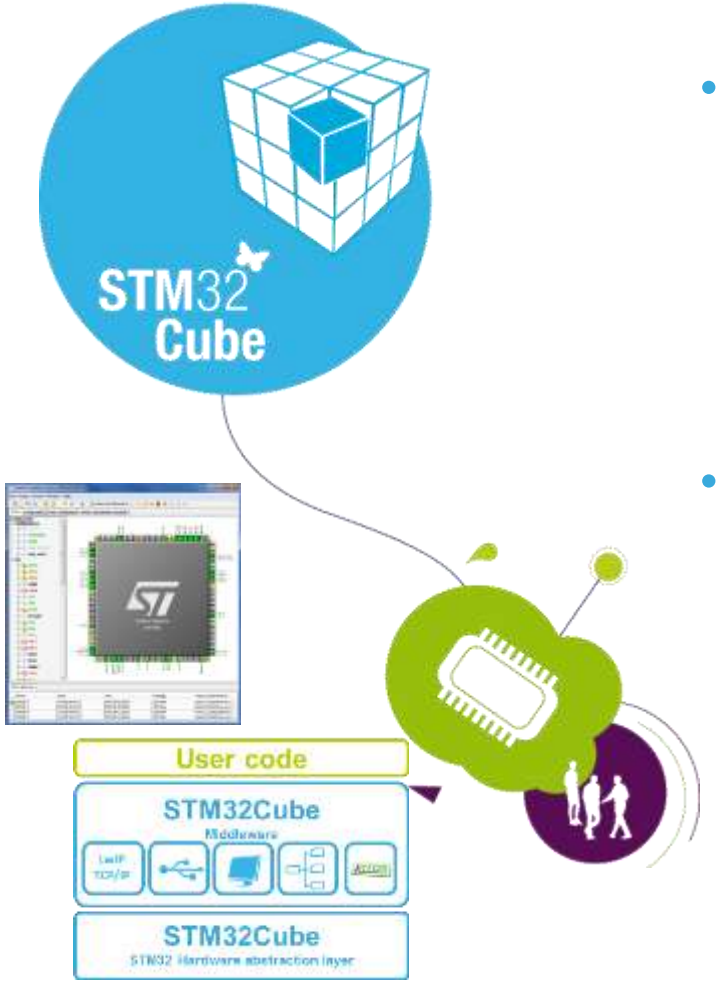
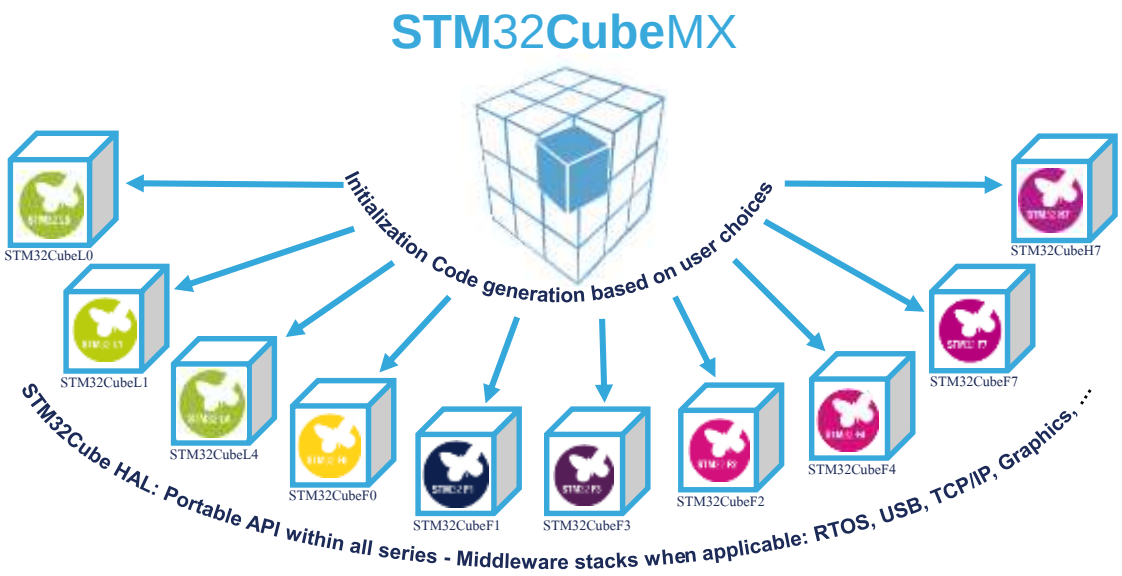
STM32Cube Overview

life.augmented

STM32Cube™ Introduction



- STMCube™ is an STMicroelectronics original initiative to ease developers life
  - By reducing development efforts
  - By reducing development time
  - By reducing development cost, with free solutions
- STM32Cube™ applies on STM32 portfolio

