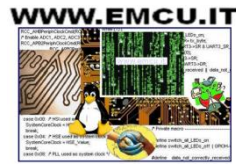


Free SW tools for STM32



Free SW tools for STM32

2

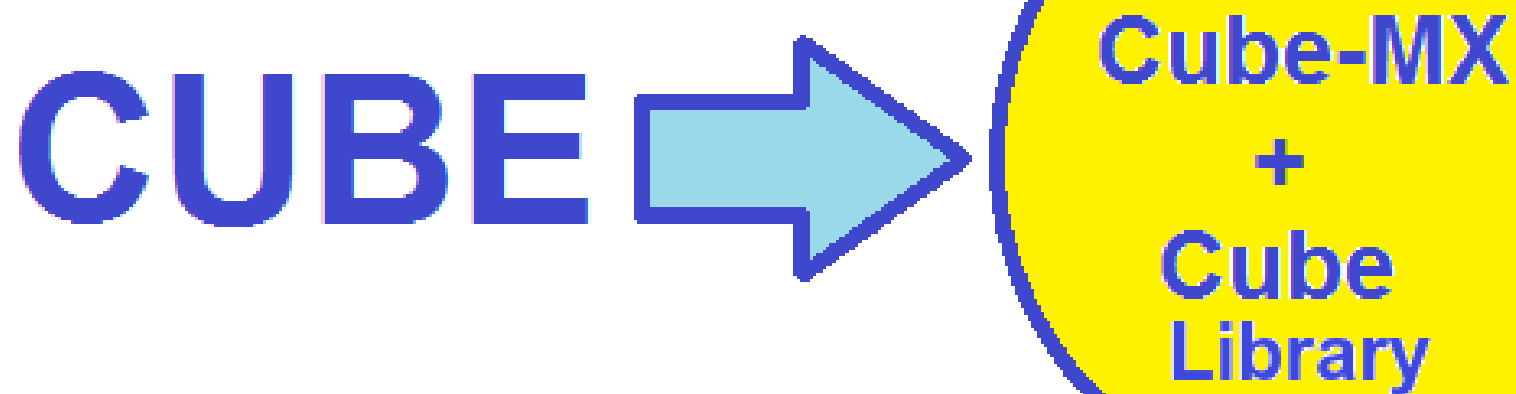


TUTORIAL



Free for STM32L0 and F0





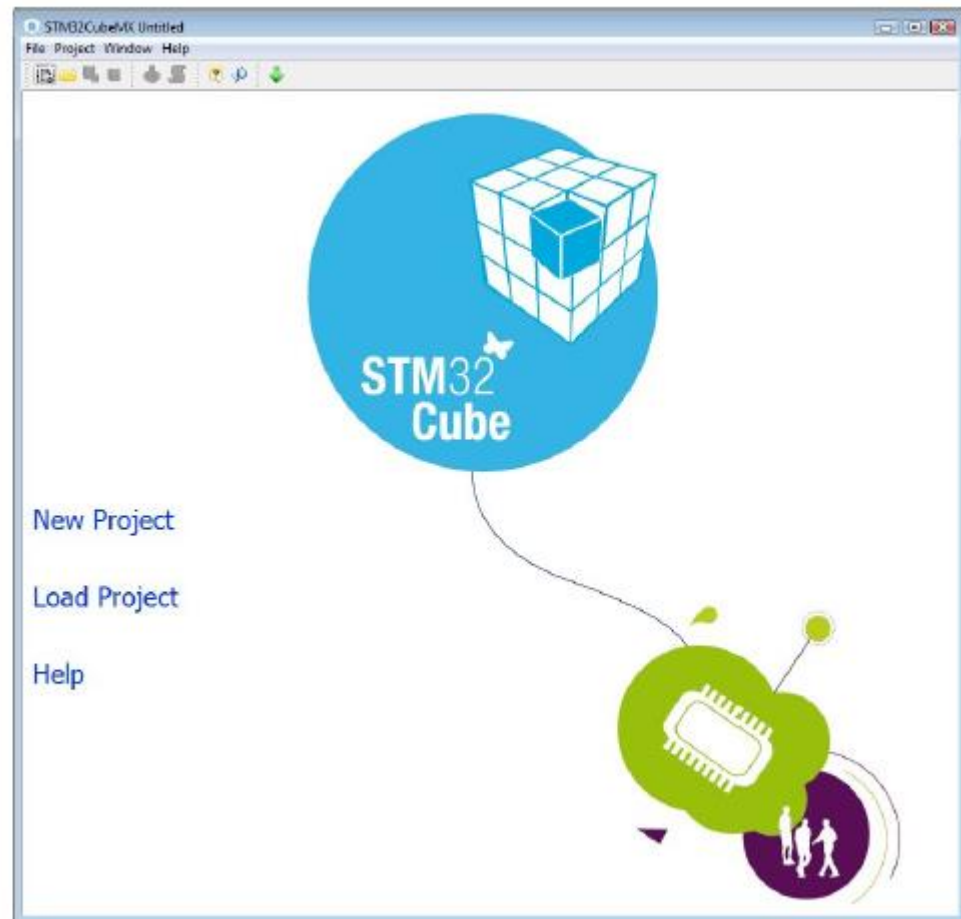
See [here](#)



CUBE-MX

4

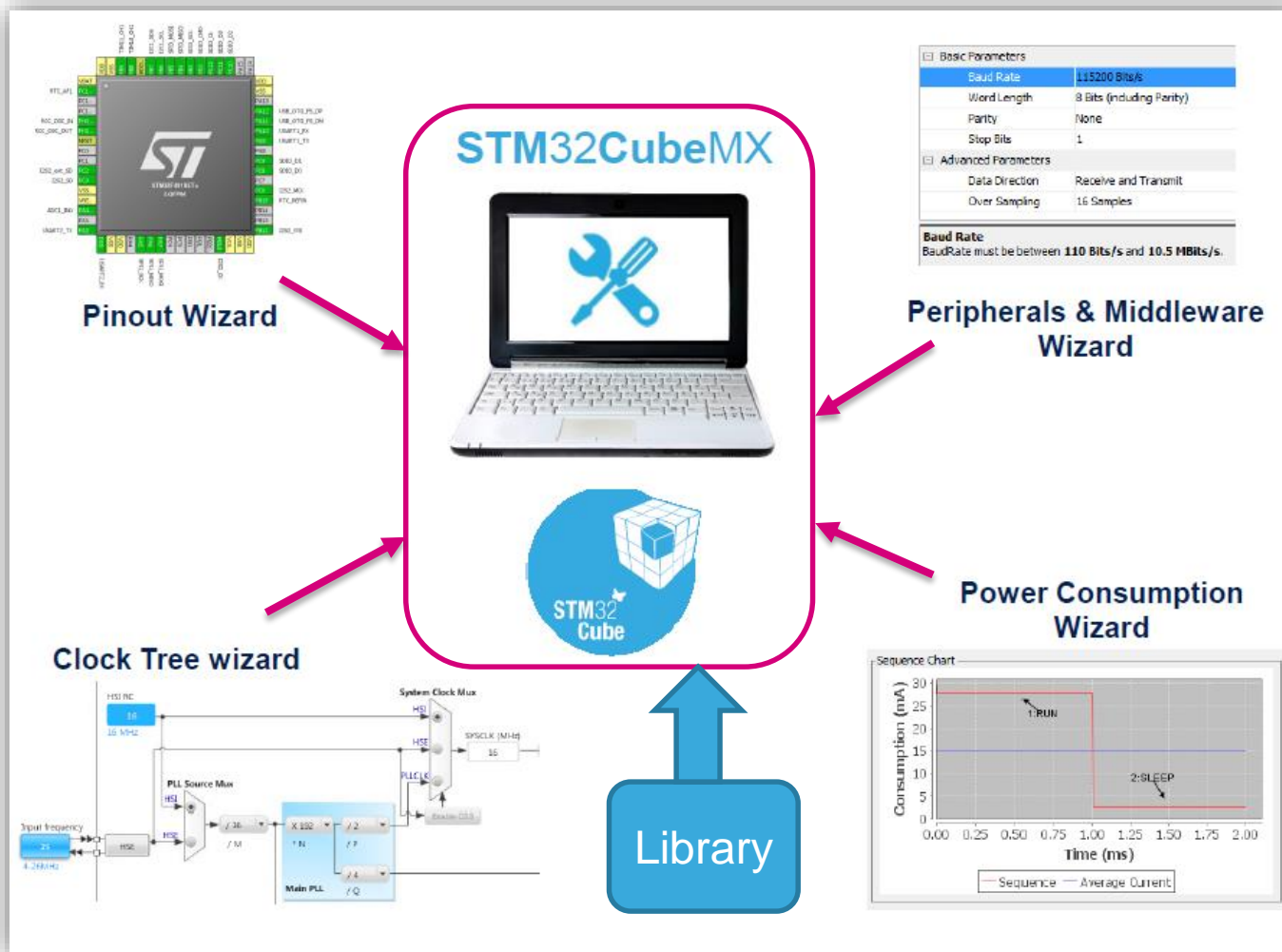
- MCU selector
- Pinout configuration
- Clock tree initialization
- Peripherals and middleware parameters
- Code generation
- Power consumption calculator





CUBE-MX

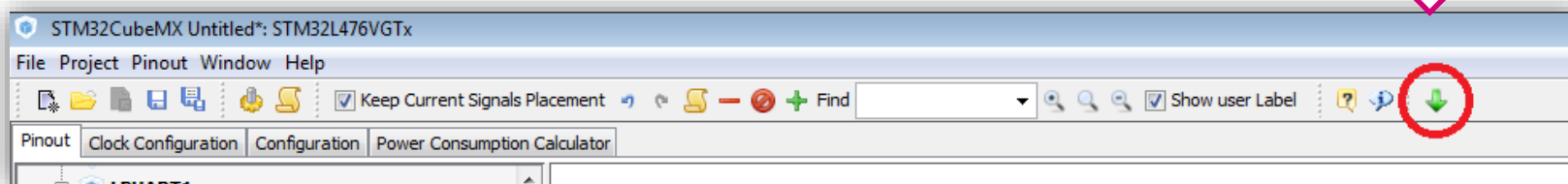
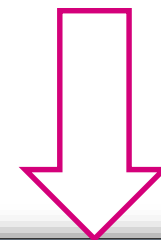
5



CUBE-MX - request update 1/2

6

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



CUBE-MX - request update 2/2

7

1. Help

2. Install New Libraries

4. Software to configure and manage STM32 MCUs

3. Check

Alternatively

STM32CubeMX Releases	STM32CubeF7 Releases	STM32CubeF4 Releases
<input type="checkbox"/> Software to configure and manage STM32 MCUs	<input checked="" type="checkbox"/> Firmware Package for Family STM32F7	<input checked="" type="checkbox"/> Firmware Package for Family STM32F4
4.12.0	1.3.1	1.10.1
4.13.0	1.2.0	1.9.0
	1.1.0	
	1.0.0	

CUBE-MX - How to move the Pin Positions

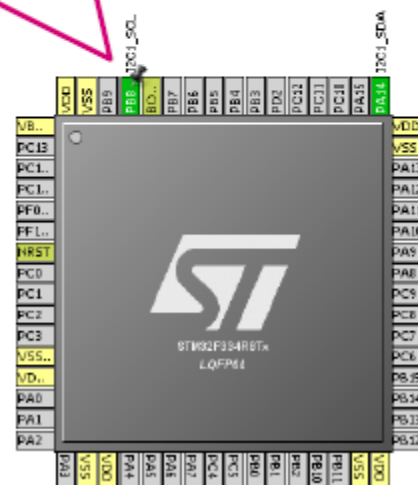
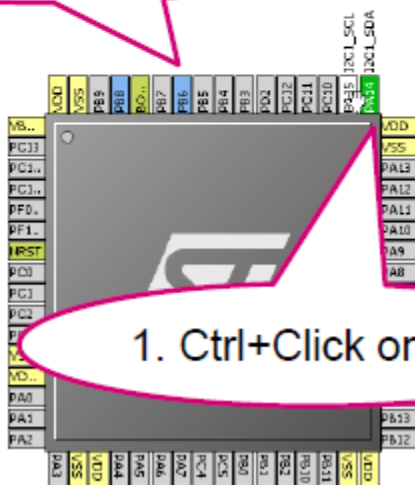
8

- 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)

