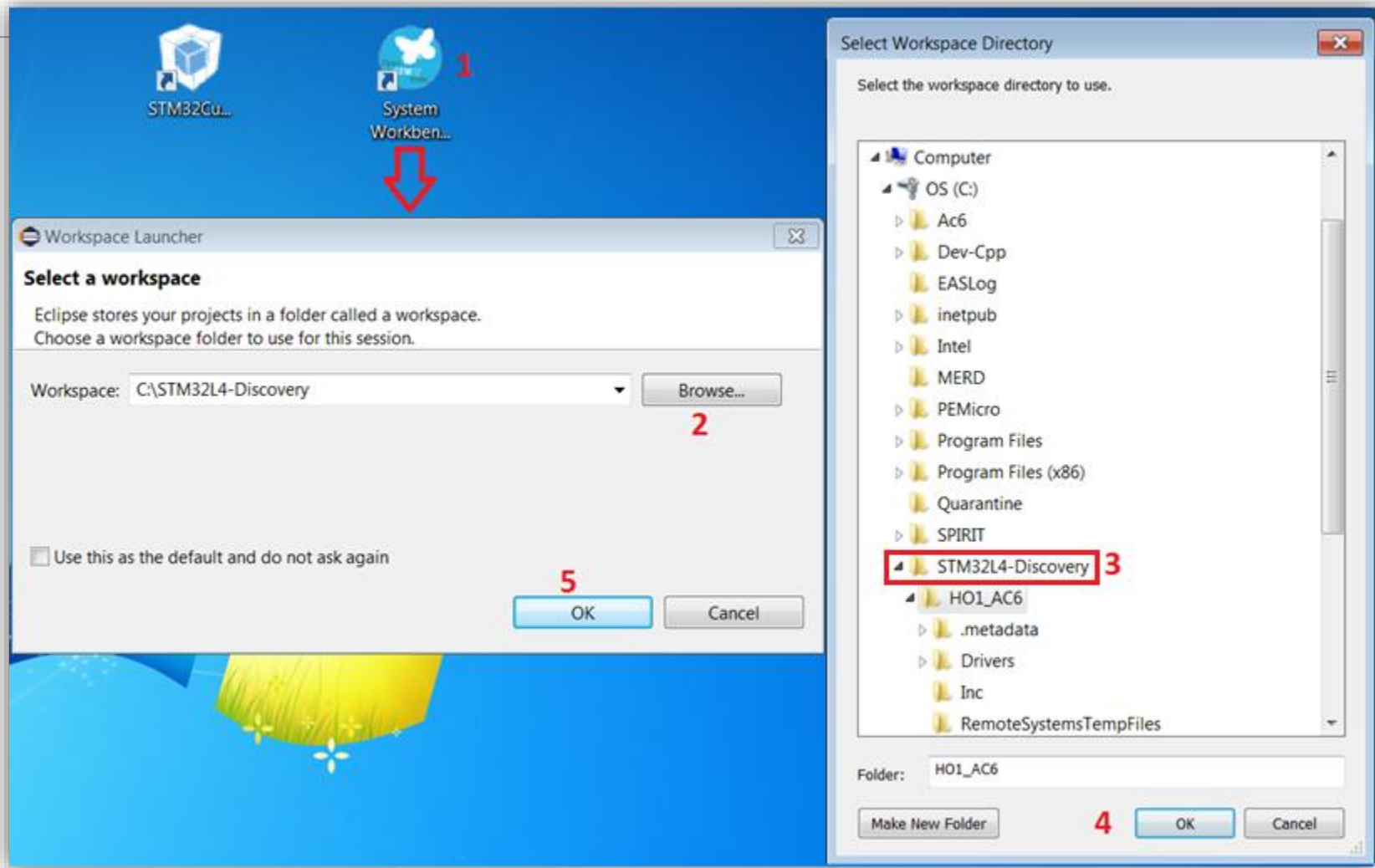
Control the proprietary of your working directory