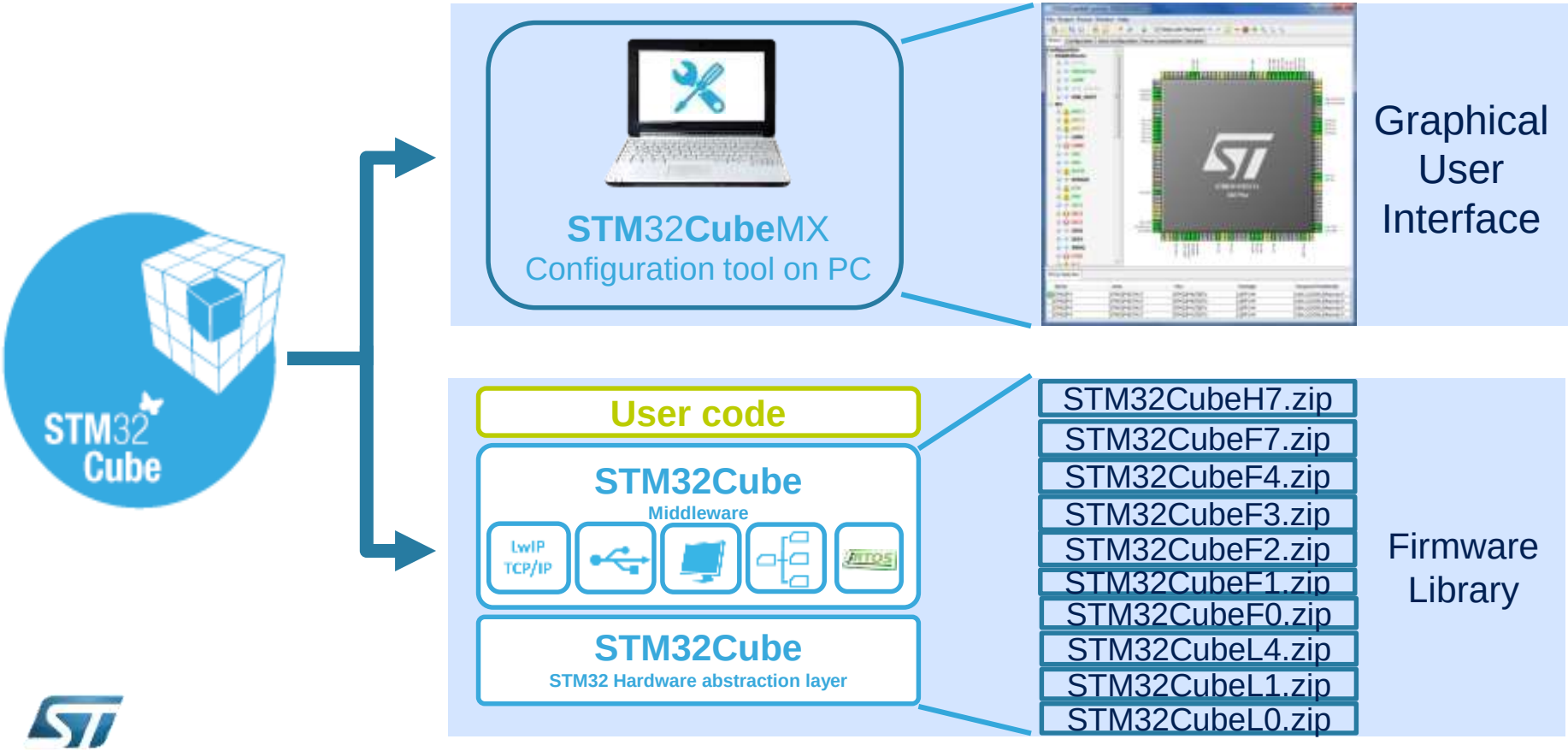


www.st.com/stm32cube

life.augmented

STM32Cube™ Overview

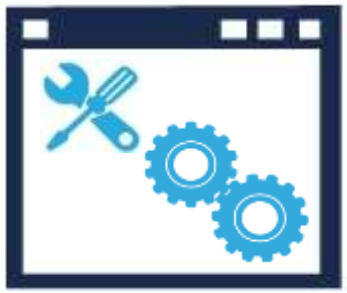
- STM32Cube™ is a software development platform that combines
  - A *PC software configuration* tool called **STM32CubeMX**
  - *STM32 embedded software* bricks called **STM32CubeFx/Lx**