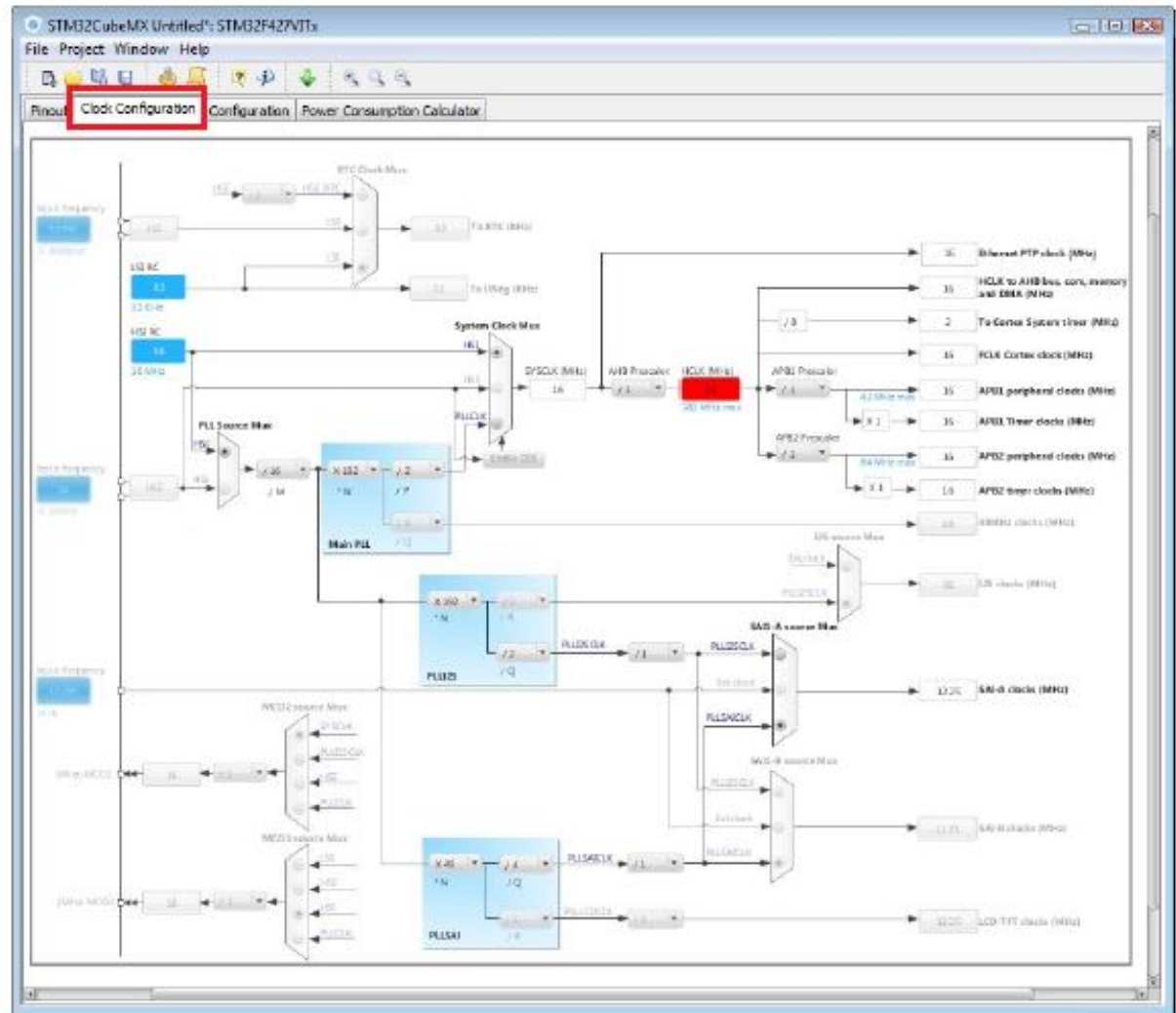
2. Show alternative positions

3. Move pin to new position

1. Ctrl+Click on pin



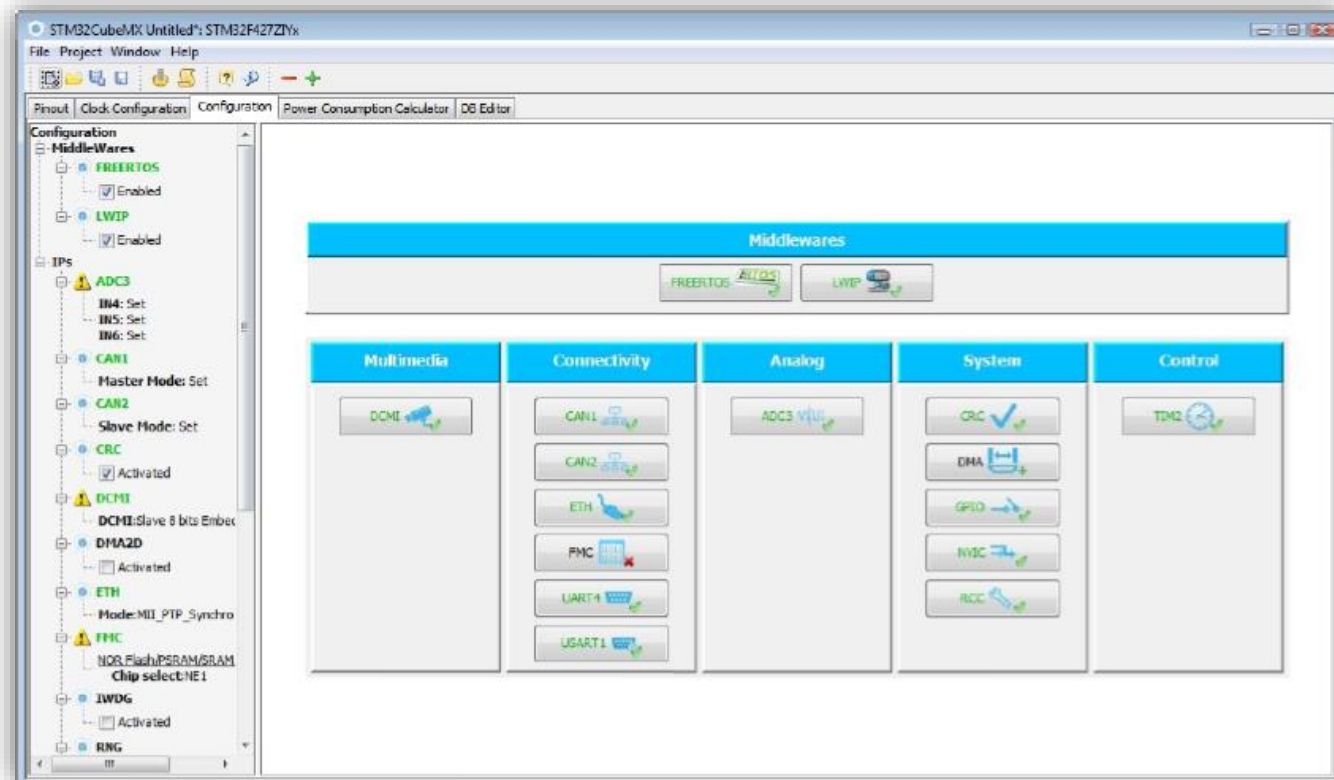
- Immediate display of all clock values
- Management of all clock constraints
- Highlight of errors



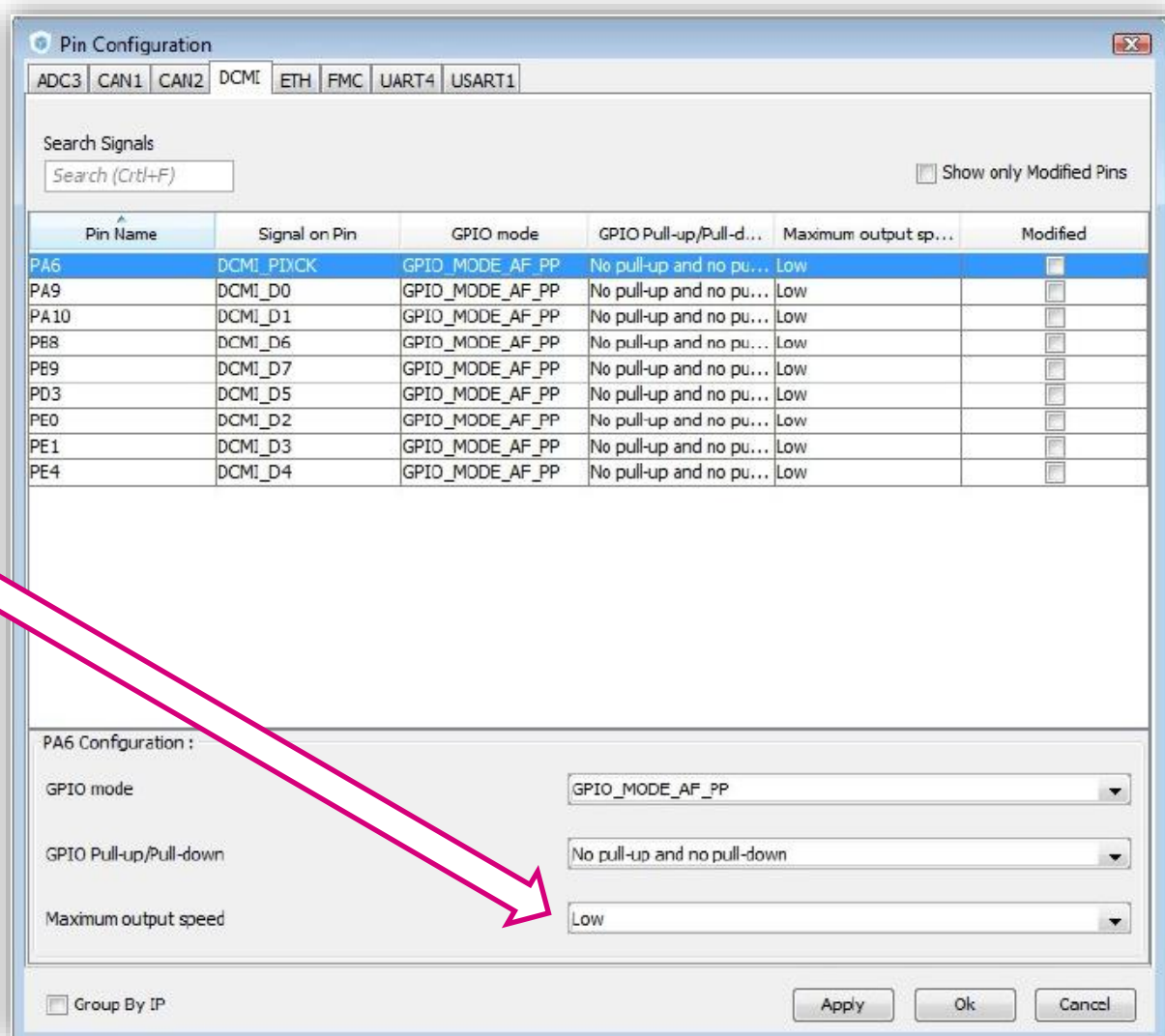
CUBE-MX - Peripheral and middleware configuration

10

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

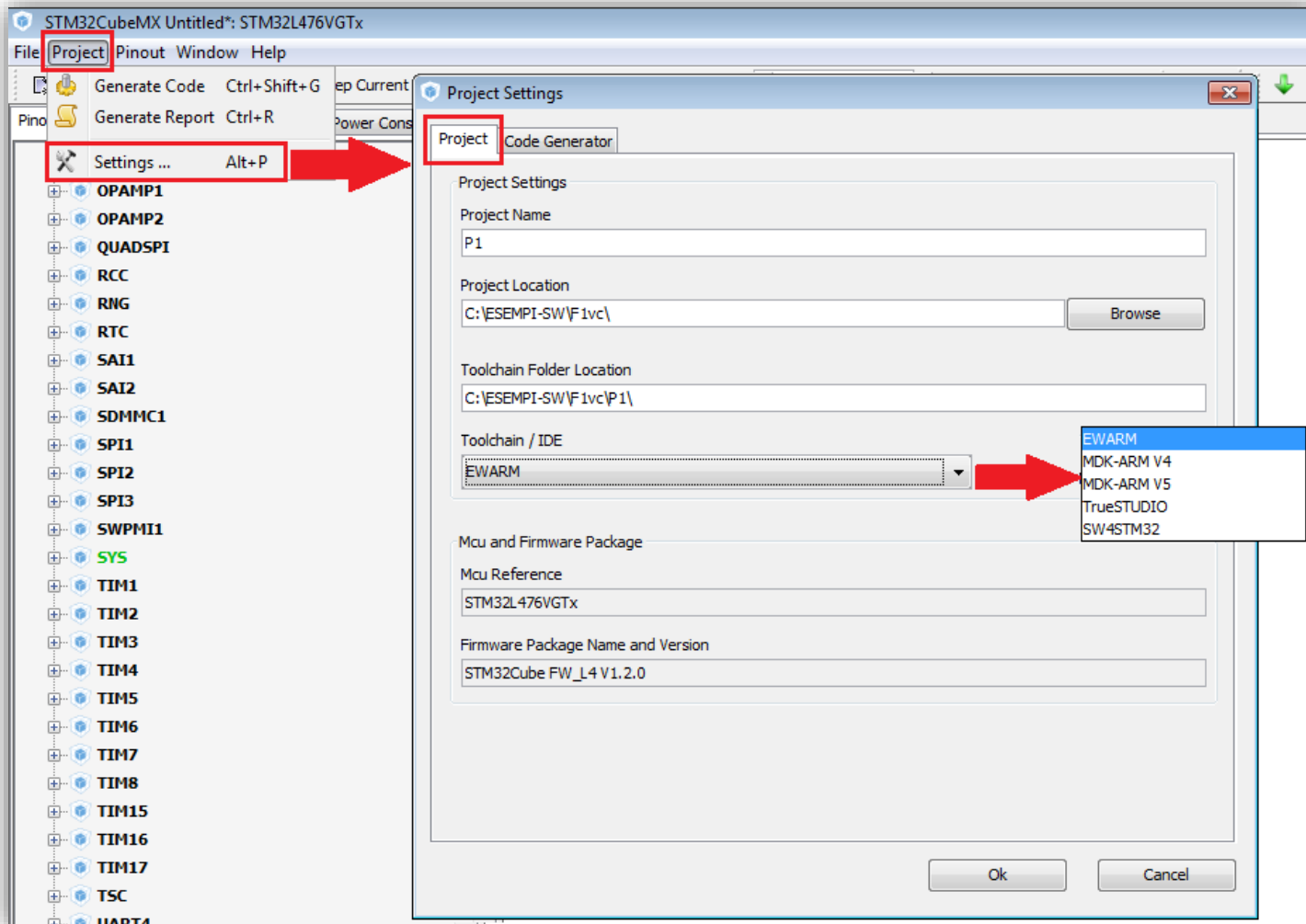


- Most of the GPIO parameters are set by default to the correct value
- You may want to change the maximum output speed
- You can select multiple pin at a time to set the same parameter



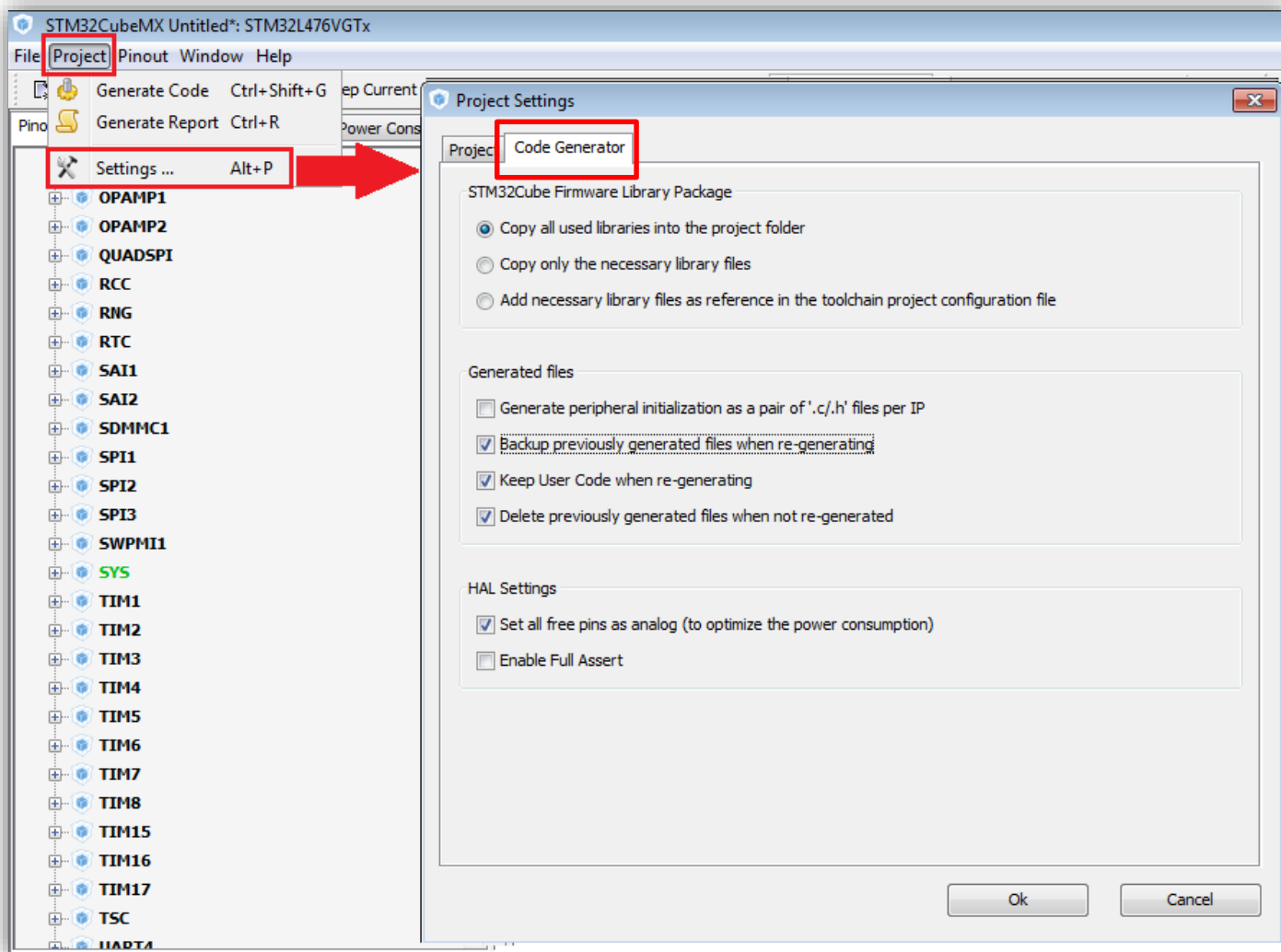
CUBE-MX - generate the code for some GUI 1/3