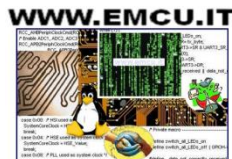
Contents of your working directory



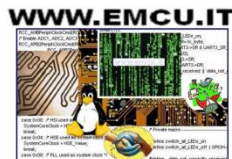
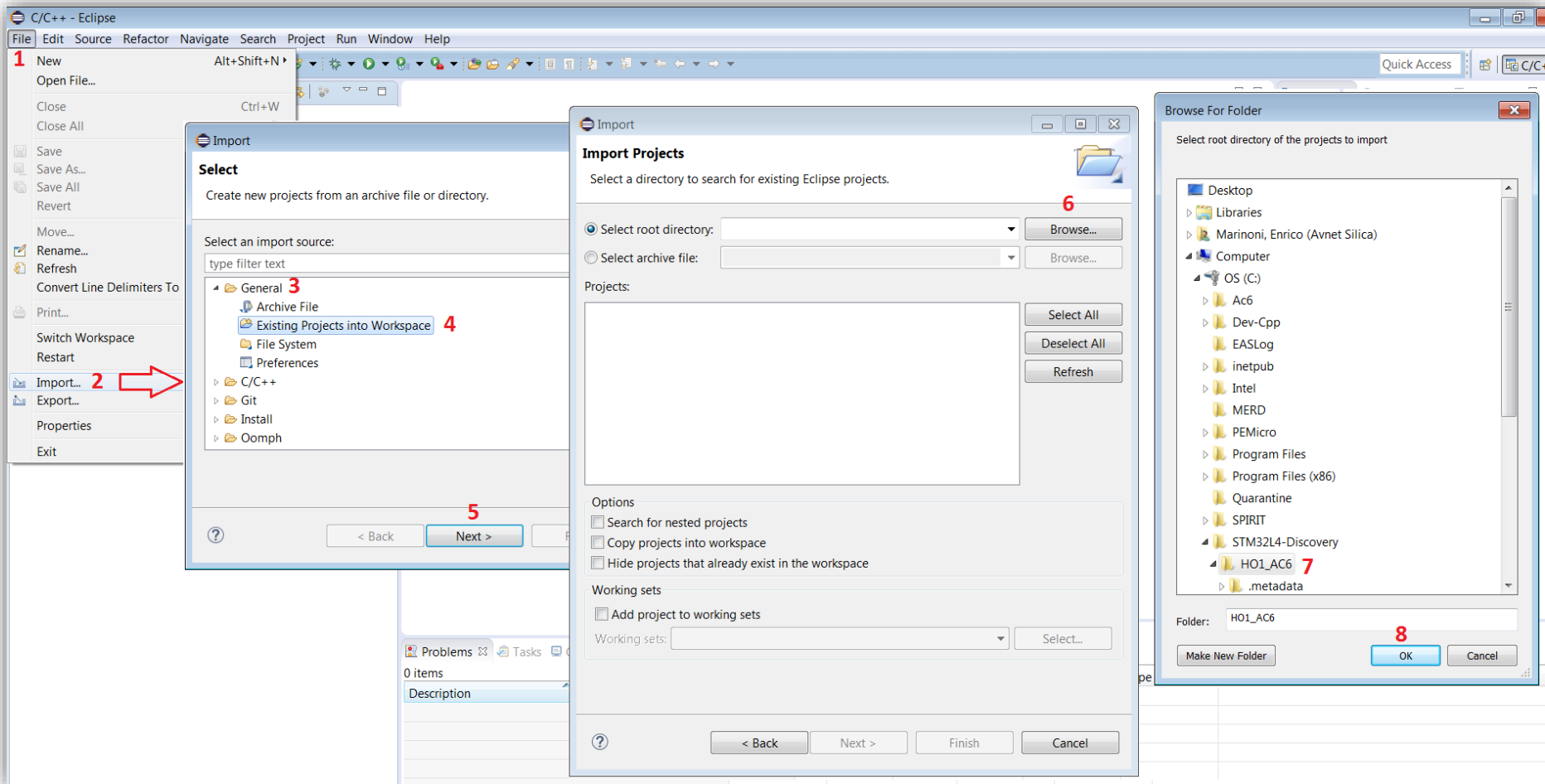
Run AC6 - (SW4STM32 - System Workbench)



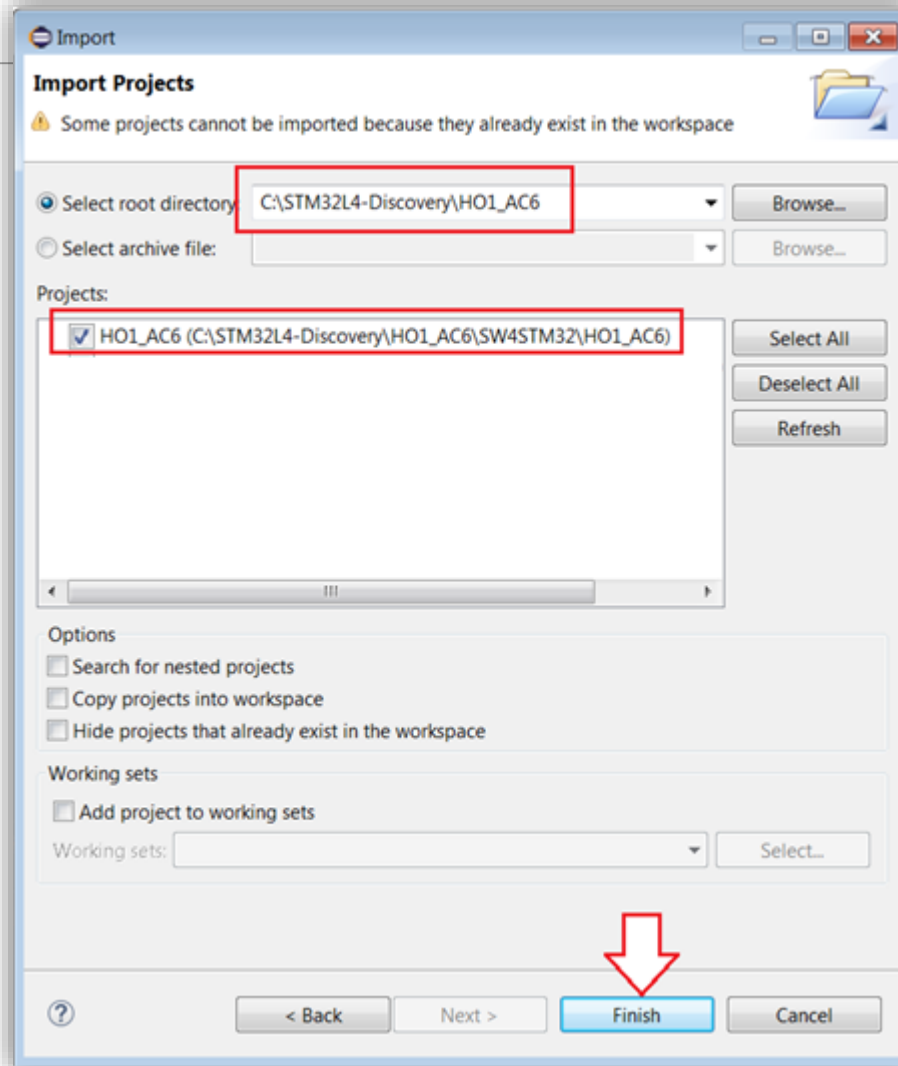
AC6 - (SW4STM32 - System Workbench)