life.augmented

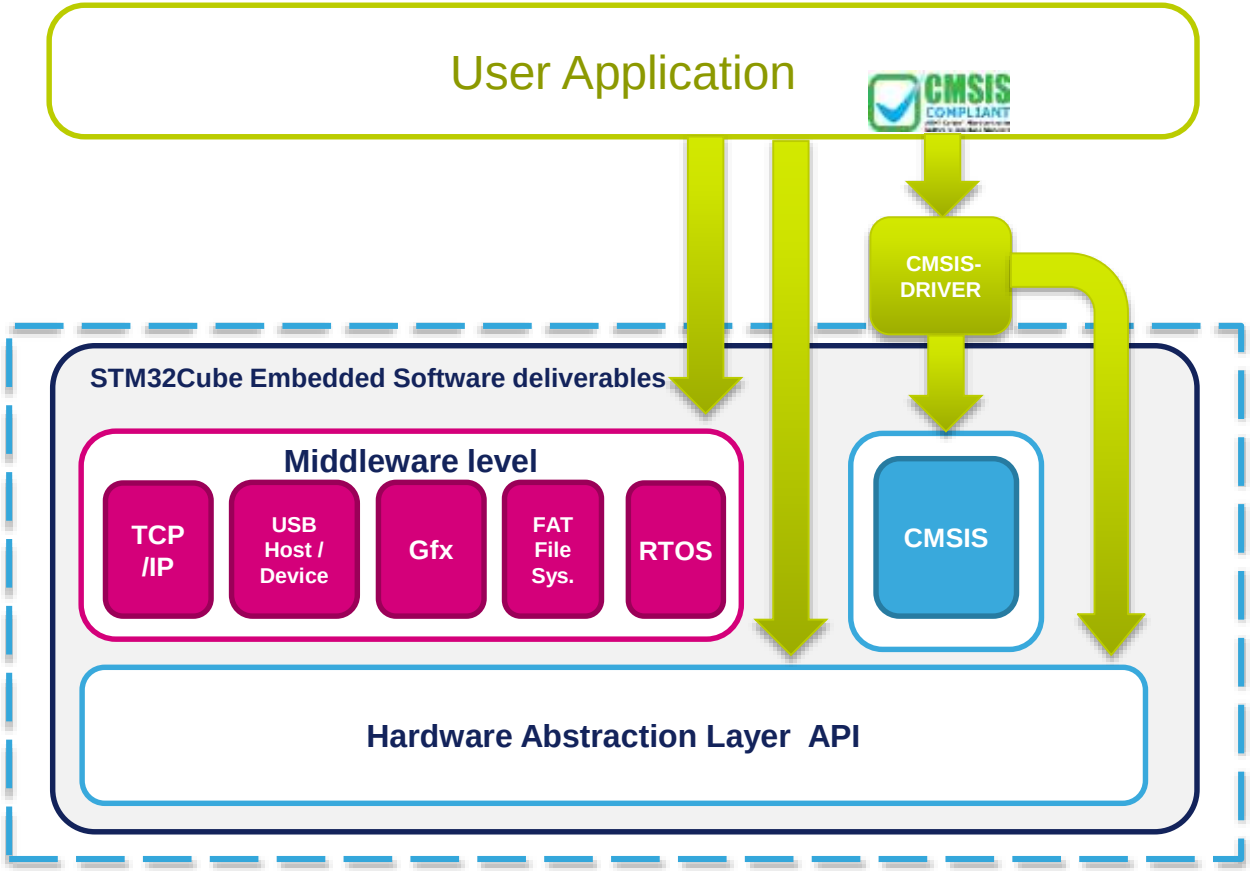
STM32Cube Work Flow

STM32CubeMX
Configuration tool on PC

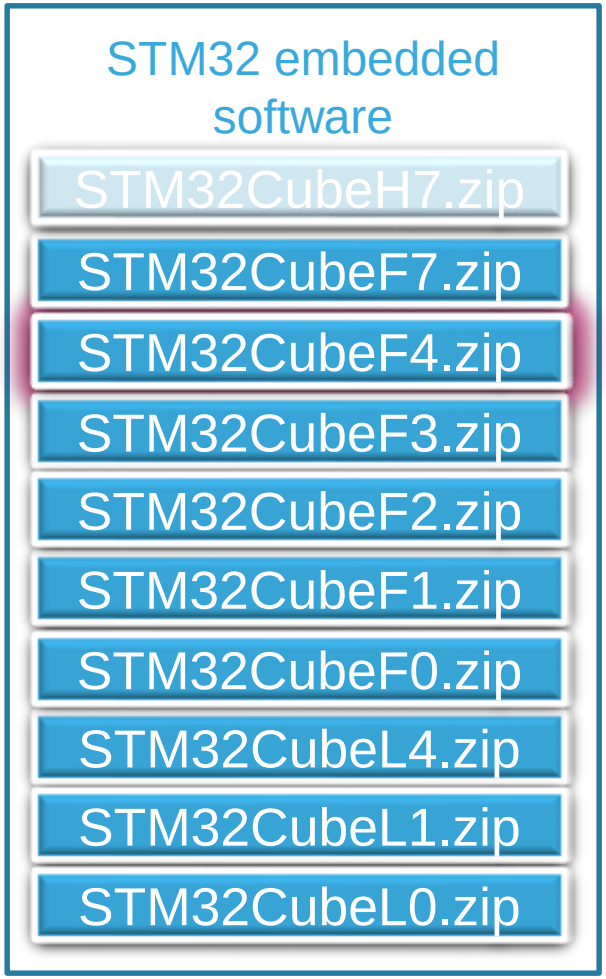


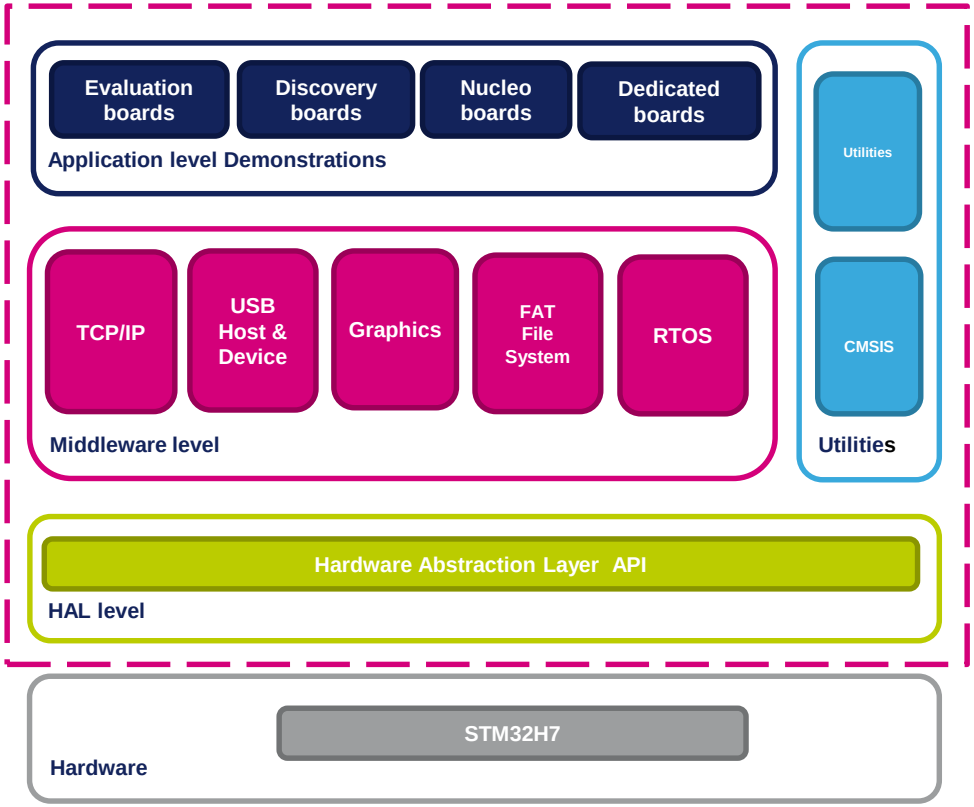
STM32F0	STM32F1	STM32F2
STM32F3	STM32F4	STM32F7
STM32H0	STM32L0	STM32L1
STM32L4		

C code generation for initialization, depending on user choices



STM32CubeFx/Lx FW Package

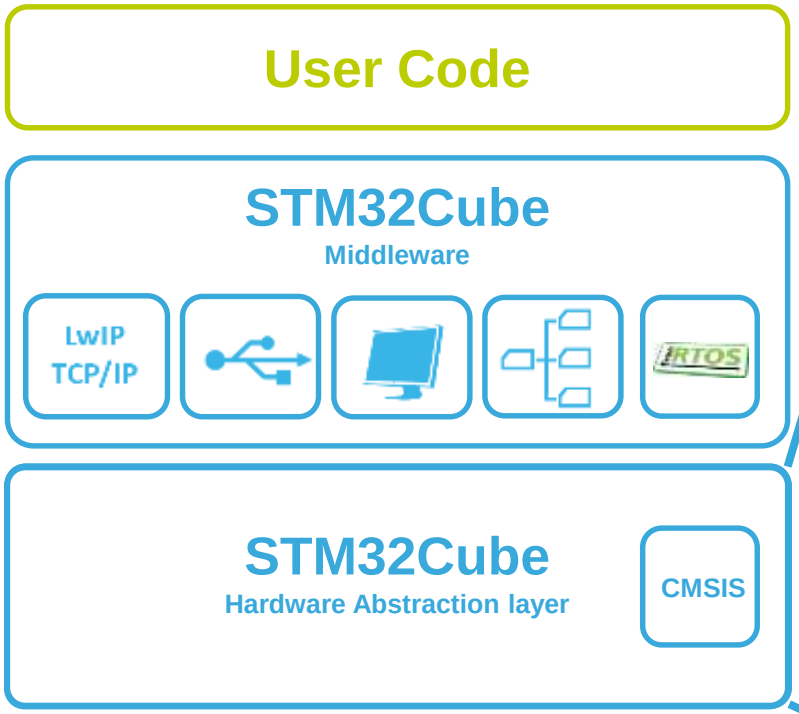




Downloadable manually from www.st.com/stm32cube
or via ***STM32CubeMX*** download libraries menu

STM32Cube V1

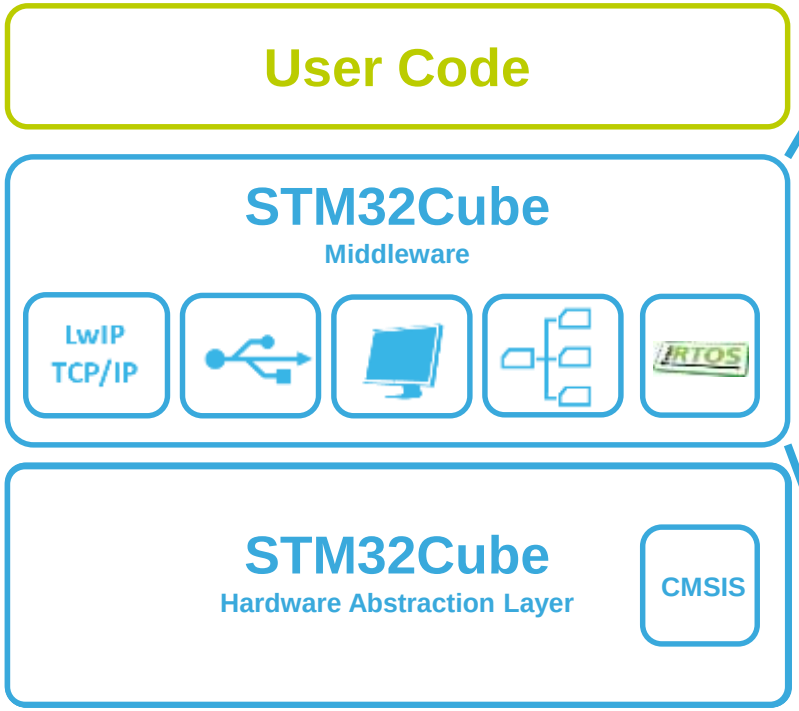
STM32CubeFx/Lx FW Package



- Abstraction of STM32
  - Through portable APIs
- High STM32 coverage
  - Most of peripherals covered
- Production Ready
  - Quality: CodeSonar™
- Complete
  - ***>150 peripheral examples!***
- Permissive terms
  - Open source BSD license