12



CUBE-MX - generate the code for some GUI 2/3

13



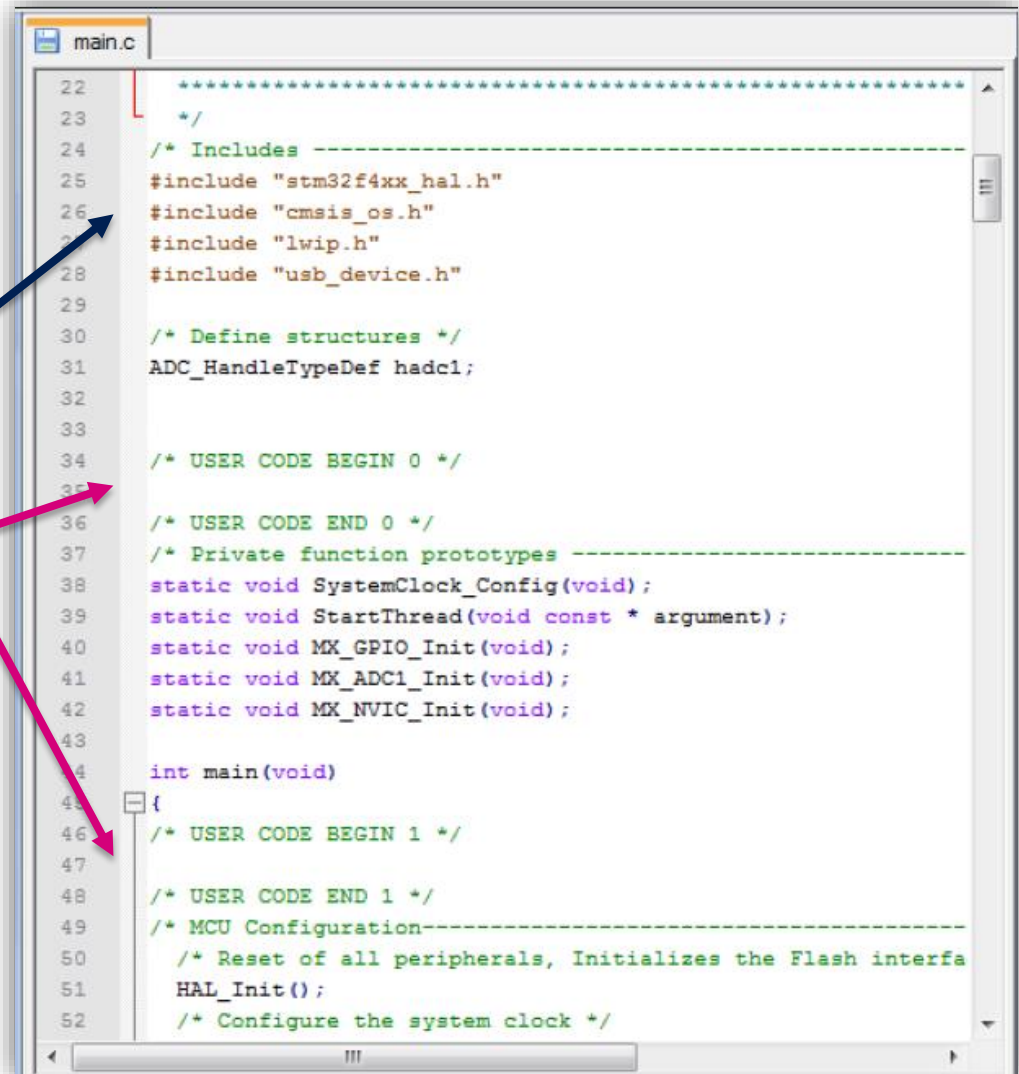
CUBE-MX - generate the code for some GUI 3/3

14

- 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



```
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 interface
51  HAL_Init();
52  /* Configure the system clock */
```

CUBE-MX - Power consumption calculator

15

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)

Idd by Step Average Current

Results Summary

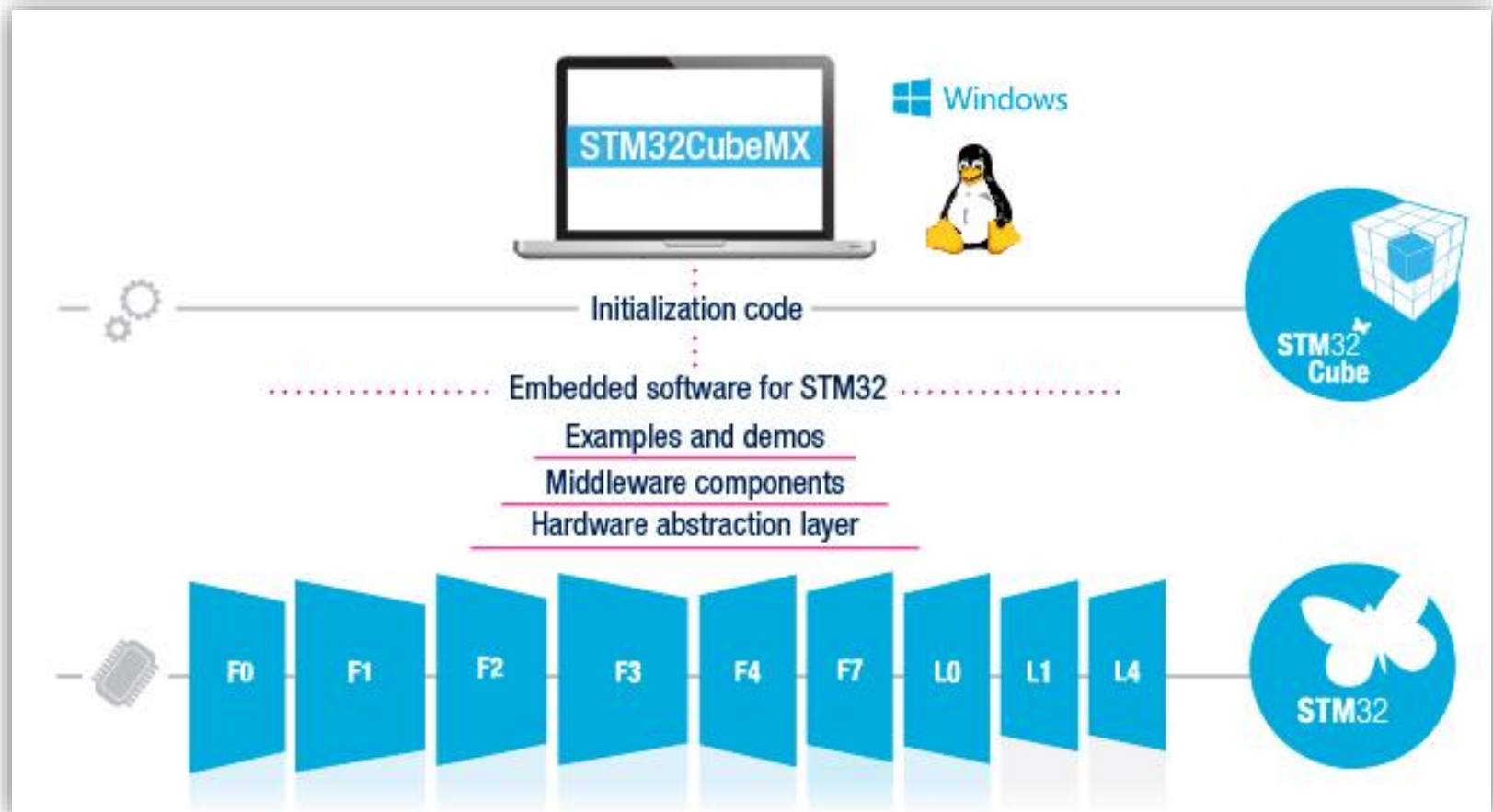
Sequence Time / Ta Max 104 ms / 85.0 °C
Battery Life Estimation 9 years, 1 month, 27 days & 11 hours

Average Consumption 38.61 µA
Average DMIPS 4.75 DMIPS

HAL library

HAL library == hardware abstraction layer

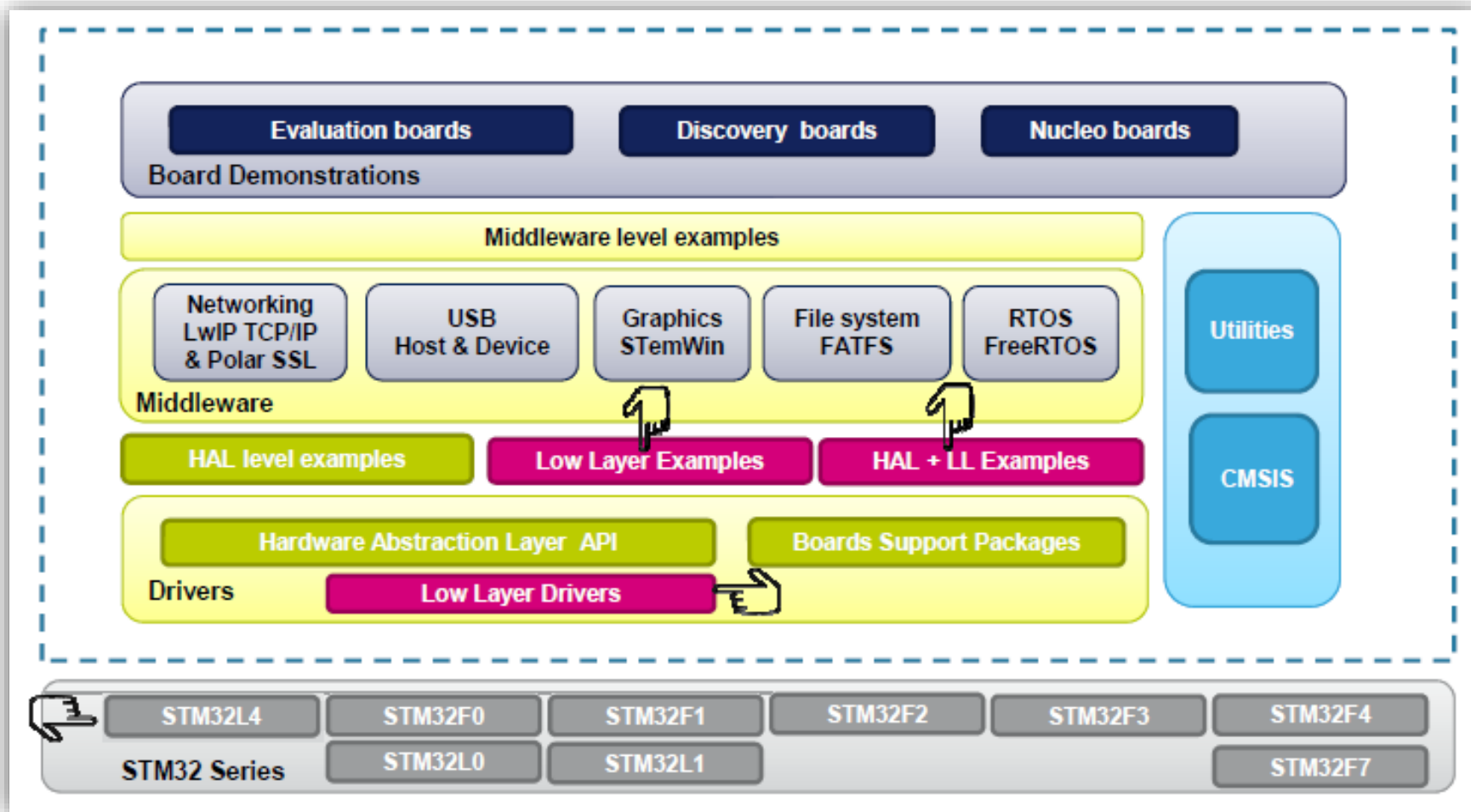
17



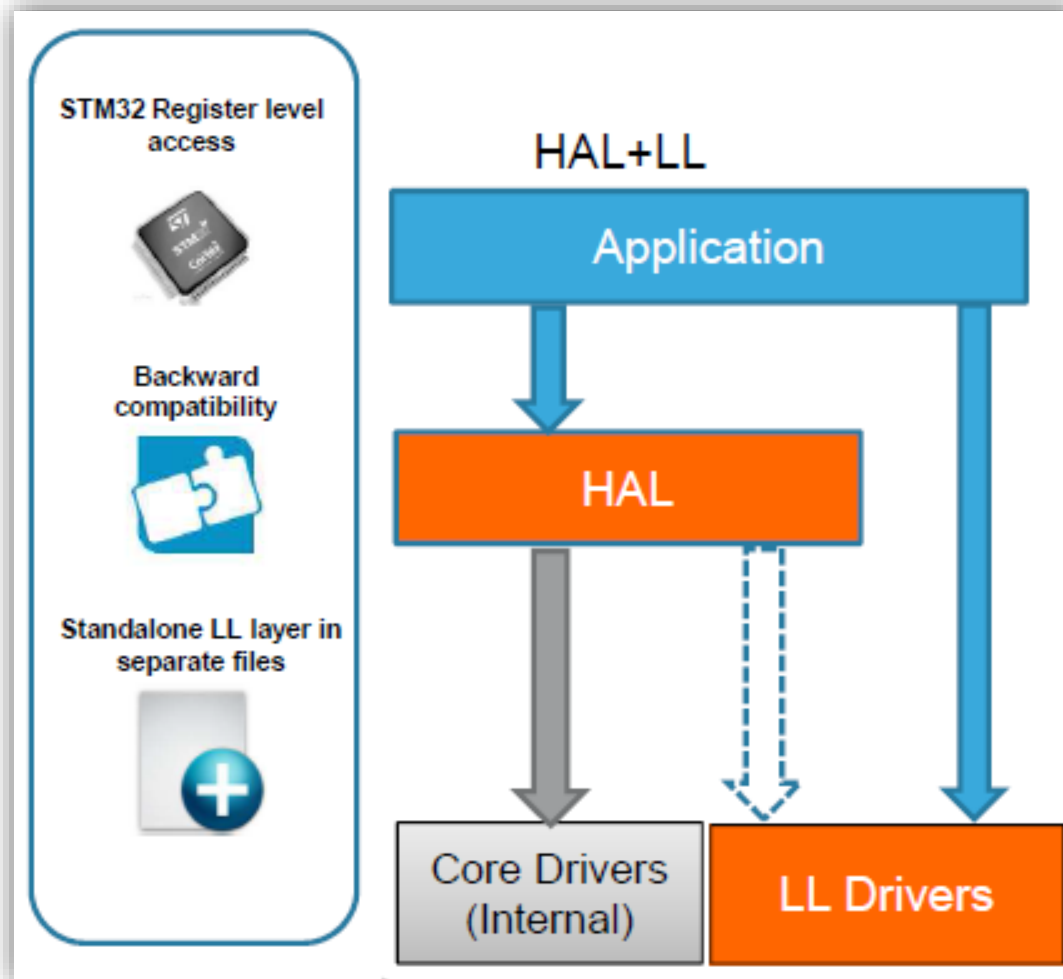
The HAL library are [here](#).