AC6 - (SW4STM32 - System Workbench)



AC6 - (SW4STM32 - System Workbench)

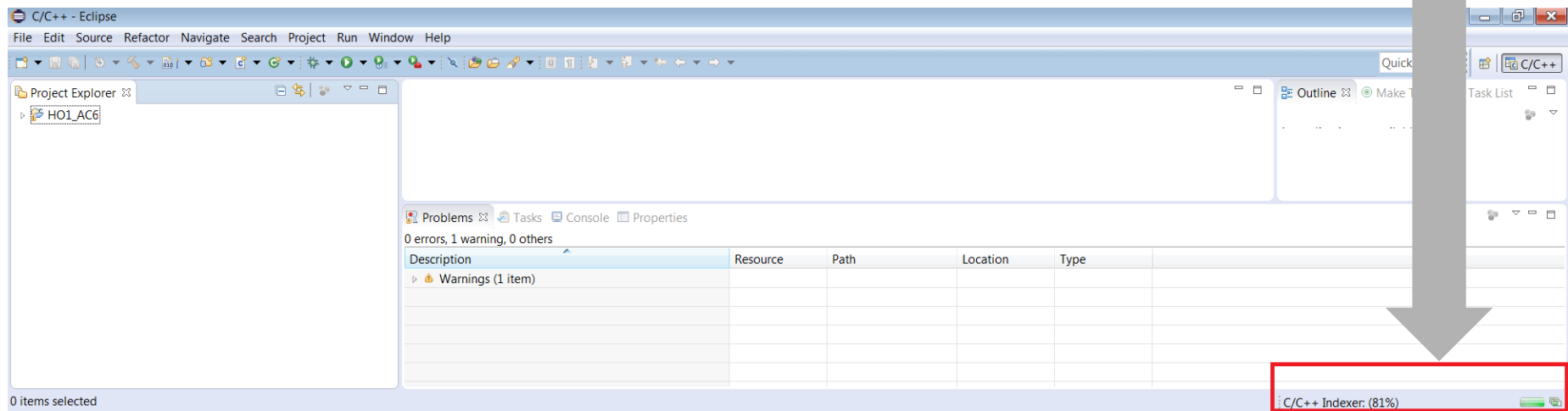


WWW.EMCU.IT

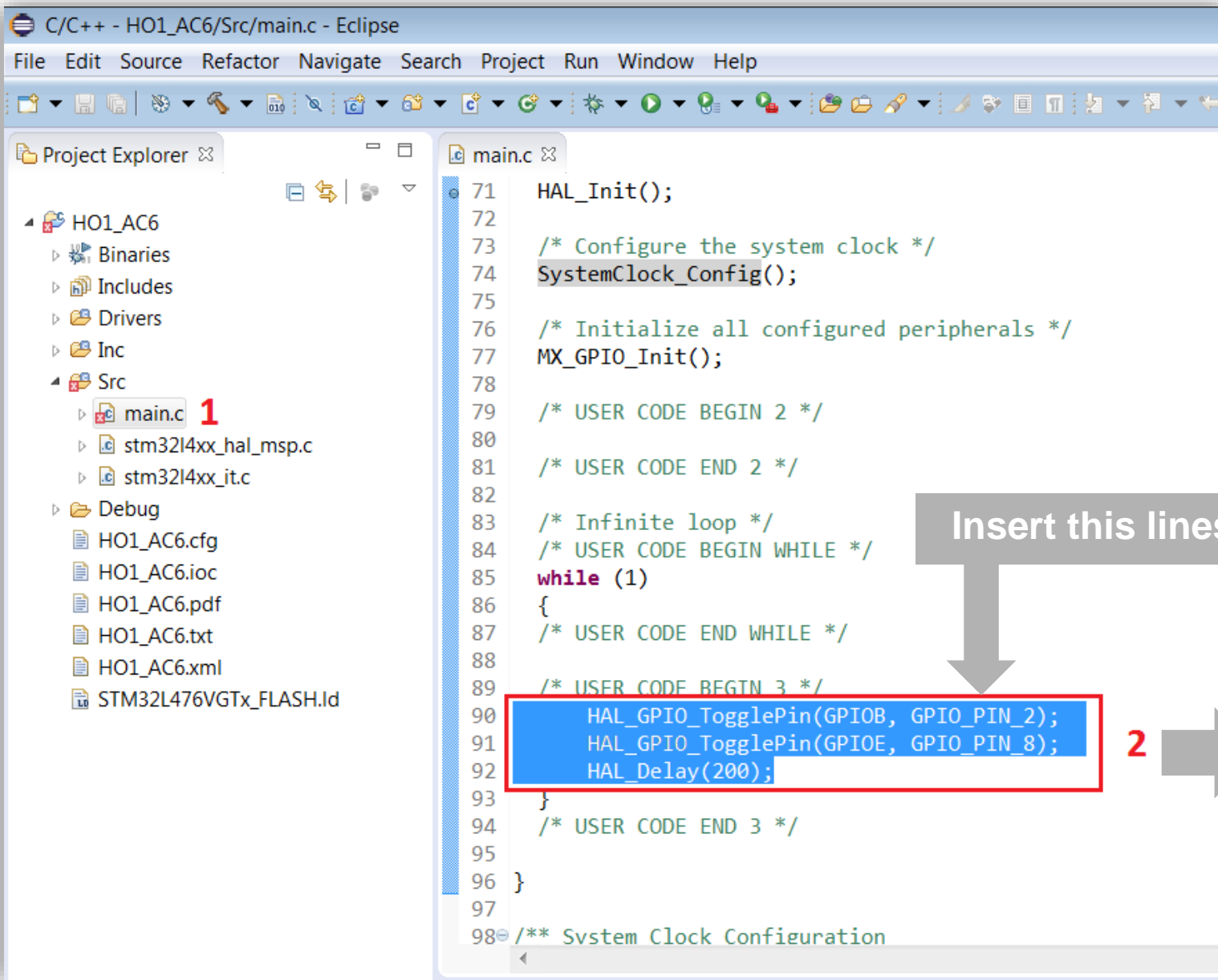