STM32Cube V1

STM32CubeFx/Lx FW Package

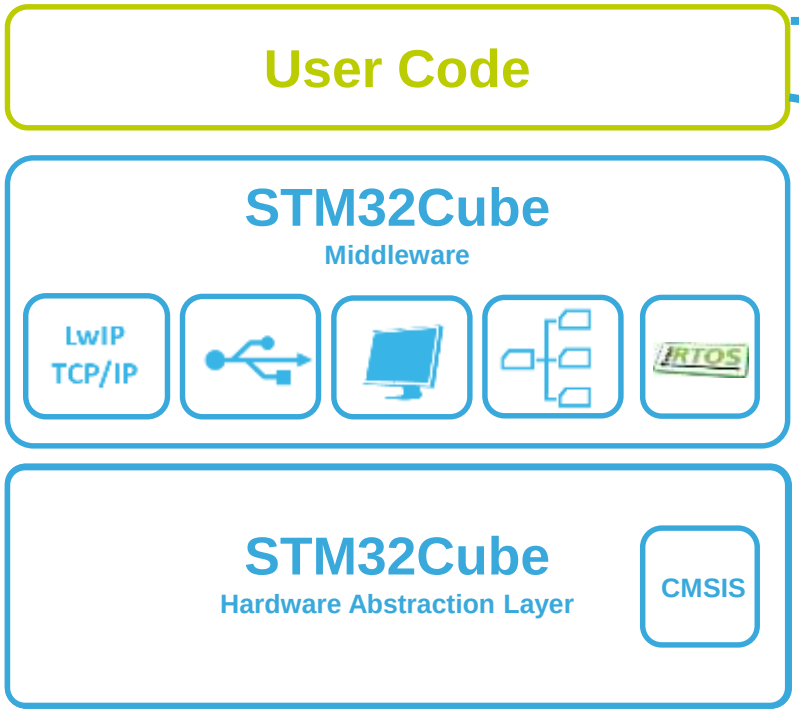


- TCP/IP stack
  - LwIP open source standard
- USB Library
  - Host & Device made by ST
- Graphics
  - STemWin from ST and SEGGER
- File System
  - FatFS open source standard
- RTOS
  - FreeRTOS open source standard (with CMSIS-RTOS abstraction)
- ***>40 examples !***

life.augmented

STM32Cube V1

STM32CubeFx/Lx FW Package

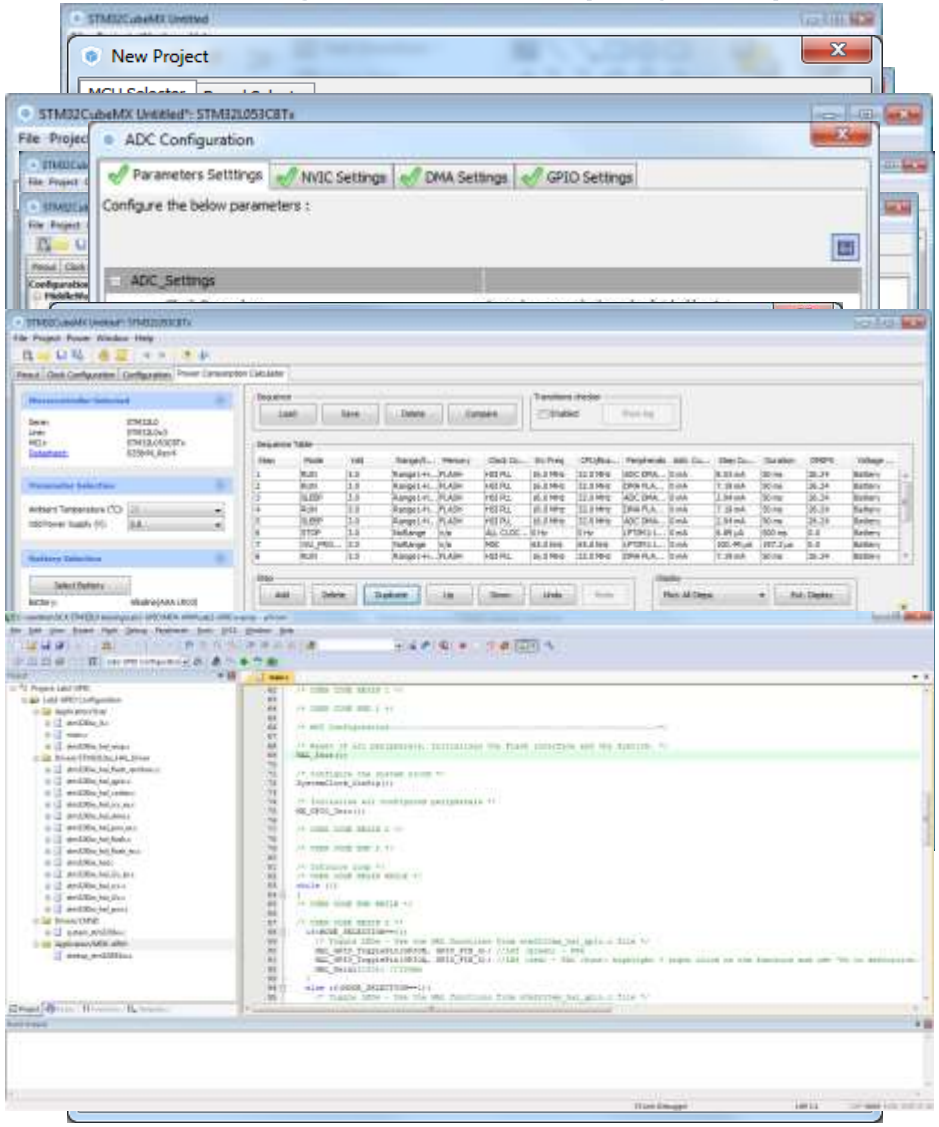


- Customer focusing on his application differentiation

STM32Cube V1 – STM32CubeMX

Microcontroller configuration, step by step

- STM32Cube includes the *STM32CubeMX* which is a graphical software configuration tool that allows generating C initialization code using graphical wizards.

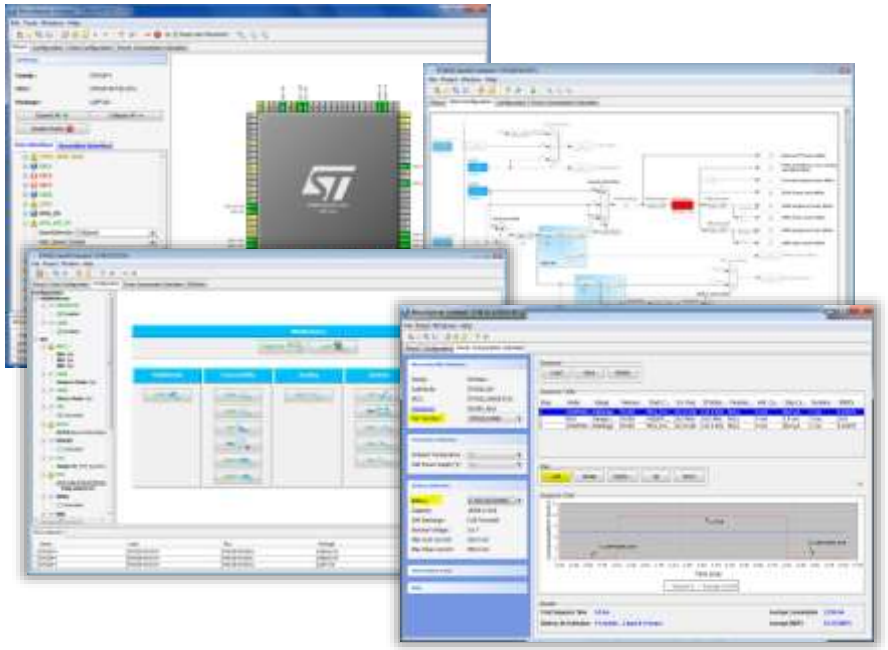


- **Step 1**: Select the microcontroller
  - Through easy filtering capabilities
- **Step 2:** Configure the microcontroller
  - Pin out wizard
  - Clock tree wizard
  - Peripherals and Middleware wizards
  - Power consumption wizard
- **Step 3**: Initialization code generation
  - Generates code for your favorite IDE !