HAL library

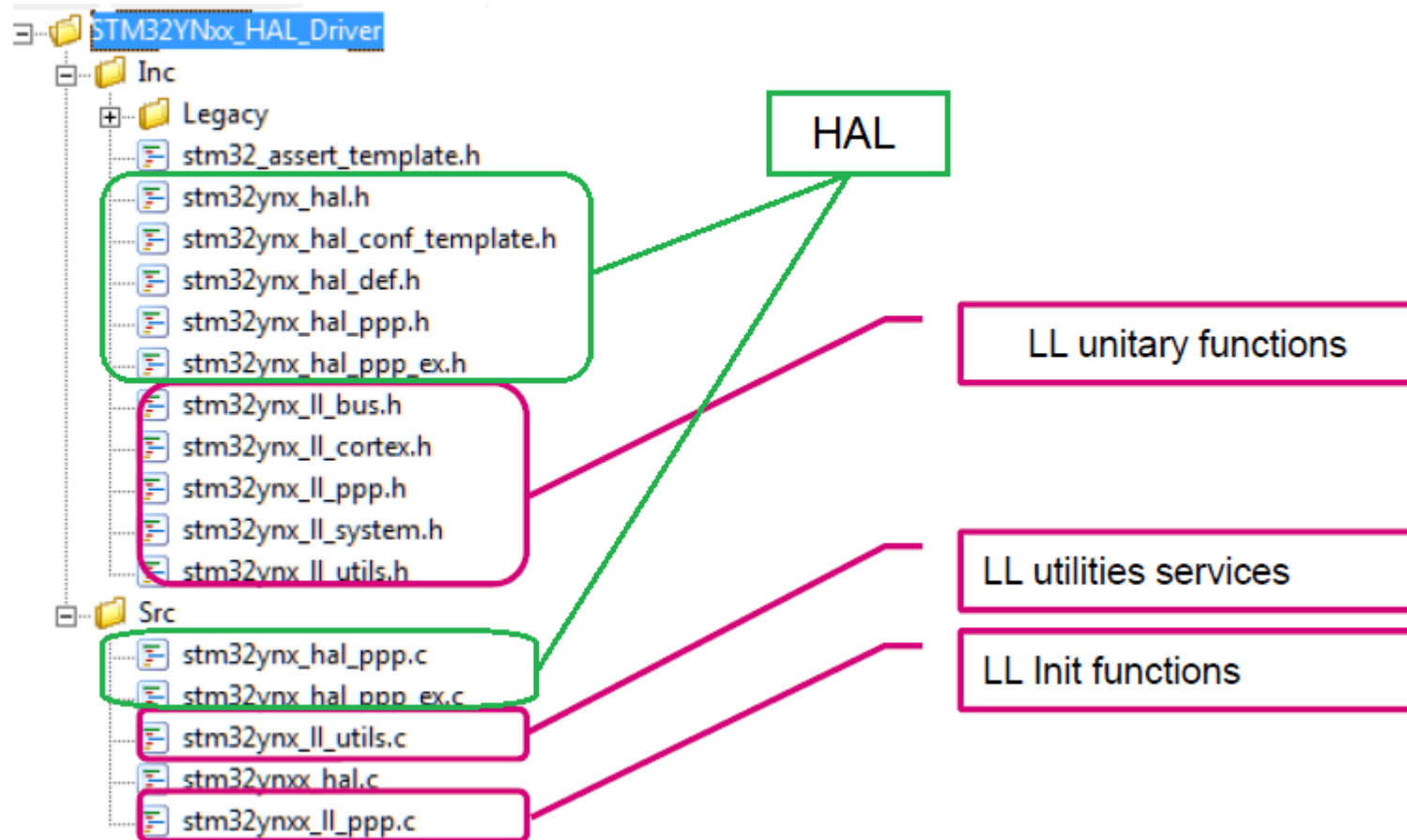
18



- **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



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



Covered peripherals (1/2)

Peripherals (IPs)		STM32Cube Support	
System	Flash	HAL	LL
		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, 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

Covered peripherals (2/2)

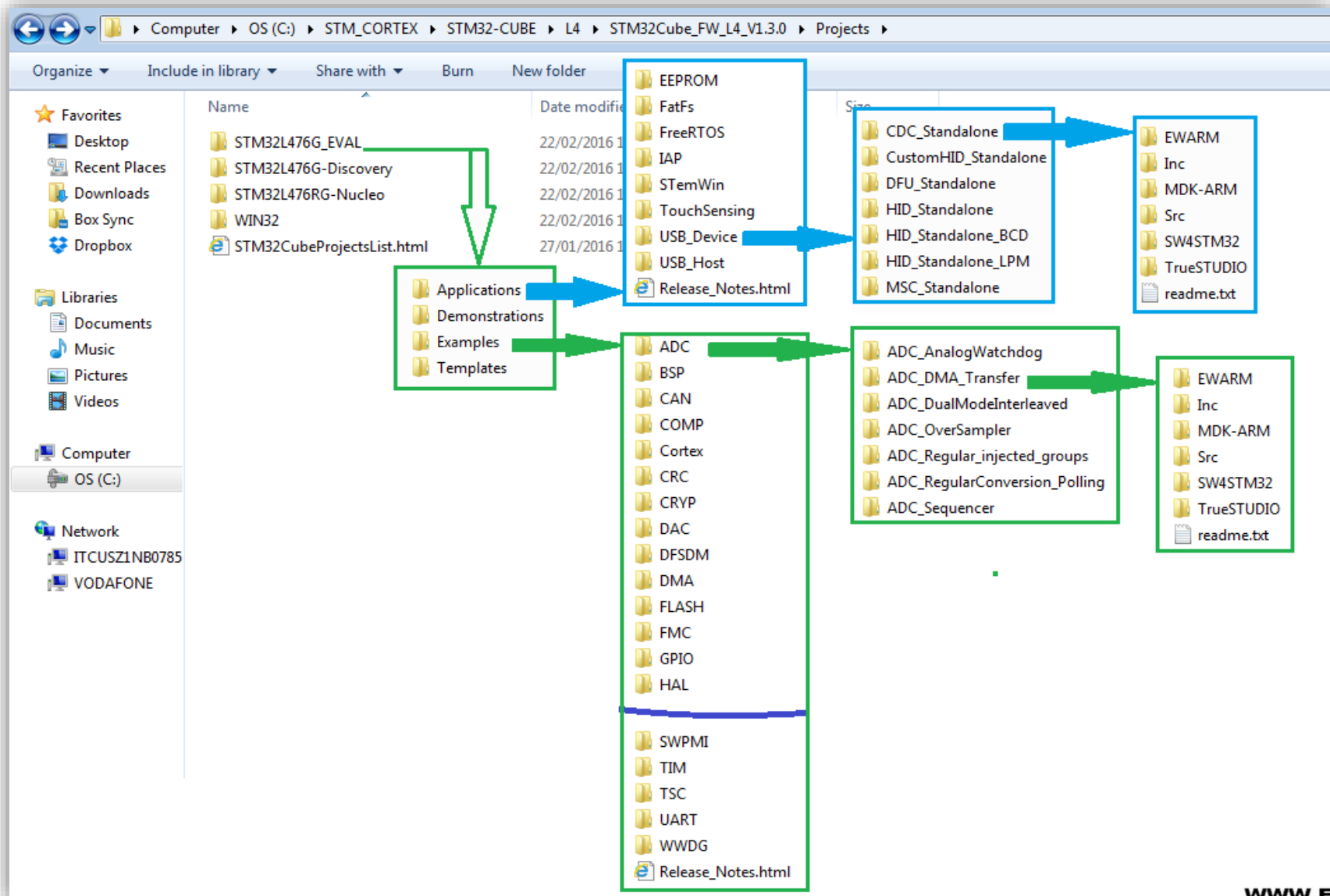
Peripherals (IPs)		STM32Cube Support	
		HAL	LL
Basic Connectivity	I2C/SMBUS	Yes	Yes
	UART/USART/LPUART	Yes	Yes
	SWPMI	Yes	Yes
	SPI/I2S	Yes	Yes
	SDMMC(SDIO)	Yes	No
	CAN	Yes	No
	CEC	Yes	No
Advanced Connectivity	USB-FS-Device	Yes	No
	USB-OTG-FS/HS	Yes	No
	Ethernet	Yes	No
	MDIOS	Yes	No
Interface	FSMC(FMC)	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 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 ?

25



HAL library - HAL examples

26

The diagram illustrates the HAL library structure and a project example. On the left, a file explorer shows the HAL library hierarchy: _htmresc, Documentation, Drivers, Middlewares, Projects, Applications, Demonstrations, and Examples. The Examples folder is expanded, showing subfolders for Examples_LL and Examples_MIX. A callout box points to the Examples_LL folder, stating: "Examples that are based ONLY on HAL drivers (as of today)". Below this, a list of HAL drivers is shown: ADC, COMP, Cortex, CRC, DAC, DMA, DMA_FLASHToRAM (highlighted with a pink box), and FLASH. On the right, a screenshot of the STM32L476RG_NUCLEO project in an IDE shows the project structure. The project is named "Project - STM32L476RG_NUCLEO". It contains folders for Doc, Drivers, BSP, CMSIS, and STM32L4xx_HAL_Driver. The STM32L4xx_HAL_Driver folder is expanded, showing files: stm32l4xx_hal.c, stm32l4xx_hal_cortex.c, stm32l4xx_hal_dma.c (highlighted with a pink box), stm32l4xx_hal_gpio.c, stm32l4xx_hal_pwr.c, stm32l4xx_hal_pwr_ex.c, stm32l4xx_hal_rcc.c, and stm32l4xx_hal_rcc_ex.c. Below the project structure, the text "HAL project (no LL services used in the application)" is displayed.

HAL library - LL examples

27

The image shows the STM32CubeIDE interface with a project named 'STM32L476RG_NUCLEO'. The project explorer on the left shows a tree structure with 'Examples_LL' highlighted. A red box highlights 'Examples_LL' in the project explorer, and a red line points to a text box that says 'NEW Examples that are based ONLY on LL drivers'. Another red box highlights 'stm32l4xx_ll_dma.h' in the 'User' folder of the project view. A third red box highlights 'DMA_CopyFromFlashToMemory' in the 'DMA' folder of the project view. The project view shows the following files:

- Project - STM32L476RG_NUCLEO
 - Doc
 - Drivers
 - CMSIS
 - STM32L4xx_LL_Driver
 - stm32l4xx_utils.c
 - Example
 - EWARM
 - User
 - main.c
 - Output
 - cmsis_lcr.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
 - stm32l476xx.h
 - stm32l4xx.h
 - stm32l4xx_ll_bus.h
 - stm32l4xx_ll_dma.h
 - stm32l4xx_ll_gpio.h

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

The project view also shows the following folders:

- 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

28

The image displays the project structure of an STM32CubeIDE project named "Project - STM32L475RG_NUCLEO". The project is organized into several folders: "Doc", "Drivers", "BSP", "CMSIS", "STM32L4xx_HAL_Driver", "Example", "EWARM", and "User". The "STM32L4xx_HAL_Driver" folder contains a list of HAL drivers, including "stm32l4xx_hal.c", "stm32l4xx_hal_cortex.c", "stm32l4xx_hal_dma.c", "stm32l4xx_hal_gpio.c", "stm32l4xx_hal_i2c.c", "stm32l4xx_hal_iwdg.c", "stm32l4xx_hal_lptim.c", "stm32l4xx_hal_opamp.c", "stm32l4xx_hal_pwr.c", "stm32l4xx_hal_rcc.c", and "stm32l4xx_hal_tim.c". The "Example" folder contains a "main.c" file. The "User" folder contains a "main.c" file. The "Output" folder contains a list of generated files, including "cmsis_jar.h", "core_cm4.h", "core_cm4func.h", "core_cm4init.h", "core_cm4simd.h", "DLib_Config_Full.h", "DLib_Defaults.h", "DLib_Product.h", "DLib_Threads.h", "intrinsics.h", "main.h", "stdio.h", "stdlib.h", "stm32_hal_legacy.h", "stm32l475rg.h", "stm32l4xx.h", "stm32l4xx_hal.h", "stm32l4xx_hal_conf.h", "stm32l4xx_hal_cortex.h", "stm32l4xx_hal_def.h", "stm32l4xx_hal_dma.h", "stm32l4xx_hal_gpio.h", "stm32l4xx_hal_gpio_ex.h", "stm32l4xx_hal_i2c.h", "stm32l4xx_hal_i2c_ex.h", "stm32l4xx_hal_iwdg.h", "stm32l4xx_hal_lptim.h", "stm32l4xx_hal_opamp.h", "stm32l4xx_hal_pwr.h", "stm32l4xx_hal_rcc.h", "stm32l4xx_hal_rcc_ex.h", and "stm32l4xx_hal_tim.h".