AC6 - (SW4STM32 - System Workbench)

Wait the import of the project



AC6 - (SW4STM32 - System Workbench)



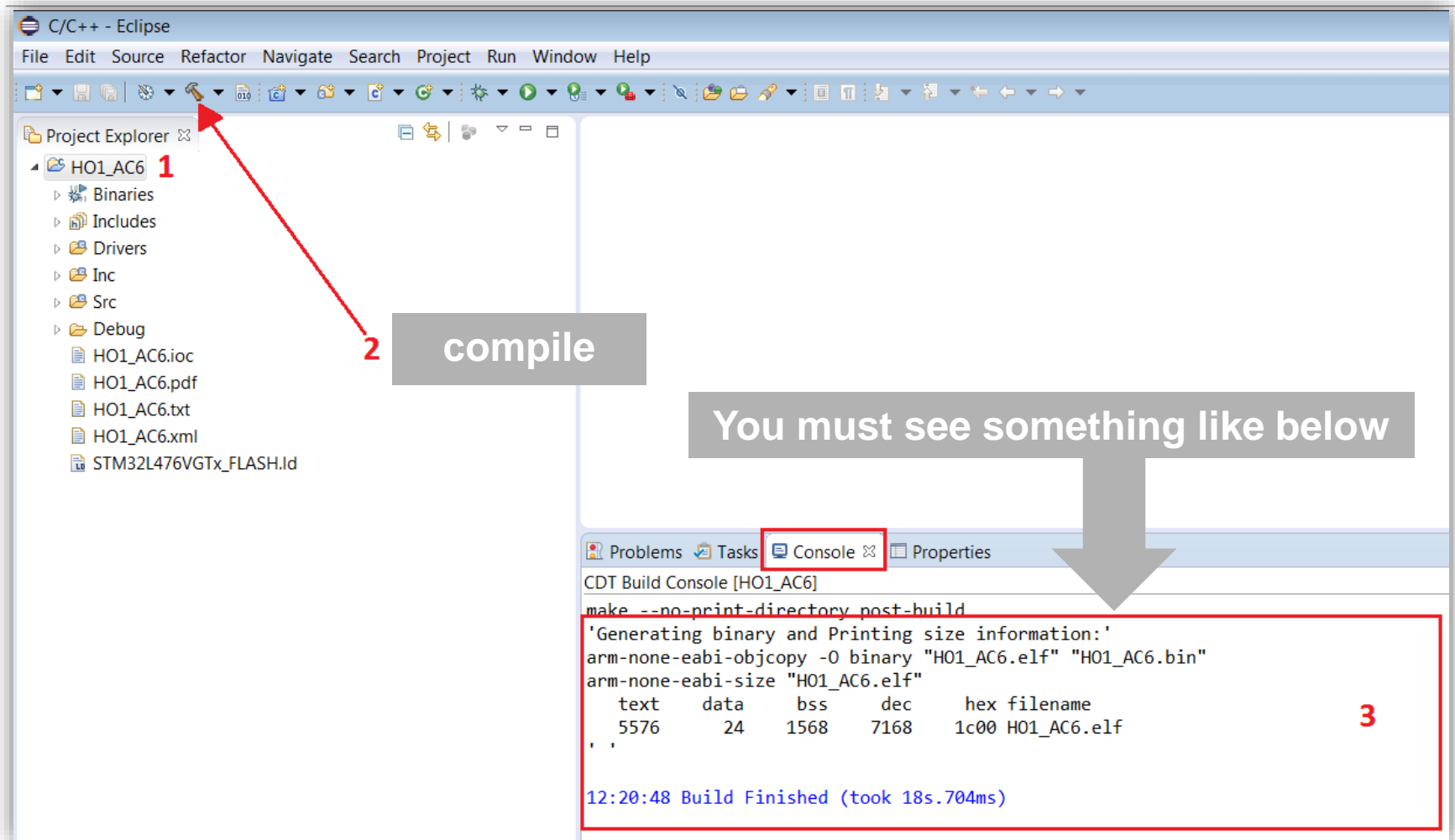
Insert this lines in main.c

See the:

UM1884

Description
of STM32L4
HAL and LL
drivers

AC6 - (SW4STM32 - System Workbench)



1 HO1_AC6

- Binaries
- Includes
- Drivers
- Inc
- Src
- Debug
 - HO1_AC6.ioc
 - HO1_AC6.pdf
 - HO1_AC6.txt
 - HO1_AC6.xml
 - STM32L476VGTx_FLASH.ld

2 compile

You must see something like below

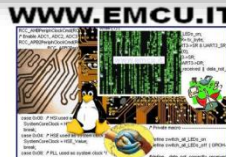
Problems Tasks Console Properties

CDT Build Console [HO1_AC6]

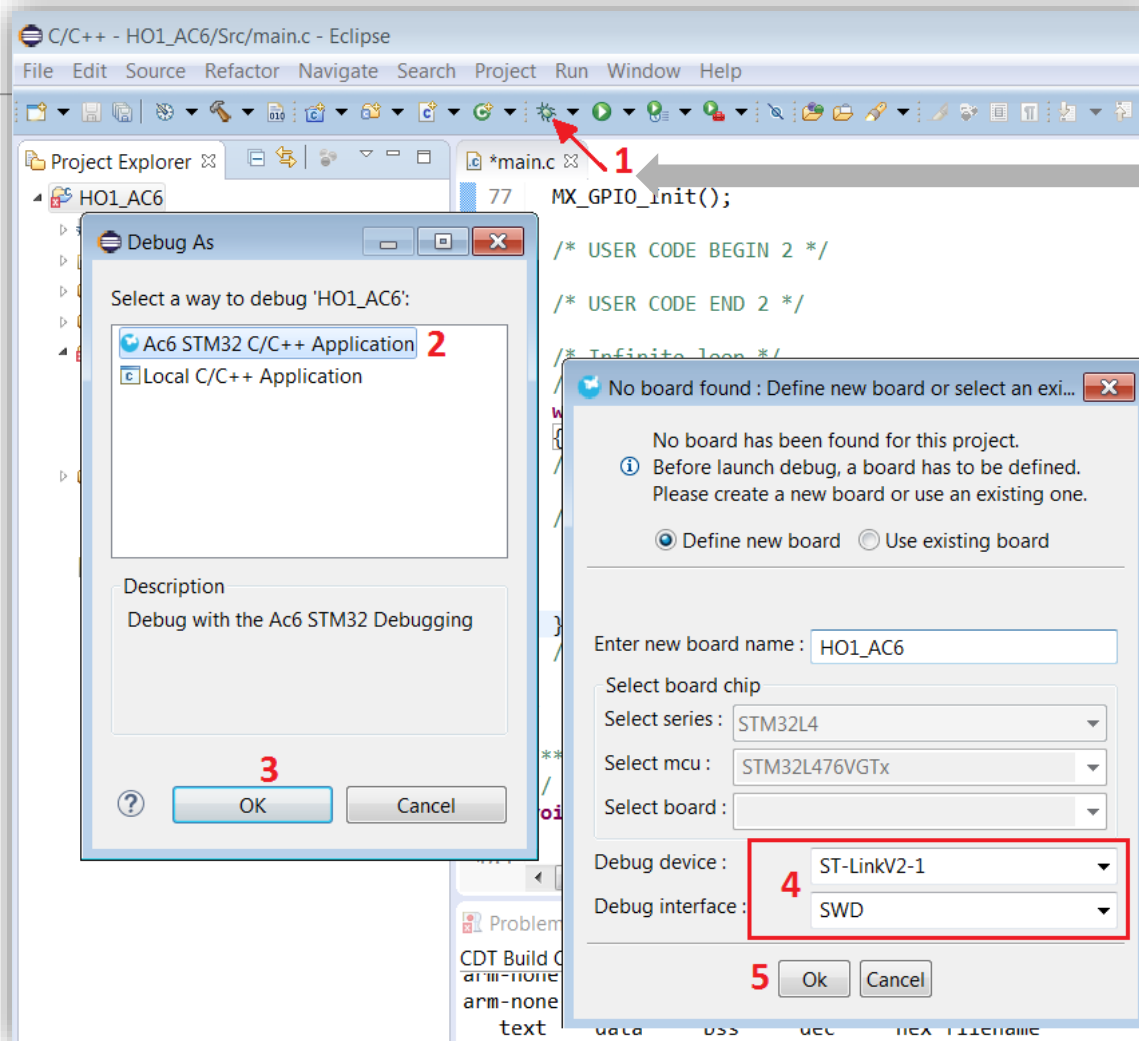
```
make --no-print-directory post-build
'Generating binary and Printing size information:'
arm-none-eabi-objcopy -O binary "HO1_AC6.elf" "HO1_AC6.bin"
arm-none-eabi-size "HO1_AC6.elf"
  text    data    bss     dec     hex filename
  5576     24   1568    7168    1c00 HO1_AC6.elf
  , ,
```

3

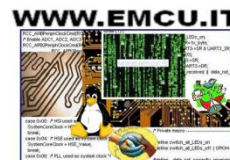
12:20:48 Build Finished (took 18s.704ms)