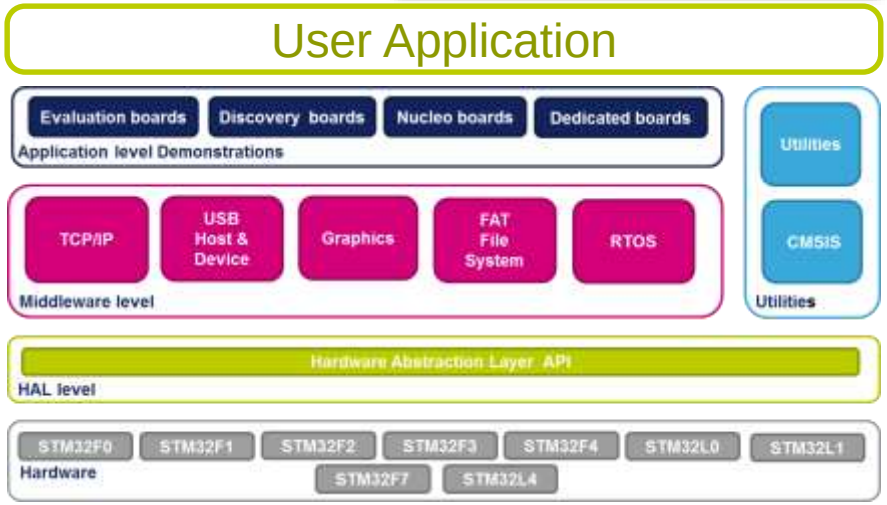
life.augmented

STM32Cube™ V1 – Key Benefits

- **Simplify and Speed up Application Development for STM32!**
  - Through STM32CubeMX:
    - MCU Selector
    - Graphical Peripheral Configuration
    - Power Consumption Wizard
    - Peripheral Initialization Code Generation
    - With automatic updater feature
      - Ensuring the developer is aware of new versions and fixes, as well as new components
  - Through extensive set of "*ready-to-run*" peripheral examples and application examples, with ready project files for IAR, Keil and GCC included in the STM32CubeFx/Lx packages



- **More than Cost-friendly !**
  - **100% FREE** embedded software!
  - **100% FREE** software tool !
  - ST-branded, ST-supported !
  - **Users gain time** with initialization code generation, and remain focused on their key application code



www.st.com/stm32cube

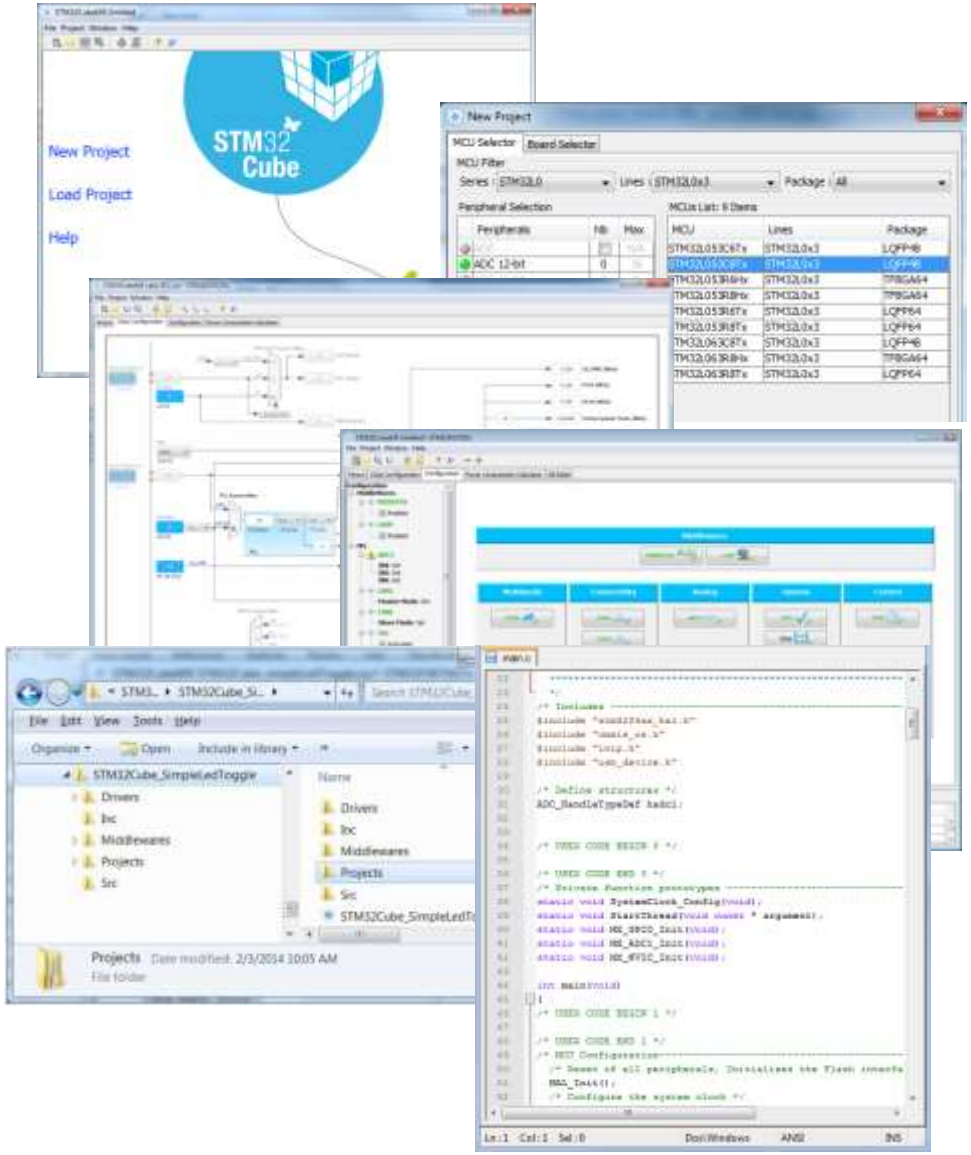
life.augmented

STM32CubeMX Overview

STM32CubeMX for STM32 configuration and initialization C code generation

- The STM32CubeMX, a graphical software configuration tool that allows to generate C initialization C code using graphical wizards.
- STM32CubeMX has the following key features:
  - Easy microcontroller selection covering whole STM32 portfolio.
  - Board selection from a list of STMicroelectronics boards.
  - Easy microcontroller configuration (pins, clock tree, peripherals, middleware) and generation of the corresponding initialization C code.
  - Generation of configuration reports.
  - Generation of IDE ready projects for a selection of integrated development environment tool chains.