NEW Examples that are based on HAL and LL drivers (Mixed)

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

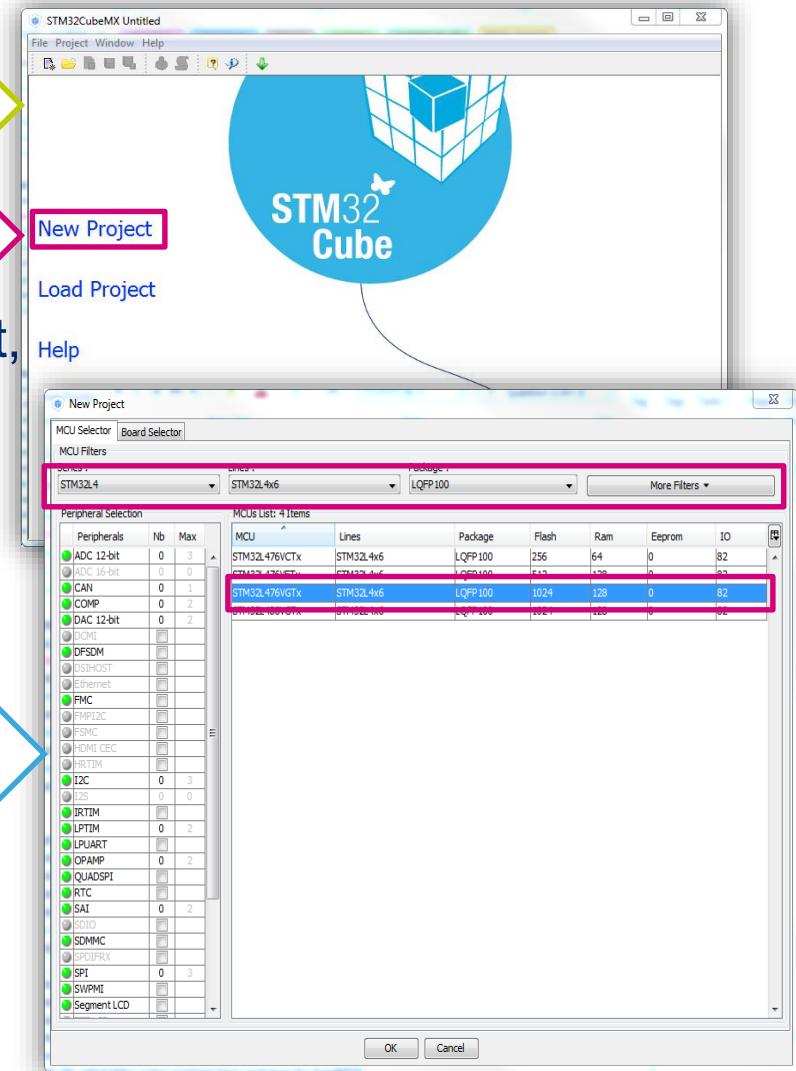
Mixed HAL and LL drivers used in the application

New Project

CUBE - Start new project

30

- Run CubeMX tool
- Start **new project**
 - Click “New Project” desktop shortcut, or
 - Go to “Menu->File->New Project”

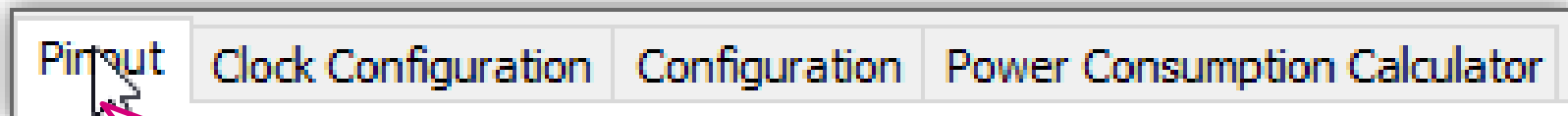


- Filter:
 - Series: STM32L4
 - Line: STM32L4x6
 - Package: LQFP100
- Select: **STM32L476VGTx**
(STM32L476-DISCOVERY)

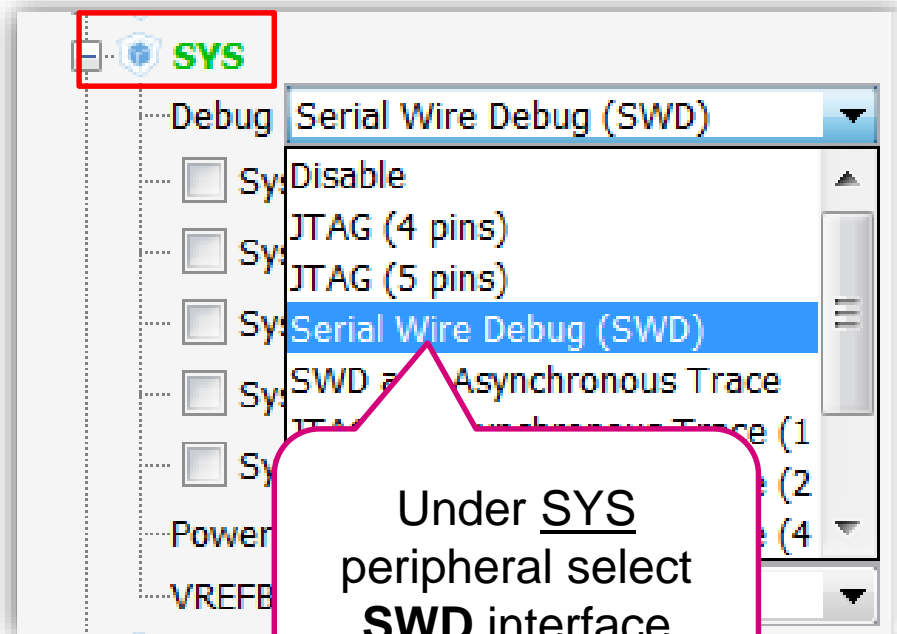
CUBE - Start new project

Configure debug interface 1/2

31



Go to Pinout settings



Under SYS
peripheral select
SWD interface

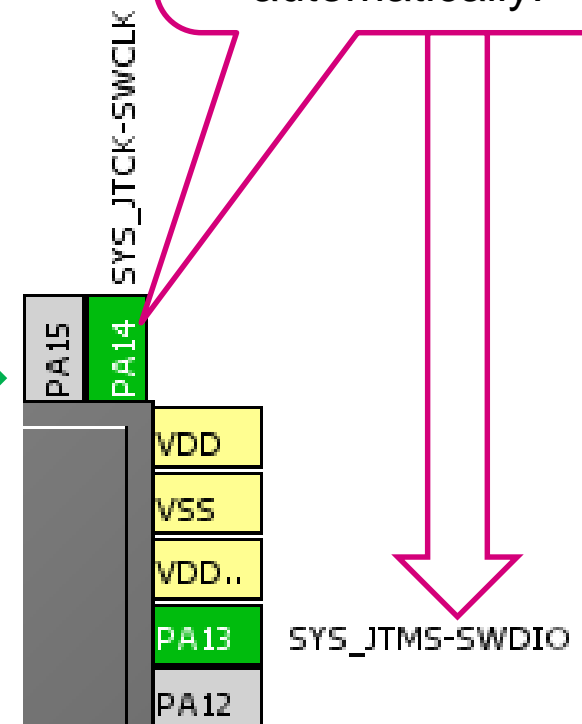
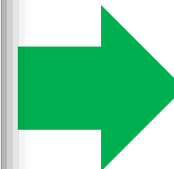
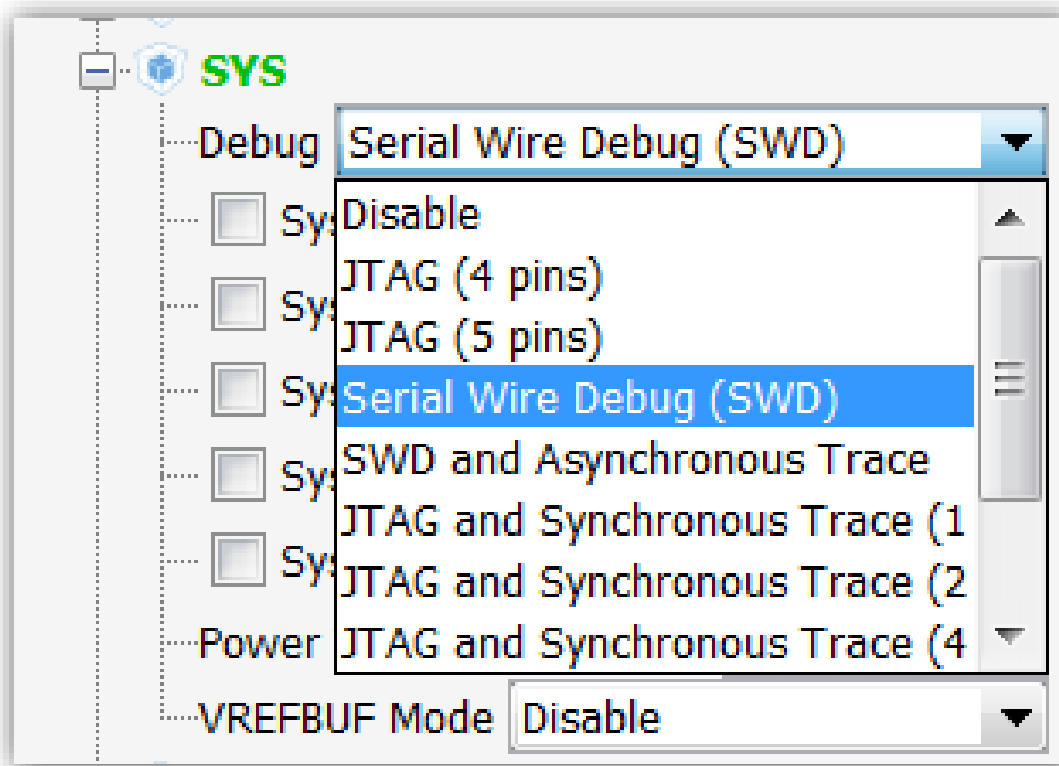
CUBE - Start new project

Configure debug interface 2/2

32



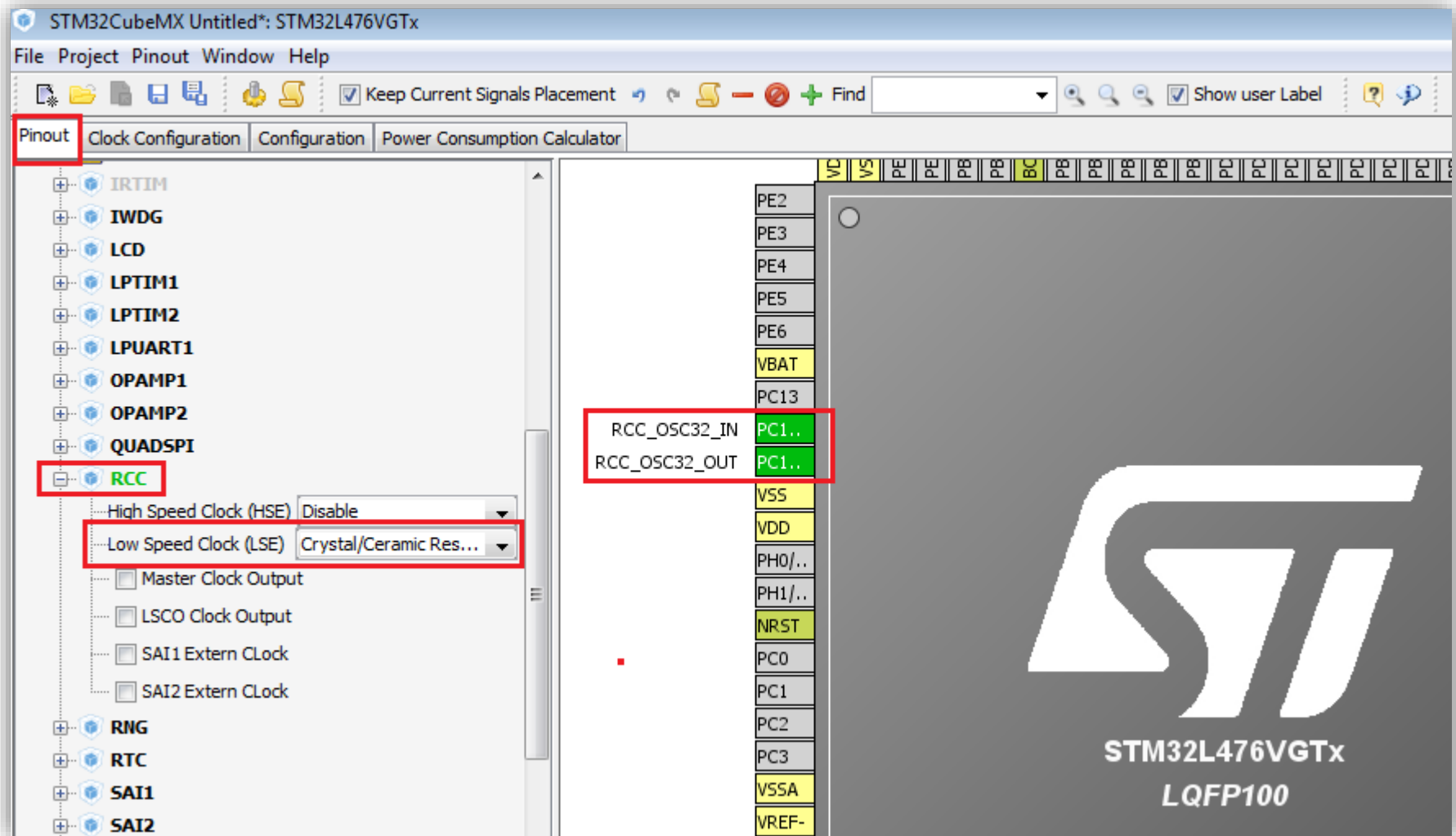
The corresponding pins are assigned and configured automatically!



CUBE - Start new project

Configure LSE resonator (32,768 KHz)

33

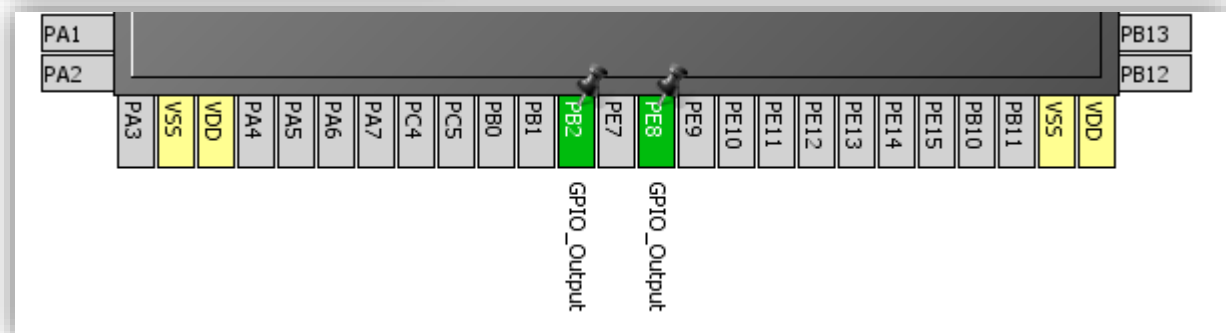
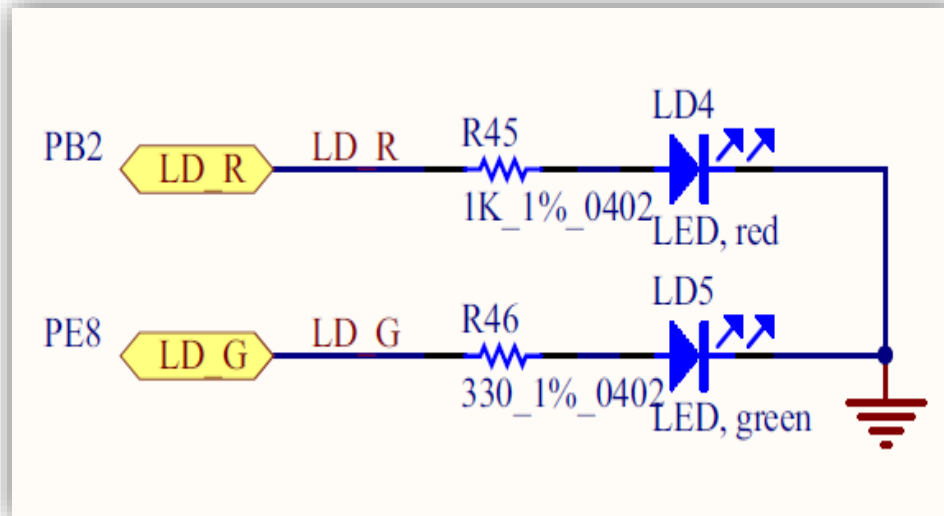