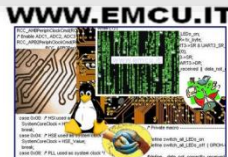
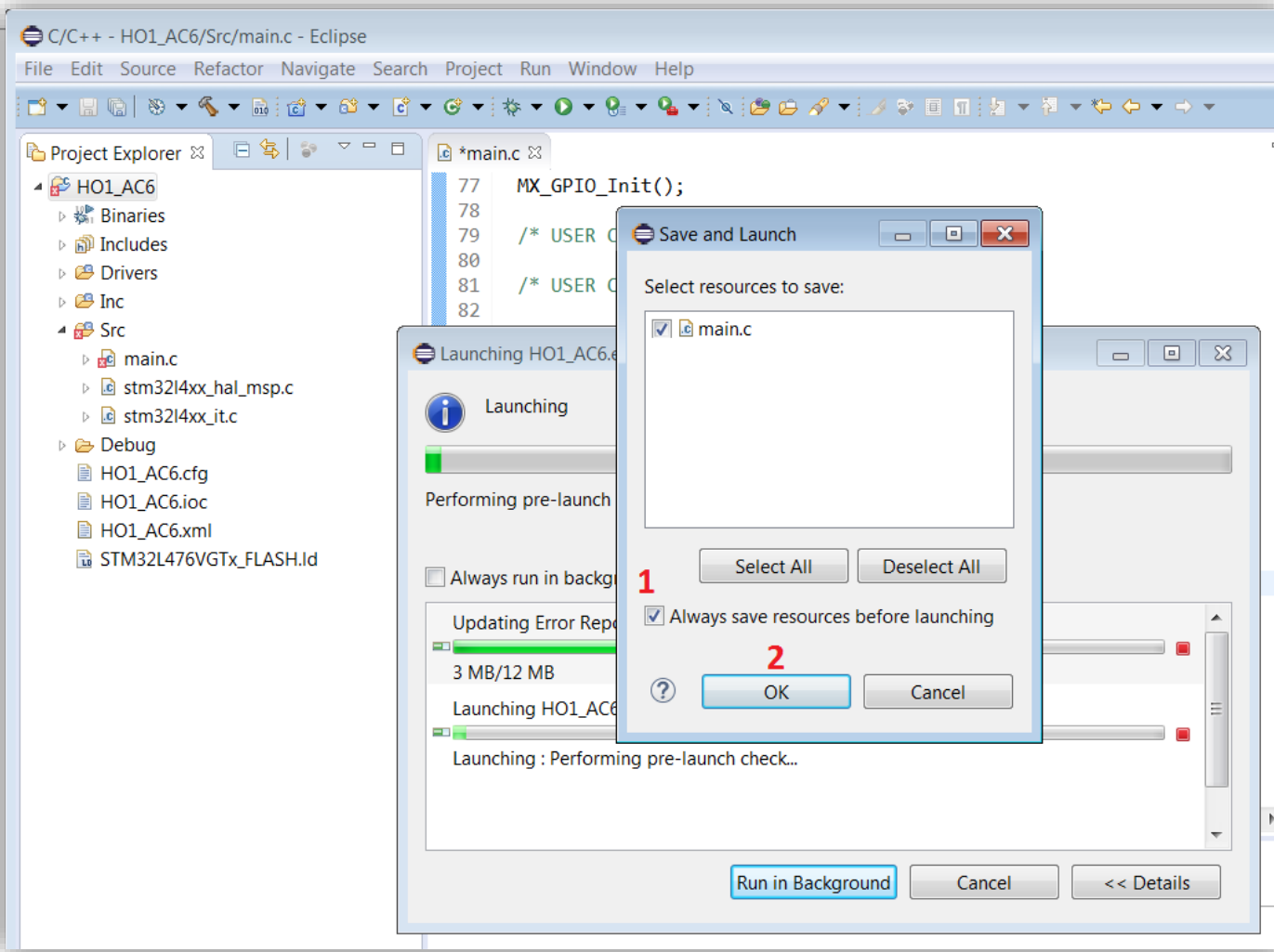
AC6 - (SW4STM32 - System Workbench)