  STM32CubeMX projects include the generated initialization C code, STM32 HAL drivers, the middleware stacks required for the user configuration, and all the relevant files needed to open and build the project in the selected IDE.
  - Power consumption calculation for a user-defined application sequence.
  - Self-updates allowing the user to keep the STM32CubeMX up-to-date.
  - Downloading and updating STM32Cube™ firmware packages allowing the download from www.st.com of the MCU firmware package required for the development of the user application



STM32CubeMX C Code generation overview

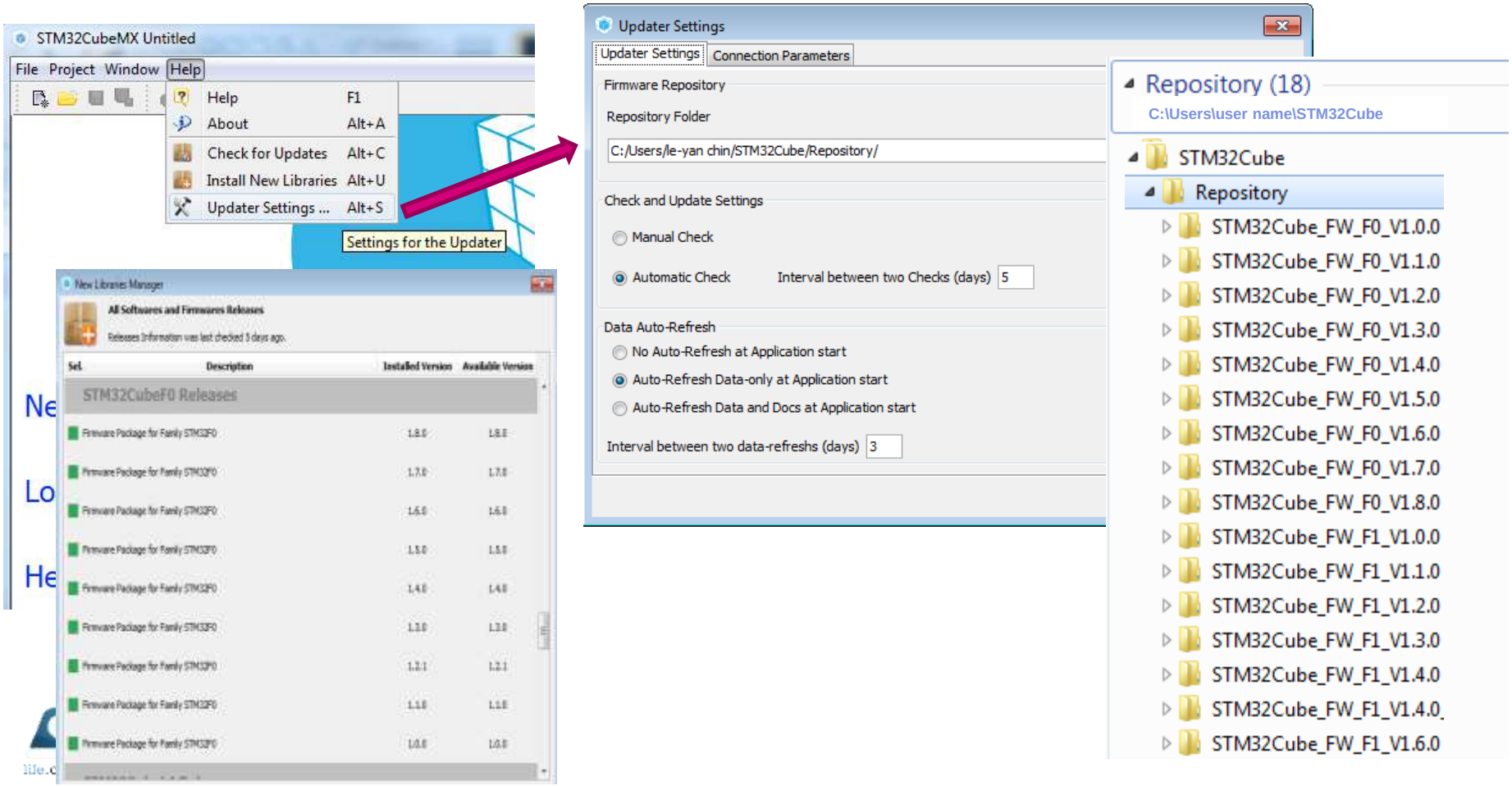
- During the C code generation process, STM32CubeMX performs the following actions:
  - It downloads the relevant STM32Cube firmware package if it is missing from the STM32CubeMX repository.
  - It copies from the firmware package, the relevant files in Drivers/CMSIS and Drivers/STM32xx_HAL_Driver folders and in the Middleware folder if a middleware was selected.
  - It generates a *Projects* folder that contains the toolchain specific files that match the user project settings.
  - It generates the initialization C code (.c/.h files) corresponding to the user MCU configuration and stores it in the Inc and Src folders. By default, the following files are included:

Files	Description
stm32f4xx_hal_conf.h	this file defines the enabled HAL modules and sets some parameters (e.g. External High Speed oscillator frequency) to pre-defined default values or according to user configuration (clock tree).
stm32f4xx_hal_msp.c (MSP=MCU Support package)	this file defines all initialization functions to configure the IP instances according to the user configuration (pin allocation, enabling of clock, use of DMA and Interrupts).
main.c	is in charge of: - Resetting the MCU to a known state by calling the HAL_init() function that resets all peripherals, initializes the Flash memory interface and the SysTick. - Configuring and initializing the system clock. - Configuring and initializing the GPIOs that are not used by IPs. - Defining and calling, for each configured IP, an IP initialization function that defines a handle structure that will be passed to the corresponding IP HAL init function which in turn will call the IP HAL MSP initialization function.

life.augmented

STM32CubeMX Repository

- Downloaded software and firmware releases will be stored in the Repository folder. The Default folder is defined in STM32CubeMX->Help->Updater Settings->Repository Folder.



STM32CubeMX Documentation

Related Tools and Software

Related Tools and Software	
Part Number	Descript
STM32CubeF3	Embedde RTOS, To STM32 N
STM32CubeF1	Embedde system, Nucleo, D
STM32CubeL1	Embedde RTOS, To STM32 N
STM32CubeF7	Embedde system, kit and E
STSW-STM32095	STM32Cu eclipse p
STM32CubeL0	Embedde RTOS, To Nucleo, D
STM32CubeF2	Embedde system,
STM32CubeF4	Embedde system, Nucleo, D
STM32CubeF0	

Technical Documentation

Product Specifications		
Description	Version	Size
DB2163: *STM32 configuration and initialization C code generation*	3.0	250 KB

Technical Notes & Articles		
Description	Version	Size
TN0072: *Software toolchains and STM32 features*	2.4	98 KB

User Manual		
Description	Version	Size
UM1718: *STM32CubeMX for STM32 configuration and initialization C code generation*	8.0	12,438 KB

Release Notes		
Description	Version	Size
RN0094: *STM32CubeMX release 4.8.0*	11.0	245 KB