CUBE - Start new project

Configure GPIO for LED toggling

34

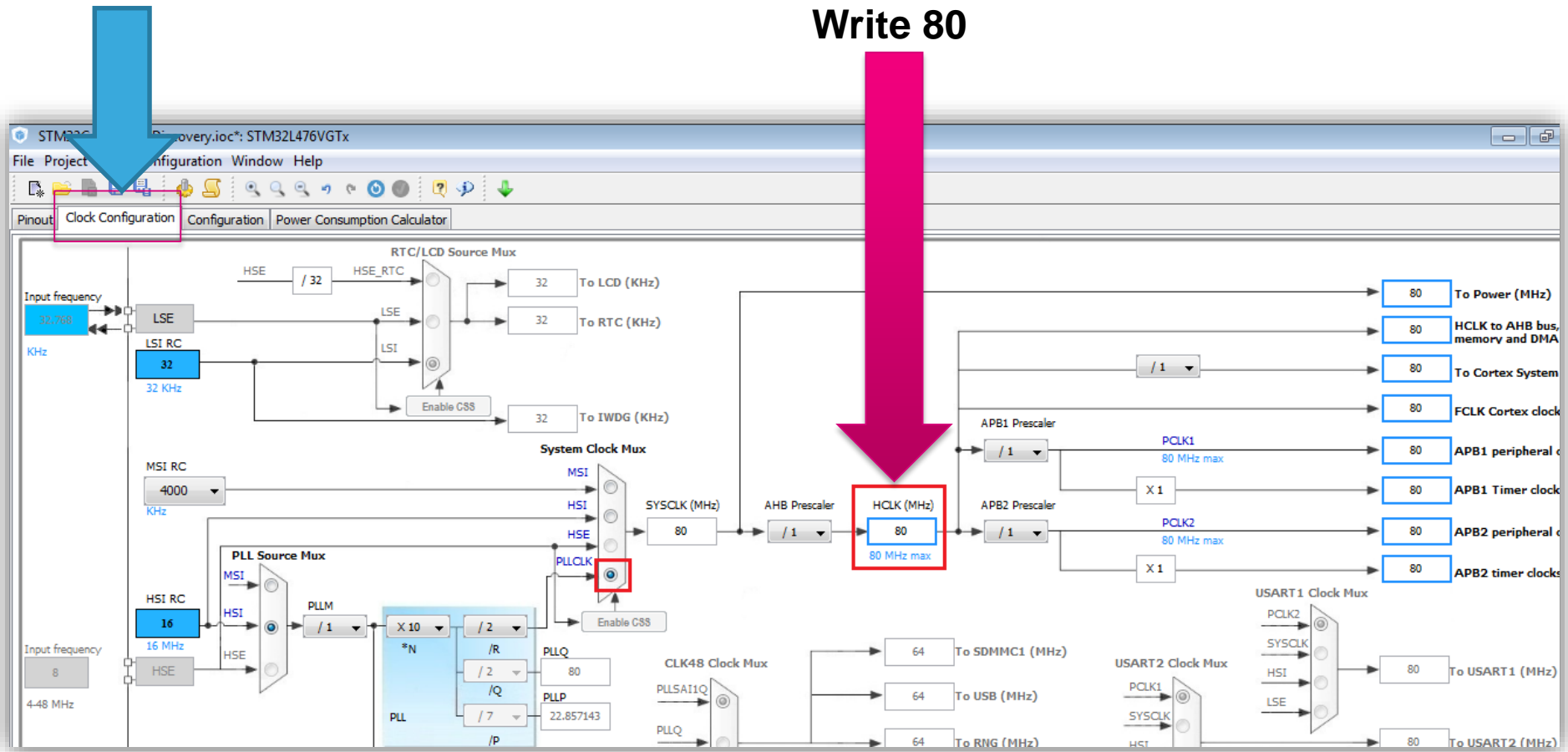
Configure LED pin as GPIO_Output



CUBE - Start new project

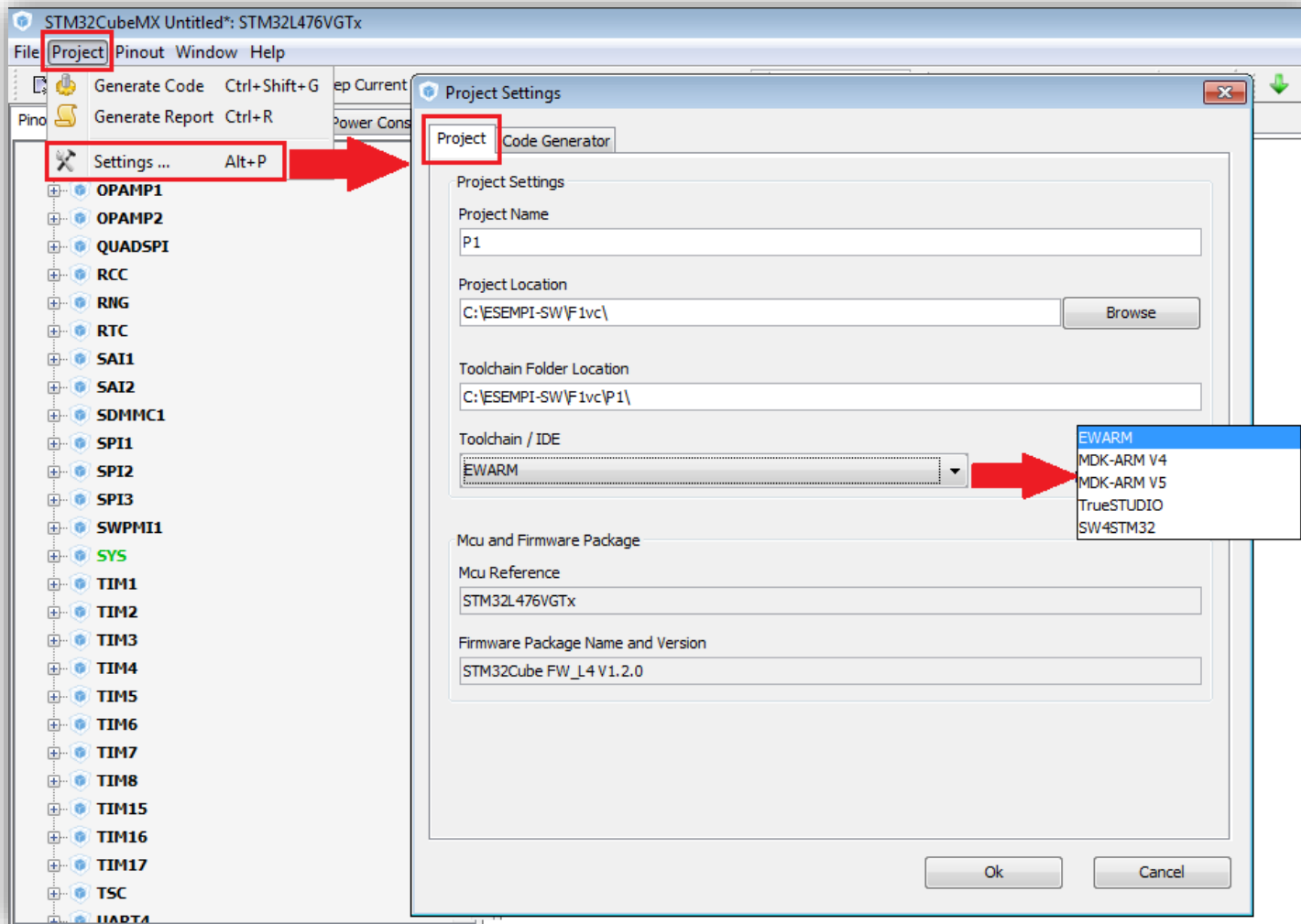
Clock configuration

35



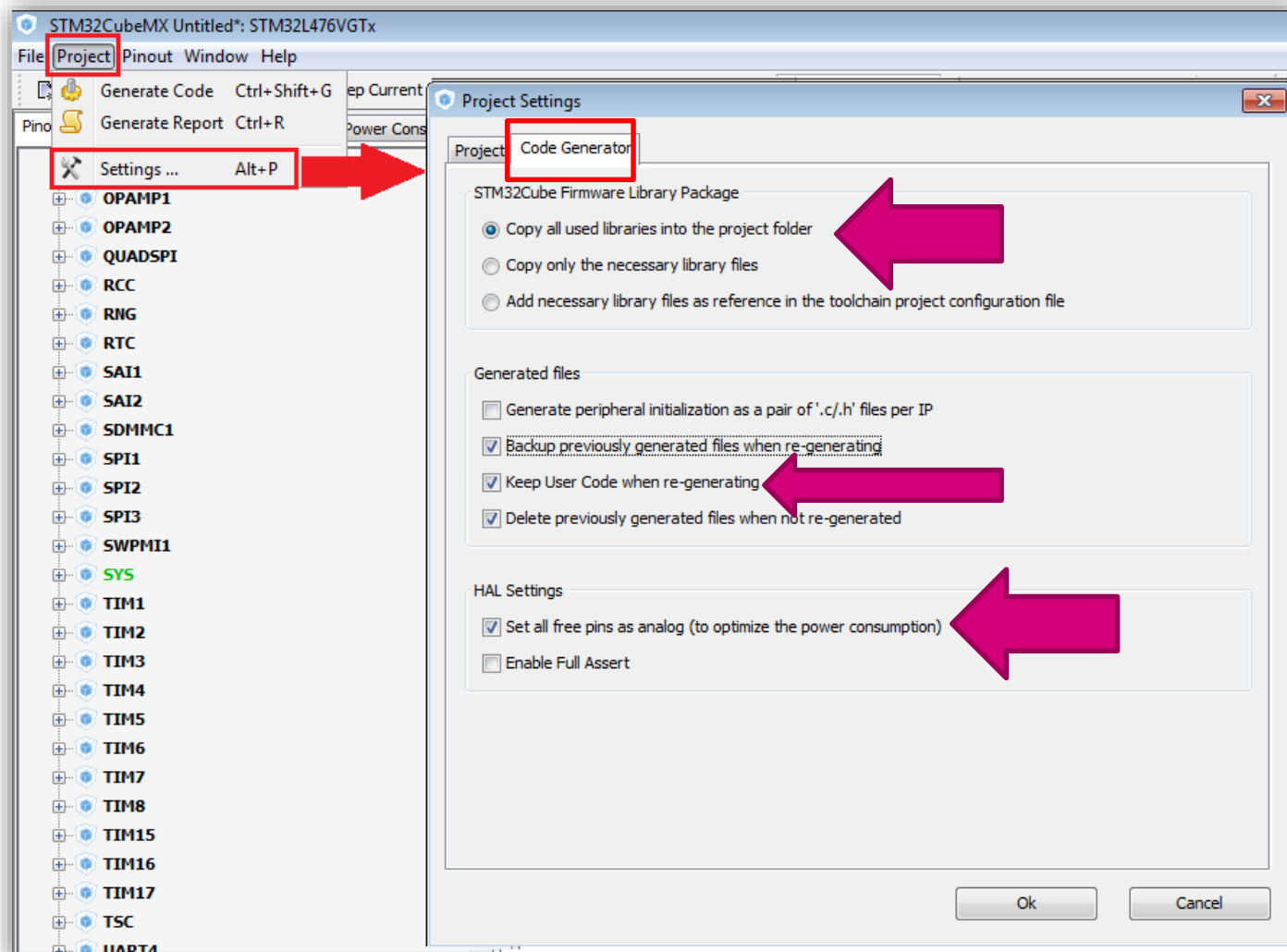
CUBE - Start new project generate the code for some GUI 1/3

36



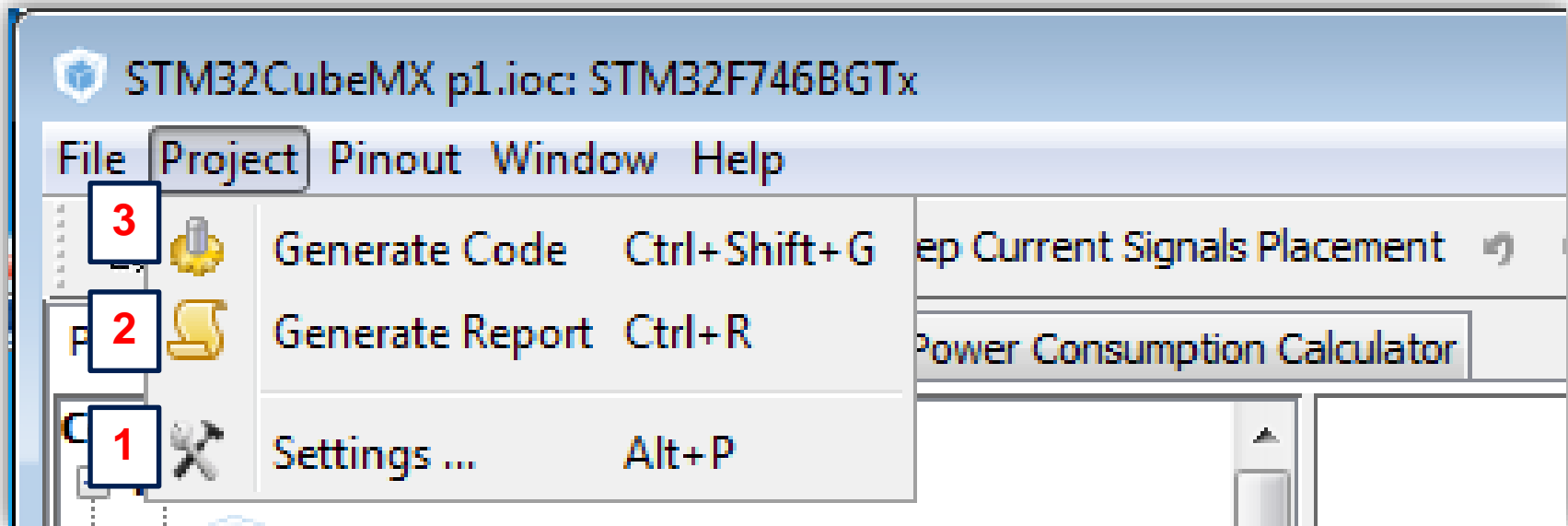
CUBE - Start new project generate the code for some GUI 2/3