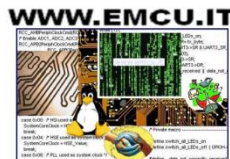
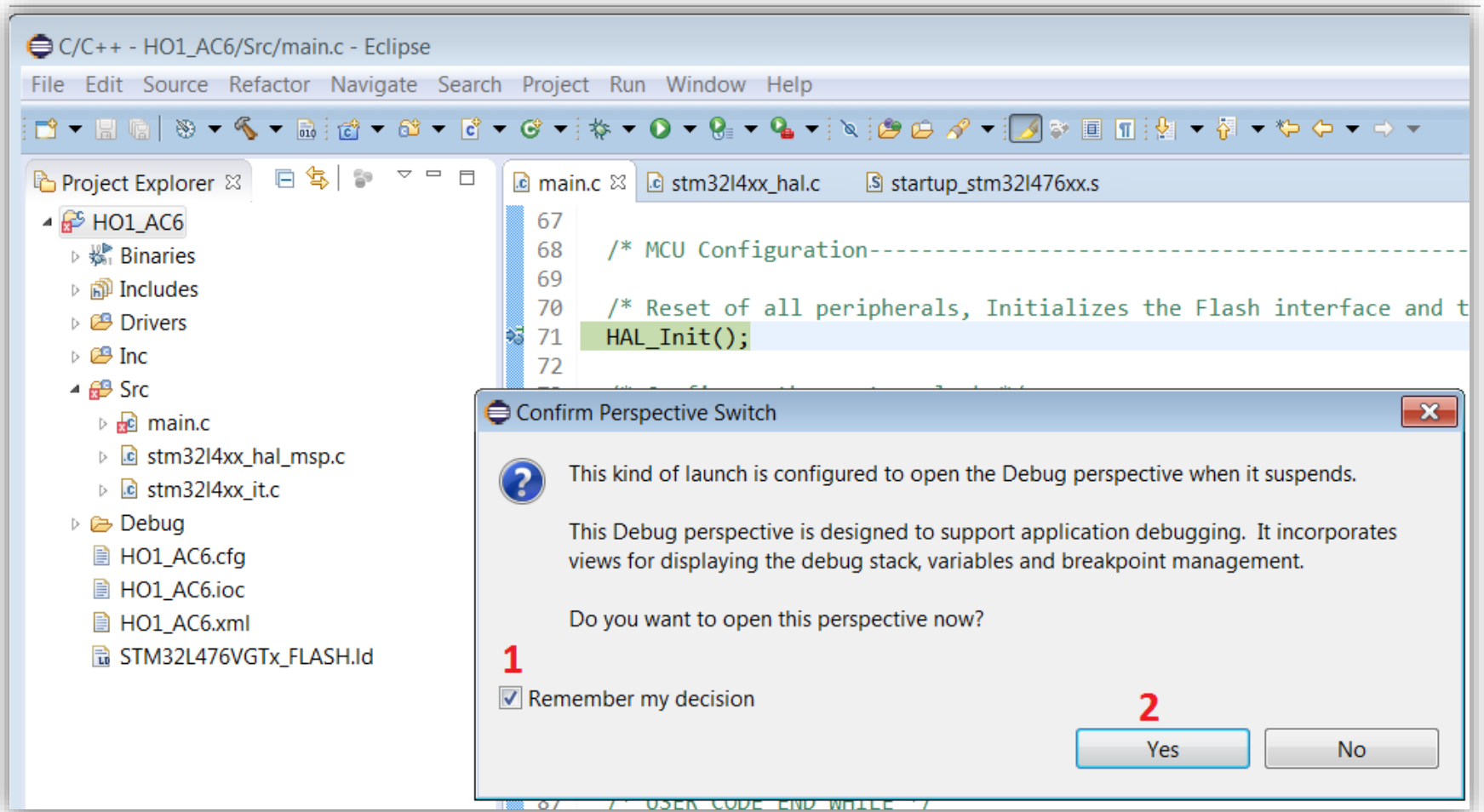
Enter in debug



AC6 - (SW4STM32 - System Workbench)



AC6 - (SW4STM32 - System Workbench)



AC6 - (SW4STM32 - System Workbench)

Now you are in debug

Click here and you must see the LEDs flashing

Debug - HO1_AC6/Src/main.c - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access C/C++ Debug

Debug

- HO1_AC6.elf [Ac6 STM32 Debugging]
- HO1_AC6.elf
 - Thread #1 (Suspended : Breakpoint)
 - main() at main.c:71 0x8001

Variables

Name	Type	Value

Breakpoints Registers I/O Registers Modules

main.c stm32l4xx_hal.c startup_stm32l476xx.s

```
67
68 /* MCU Configuration-----*/
69
70 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
71 HAL_Init();
72
73 /* Configure the system clock */
74 SystemClock_Config();
75
76 /* Initialize all configured peripherals */
77 MX_GPIO_Init();
78
79 /* USER CODE BEGIN 2 */
```

Outline

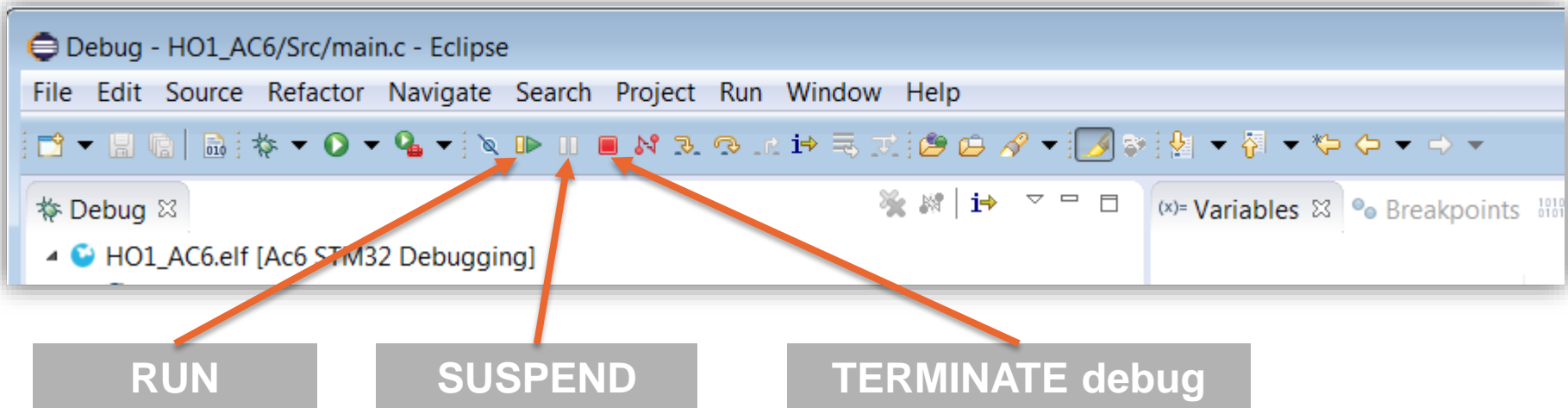
- stm32l4xx_hal.h
 - SystemClock_Config(void) : void
 - Error_Handler(void) : void
 - MX_GPIO_Init(void) : void
 - main(void) : int
 - SystemClock_Config(void) : void
 - MX_GPIO_Init(void) : void
 - Error_Handler(void) : void
 - assert_failed(uint8_t*, uint32_t) : void