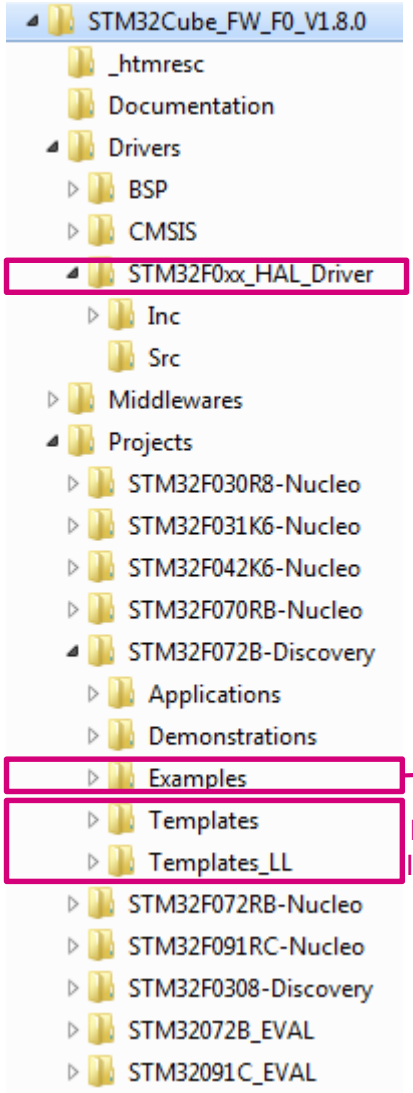
Links to various STM32Cube offering

life.augmented

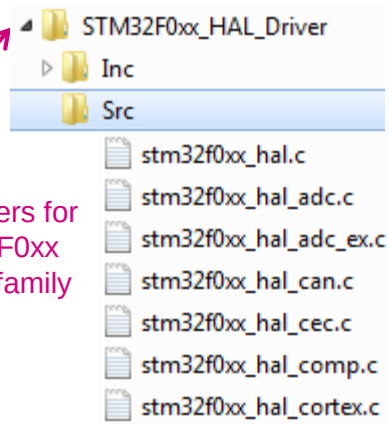
http://www.st.com/stm32cube

STM32Cube: STM32CubeF0 Firmware Package

- Browse to
- C:\users\your name\STM32Cube\Repository\STM32Cube\STM32Cube_FW_V1.8.0 or later

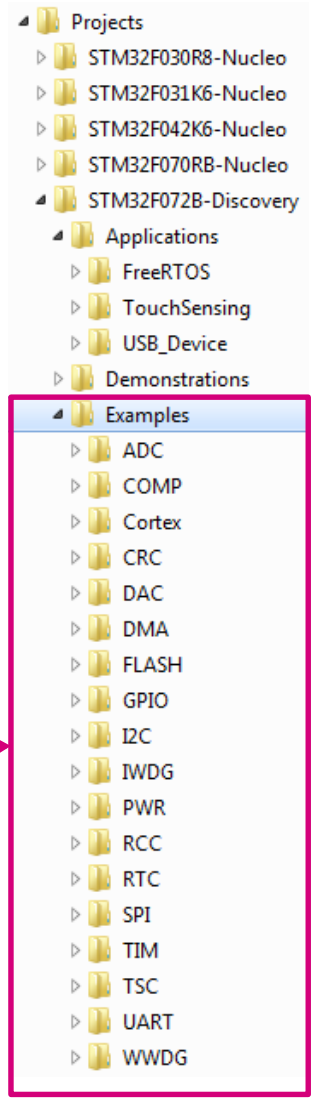


HAL drivers for STM32F0xx product family



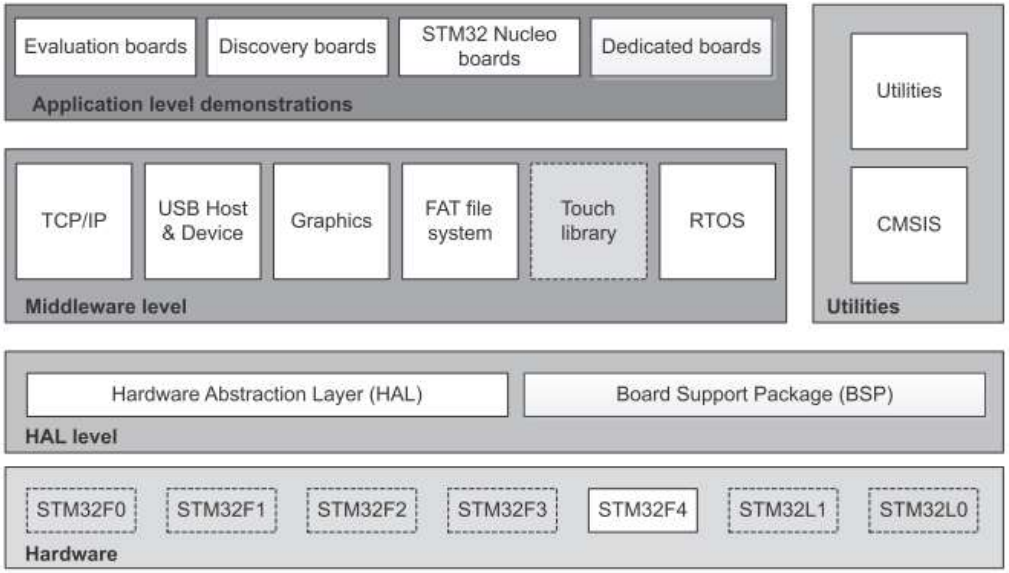
HAL examples for STM32F072 Discovery board

Project templates for IAR, Keil and System Workbench



STM32Cube FW Package Features

- STM32Cube gathers together, in a single package, all the generic and highly portable embedded software components required to develop an application on STM32 microcontrollers.
- The package includes a low level hardware abstraction layer (HAL) that covers the microcontroller hardware, together with an extensive set of examples running on STMicroelectronics boards.
- It also contains a set of middleware components **(*)** with the corresponding examples. They come with very permissive license terms:
  - Full USB Host and Device stack supporting many classes.
    - Host Classes: HID, MSC, CDC, Audio, MTP
    - Device Classes: HID, MSC, CDC, Audio, DFU
  - Graphics
    - STemWin, a professional graphical stack solution available in binary format and based on the emWin solution from ST's partner SEGGER
    - LibJPEG, an open source implementation on STM32 for JPEG images encoding and decoding.
  - CMSIS-RTOS implementation with FreeRTOS open source solution
  - FAT File system based on open source FatFS solution
  - TCP/IP stack based on open source LwIP solution
  - SSL/TLS secure layer based on open source PolarSSL
- A demonstration implementing all these middleware components is also provided



(*) middleware components availability depends on STM32 Series

STM32CubeF0 Documentation

- The STM32Cube documentation vary from one STM32 series to another.
- More high-end MCUs(e.g. STM32F4) are supported by more middleware libraries, like for STemWin graphics library and LwIP TCP/IP Stack.