37



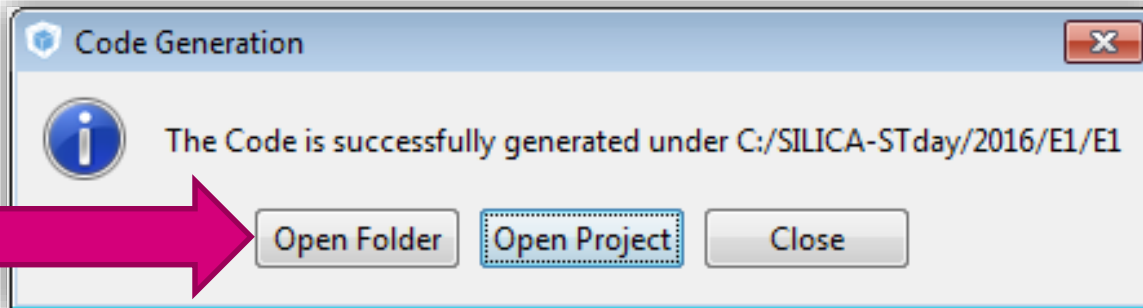
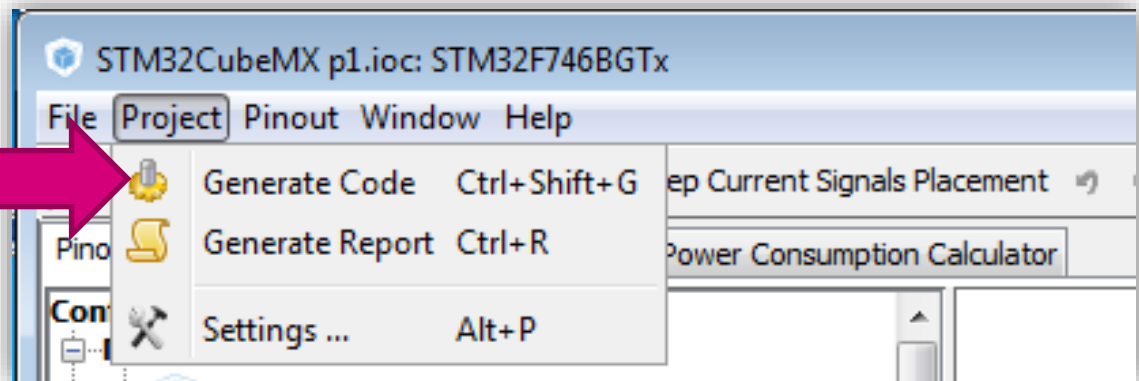
CUBE - Start new project generate the code for some GUI 1/3

38



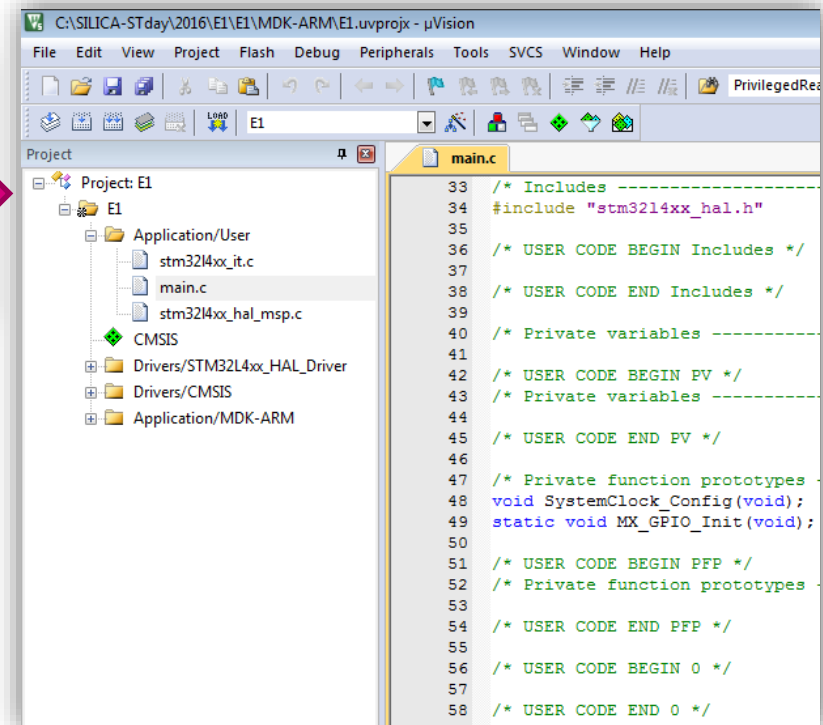
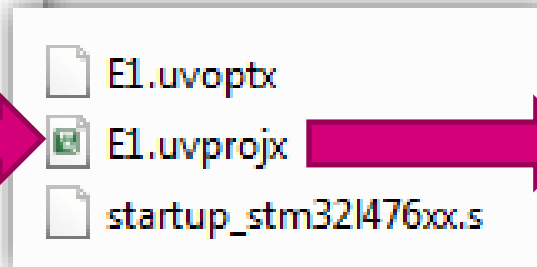
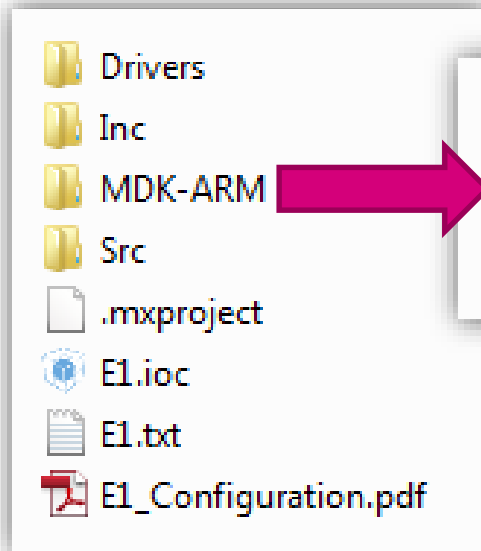
CUBE - Start new project generate the code 2/3

39



CUBE - Start new project generate the code 3/3

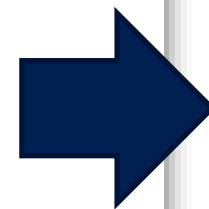
40



CUBE - Start new project add code for flashing LEDs

41

```
main.c
74
75  /* Initialize all configured peripherals */
76  MX_GPIO_Init();
77
78  /* USER CODE BEGIN 2 */
79
80  /* USER CODE END 2 */
81
82  /* Infinite loop */
83  /* USER CODE BEGIN WHILE */
84  while (1)
85  {
86    /* USER CODE END WHILE */
87
88    /* USER CODE BEGIN 3 */
89    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
90    HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);
91    /* Delay 200 ms */
92    HAL_Delay(200);
93  }
94  /* USER CODE END 3 */
95
96 }
97
```

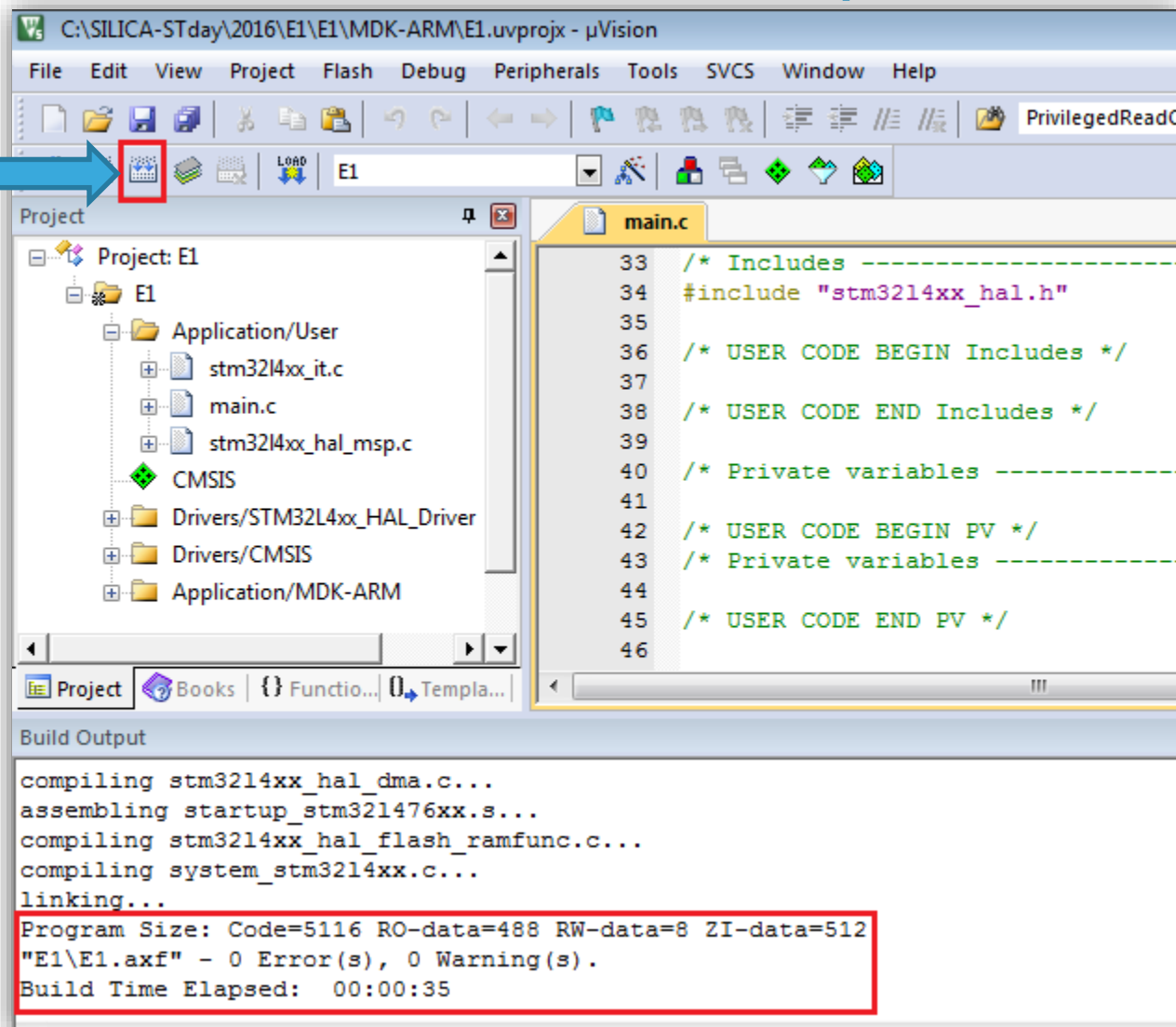


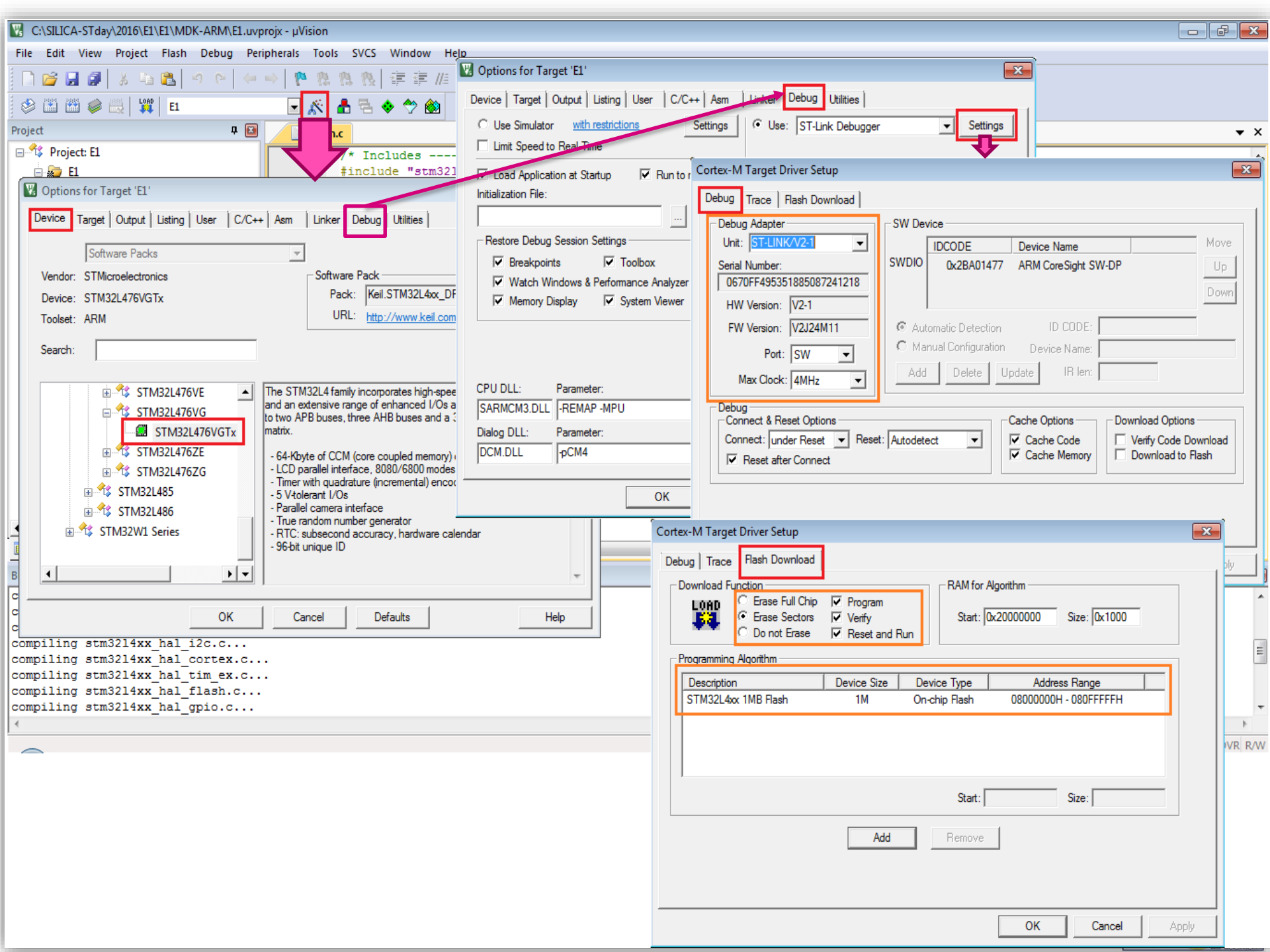
See the:
UM1884 - Description
of STM32L4 HAL and LL
drivers