Console

HO1_AC6.elf [Ac6 STM32 Debugging] C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829/tools/compiler/bin/arm-none-eabi-gdb (7.10.1.20151217) Temporary breakpoint 1, main () at ../Src/main.c:71 71 HAL_Init(); No breakpoint number 2.

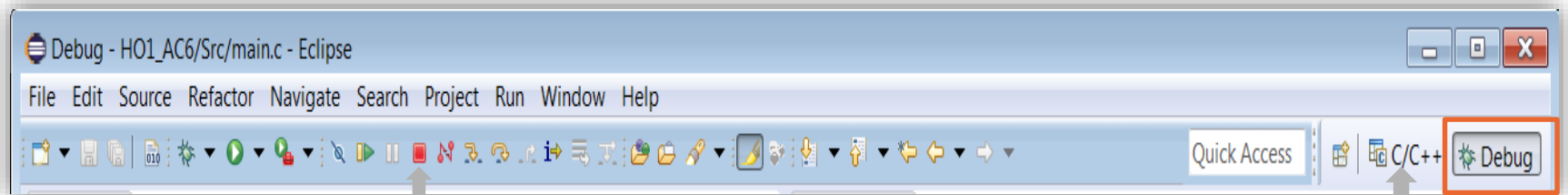
Writable Smart Insert 71 : 1

AC6 - (SW4STM32 - System Workbench)

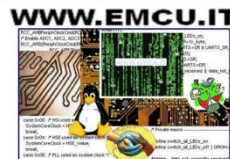
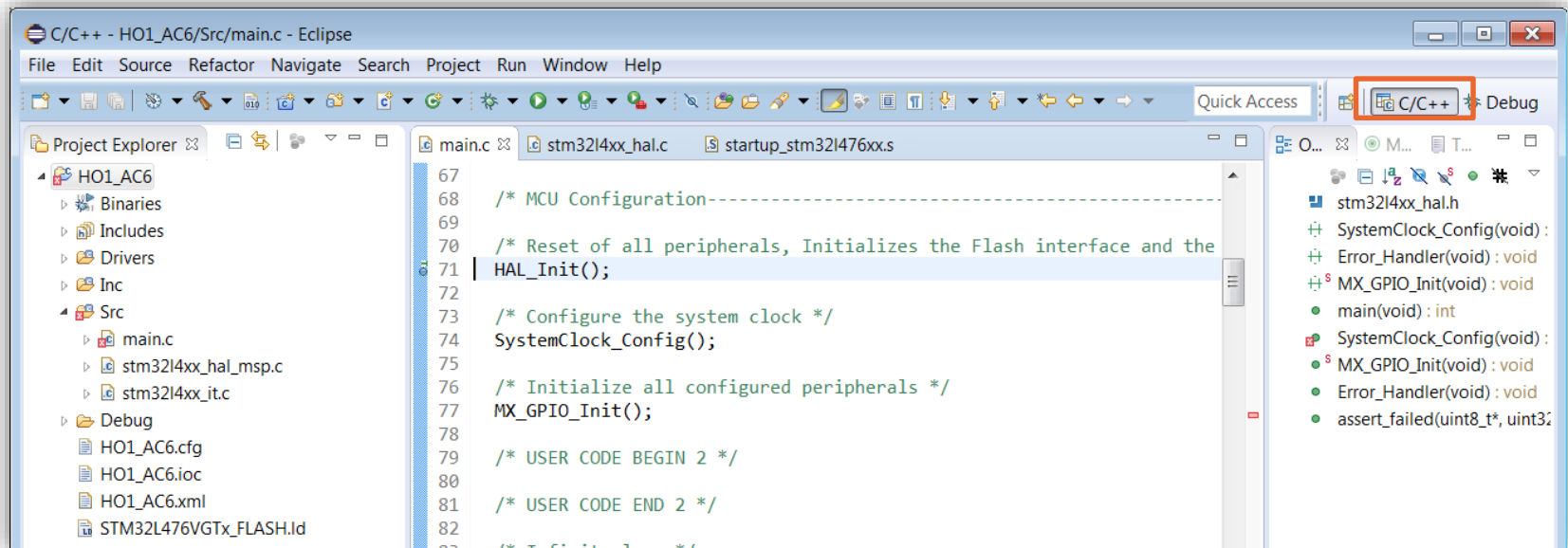


For more information regarding **AC6** syntax, tutorial, etc, see [here](#)

AC6 - (SW4STM32 - System Workbench)



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





Thank you.

WWW.EMCU.IT



56



23 July 2016



life.augmented