CUBE - Start new project compile and debug

42

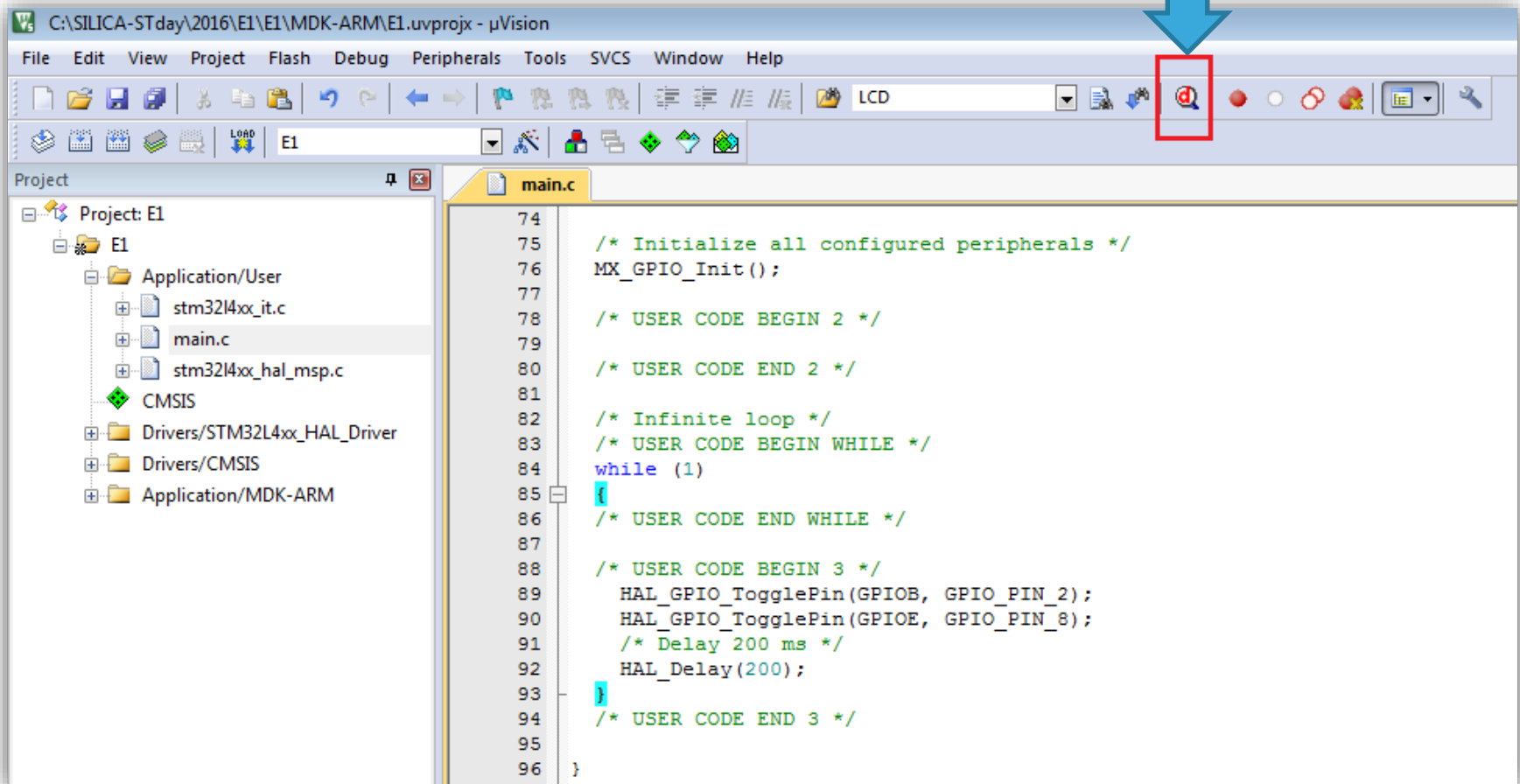
COMPILE





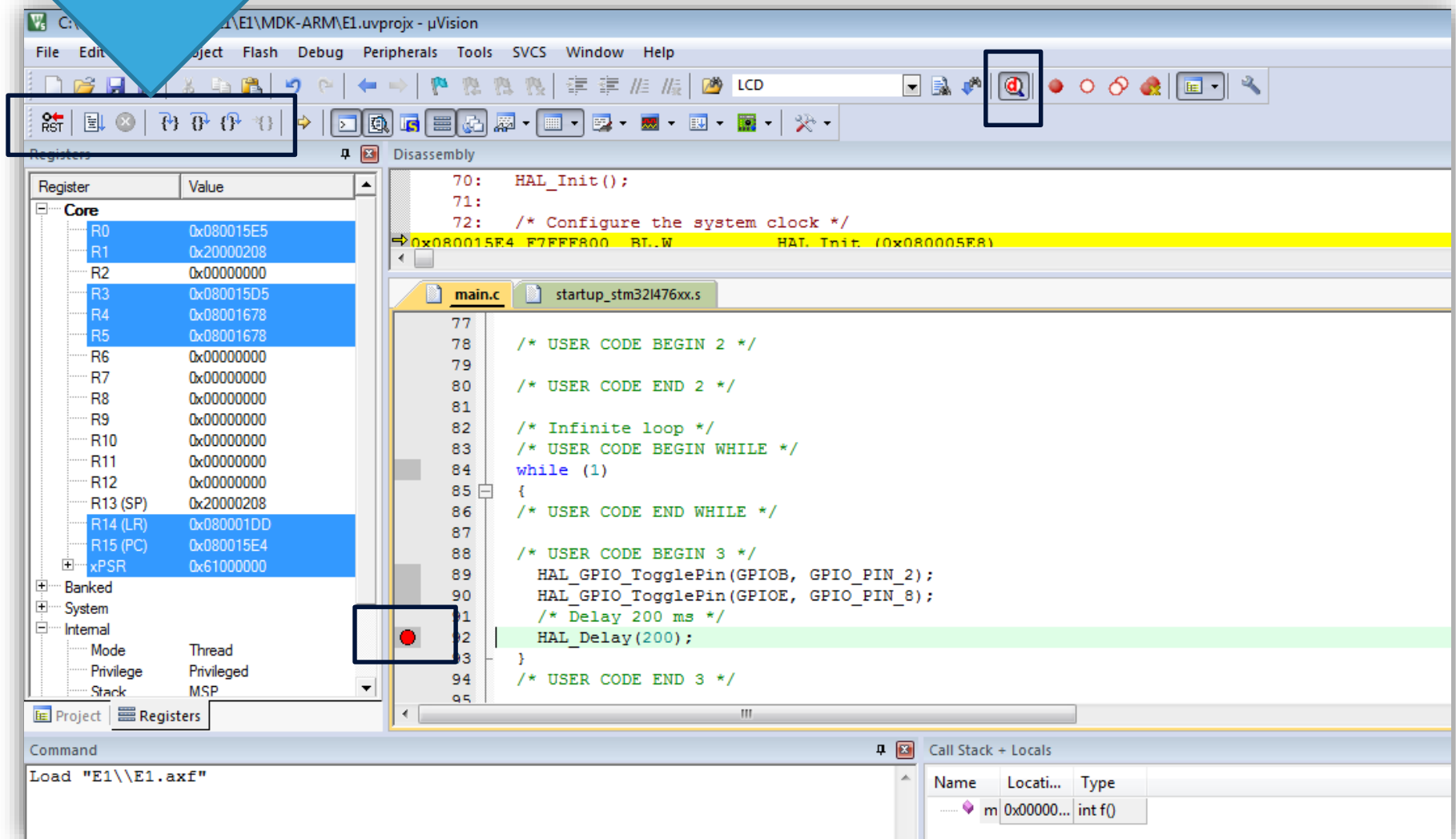
CUBE - Start new project compile and debug

44



CUBE - Start new project compile and debug

45



The screenshot displays the uVision IDE interface for a CUBE project. A large blue arrow points to the 'RST' button in the toolbar. The 'Registers' window on the left shows the Core registers, with R14 (LR) and R15 (PC) highlighted. The 'Disassembly' window shows the HAL_Init() function. The 'main.c' file is open, showing the user code. The 'Command' window at the bottom shows the command 'Load "E1\\E1.axf"'. The 'Call Stack + Locals' window shows the current function call.

Register	Value
R0	0x080015E5
R1	0x20000208
R2	0x00000000
R3	0x080015D5
R4	0x08001678
R5	0x08001678
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x08001DD
R15 (PC)	0x080015E4
xPSR	0x61000000

```
70:  HAL_Init();
71:
72:  /* Configure the system clock */
73:  HAL_Init();
74:  /* USER CODE BEGIN 2 */
75:
76:  /* USER CODE END 2 */
77:
78:  /* Infinite loop */
79:  /* USER CODE BEGIN WHILE */
80:  while (1)
81:  {
82:      /* USER CODE END WHILE */
83:
84:      /* USER CODE BEGIN 3 */
85:      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_2);
86:      HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_8);
87:      /* Delay 200 ms */
88:      HAL_Delay(200);
89:  }
90:  /* USER CODE END 3 */
91:
92:
93:
94:
95:
```

Command: Load "E1\\E1.axf"

Name	Locati...	Type
.....	m	0x00000... int f()



TrueSTUDIO Lite

free tools for ARM

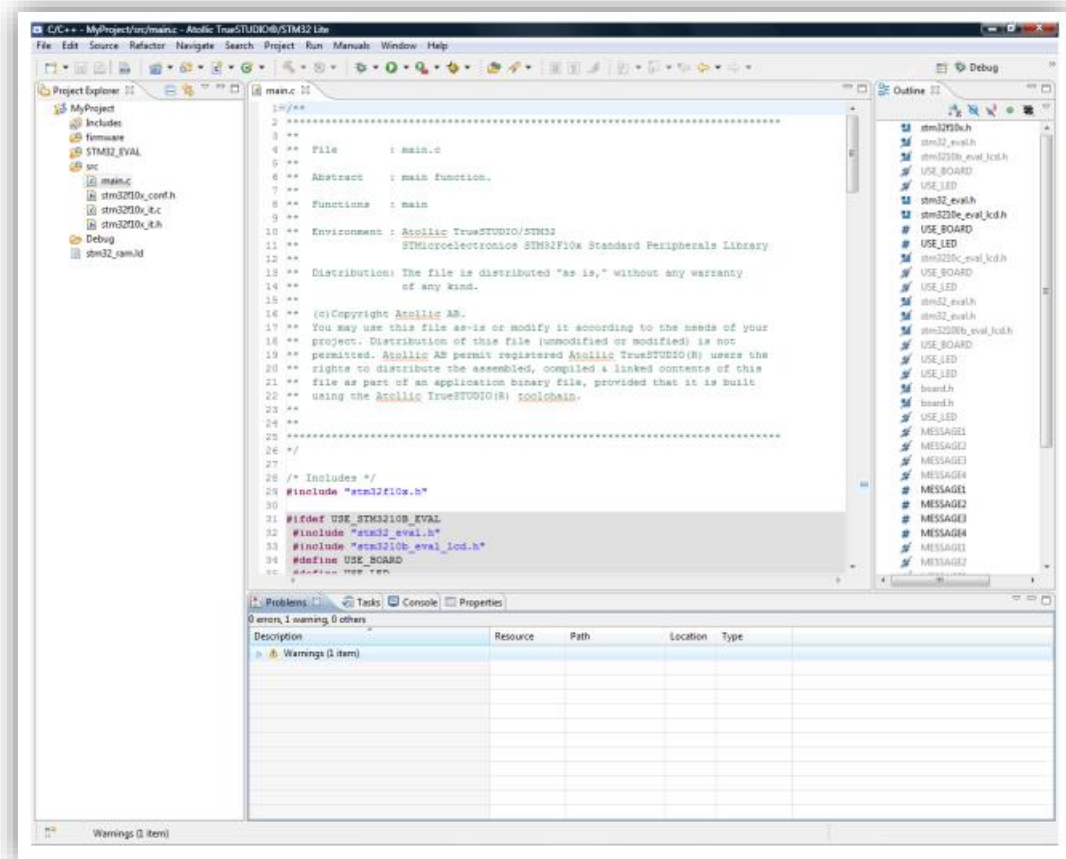
Free to download. Free to use. Free to share.

High-quality embedded systems software.

This includes compiling and assembling C/C++ source code, and linking object files and runtime libraries.

Core, multi-core and multi-processor systems.

Tight integration with popular debug probes including J-Link, ST-Link, and Multi-link.



	TrueSTUDIO Lite	TrueSTUDIO Pro
	FREE	49EUR/59USD per month
Features:	The best free ARM development tools on the planet!	Superior code analysis and advanced debug tools.
Project management and example projects	✓	✓
Edit/Compile/Debug	✓	✓
Tools for team collaboration	✓	✓
Unlimited code-size	✓	✓
Static code analysis tools		✓
Advanced debugging capabilities		✓
Technical support		✓

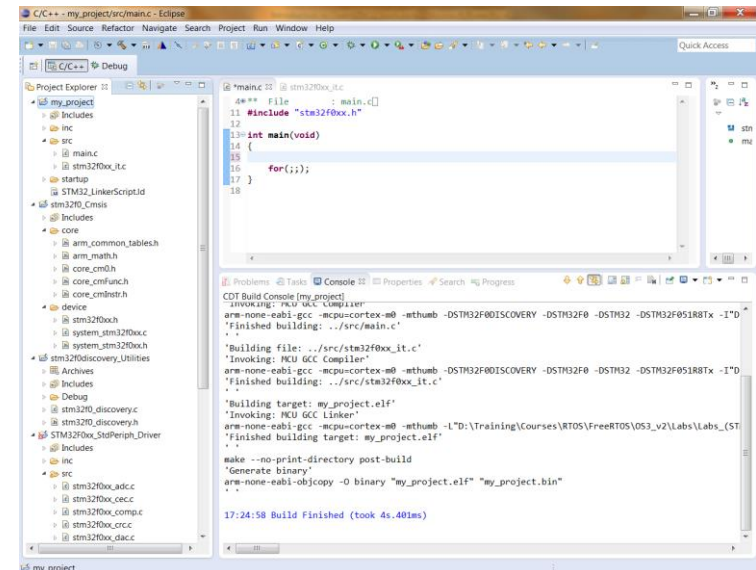
AC6 – SW4STM32

49



- Toolset developed by **AC6** partner
- Current name : « **System Workbench for STM32** » i.e **SW4STM32**
- A complete expandable free toolset :
 - A user-friendly graphical environment based on **Eclipse**
 - **GCC-based** full-fledged toolchains (without code size limitation)
 - **Multiple OS support** (Windows, Linux, OSX)
- A community website www.openSTM32.org :
 - Direct download area for toolset and add-ons
 - Wiki for toolset documentation
 - Support forums
 - Blogs

- Define your target platform from a predefined set of elements :
 - Choose any STM32 mcu configuration, any STM32 reference boards (Nucleo, Discovery, Eval) or define your own board.
- **Configure, compile** and install your application :
 - Use the automatically linked **libraries** providing **CMSIS**, Peripherals and Board related drivers.
- **Flash and Debug** your application on the target with built-in or external hardware debug probe :
 - Run, break, single step through your whole program.
 - Display MCU-specific registers in a convenient way.



Free CooCox CoIDE

CooCox[®]

Free/open ARM Cortex-M Development Tool-chain

Free CooCox CoIDE

- CooCox is a **free integrated development environment** focusing on **ARM Cortex Mx** based MCU.
- It uses **GCC** tool chain and is based on **Eclipse**.
- CooCox is devoted to providing developers with a free/open and easy-to-use ARM Cortex Mx tool-chain, including:
 - **CoLinkEx**, a low-cost and hardware open debug adapter (compatible with [ST-LINK-v2](#))
 - **CoIDE**, an integrated development environment for developing based on code components like stacking up building blocks
 - **CoSmart**, a graphical code generation tool.
- Meanwhile, CooCox encourages users to share open source code components in CooCox community. By providing a series of open source software such as CoX, a unified standard peripheral library for ARM Cortex Mx based microcontrollers.
- **CoOS**, a lightweight **real-time operating system** and a number of drivers based on CoX, CooCox is committed to building an open-source and sharing community.

Free MDK-ARM



Free MDK-ARM

- Multi-year agreement with ARM on MDK-ARM :
- **Free MDK-ARM licenses for STM32 based on Cortex-M0/M0+ cores :**
 - Applicable immediately to all **STM32F0** and **STM32L0** mcus.
 - PC-locked renewable 3-Year licenses.
 - **No code size limit.**
 - Windows support only.
 - Multiple languages supported : English, Japanese, Korean.
 - **Technical support included.**
- Direct download from Keil website :
 - **No limit of number of downloads by customer.**
 - Direct access to configuration files for all STM32 based on Cortex-M0/M0+ cores.
 - **Free access to MDK-ARM periodic updates.**



Thank you !