Product Specifications		
Description	Version	Size
DB2347: STM32Cube embedded software for STM32F0 series including HAL drivers, USB, File System, RTOS and Touch sensing	2.0	219 KB

Application Notes		
Description	Version	Size
AN4735: STM32Cube firmware examples for STM32F0 Series	2.0	281 KB

User Manual		
Description	Version	Size
UM1779: Getting started with STM32CubeF0 firmware package for STM32F0 series	4.0	387 KB
UM1785: Description of STM32F0xx HAL drivers	2.0	10,604 KB
UM1787: STM32CubeF0 Nucleo demonstration firmware	3.0	803 KB
UM1819: Demonstration firmware for STM32091C-EVAL board	1.0	4,247 KB
UM1913: Developing applications on STM32Cube with Touch Sensing	1.0	3,416 KB

http://www.st.com/web/en/catalog/tools/PF260612

life.augmented

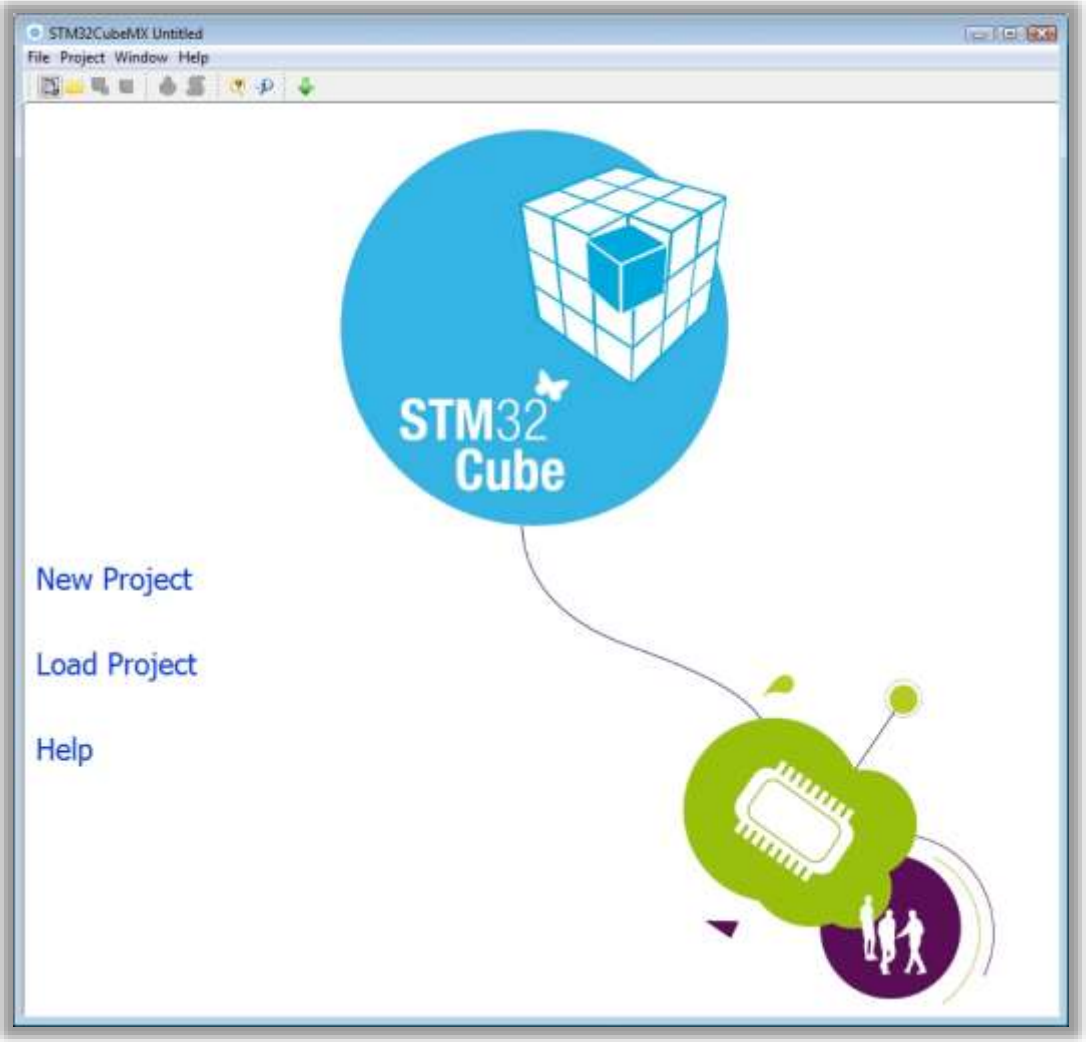
Demo: STM32CubeMX Overview

life.augmented

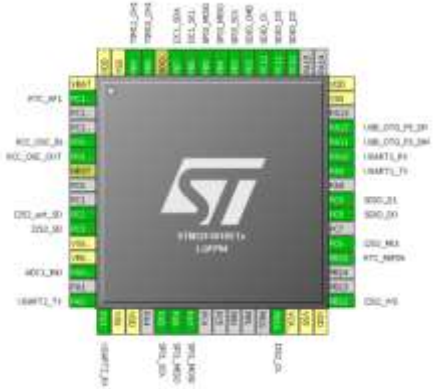
STM32Cube: STM32CubeMX

Step by step:

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



STM32CubeMX



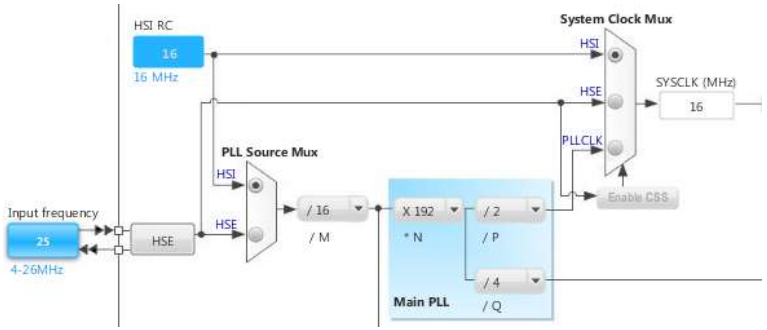
Pinout Wizard

Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

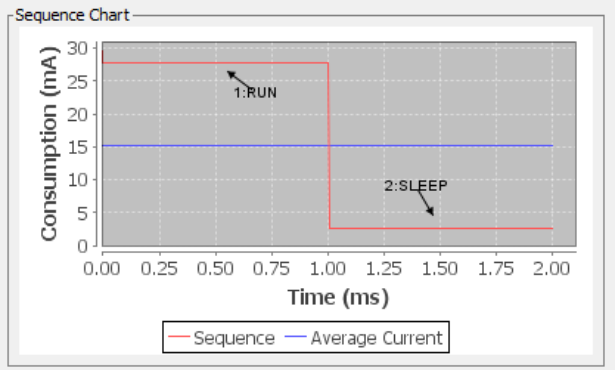
Baud Rate
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Peripherals & Middleware Wizard

Clock Tree wizard

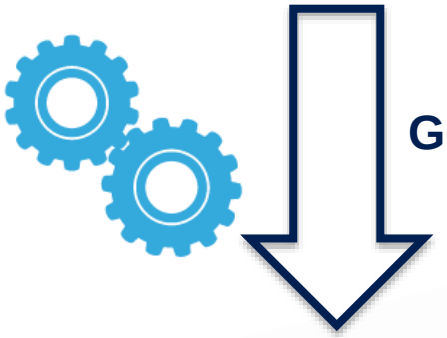


Power Consumption Wizard



life.augmented

STM32CubeMX

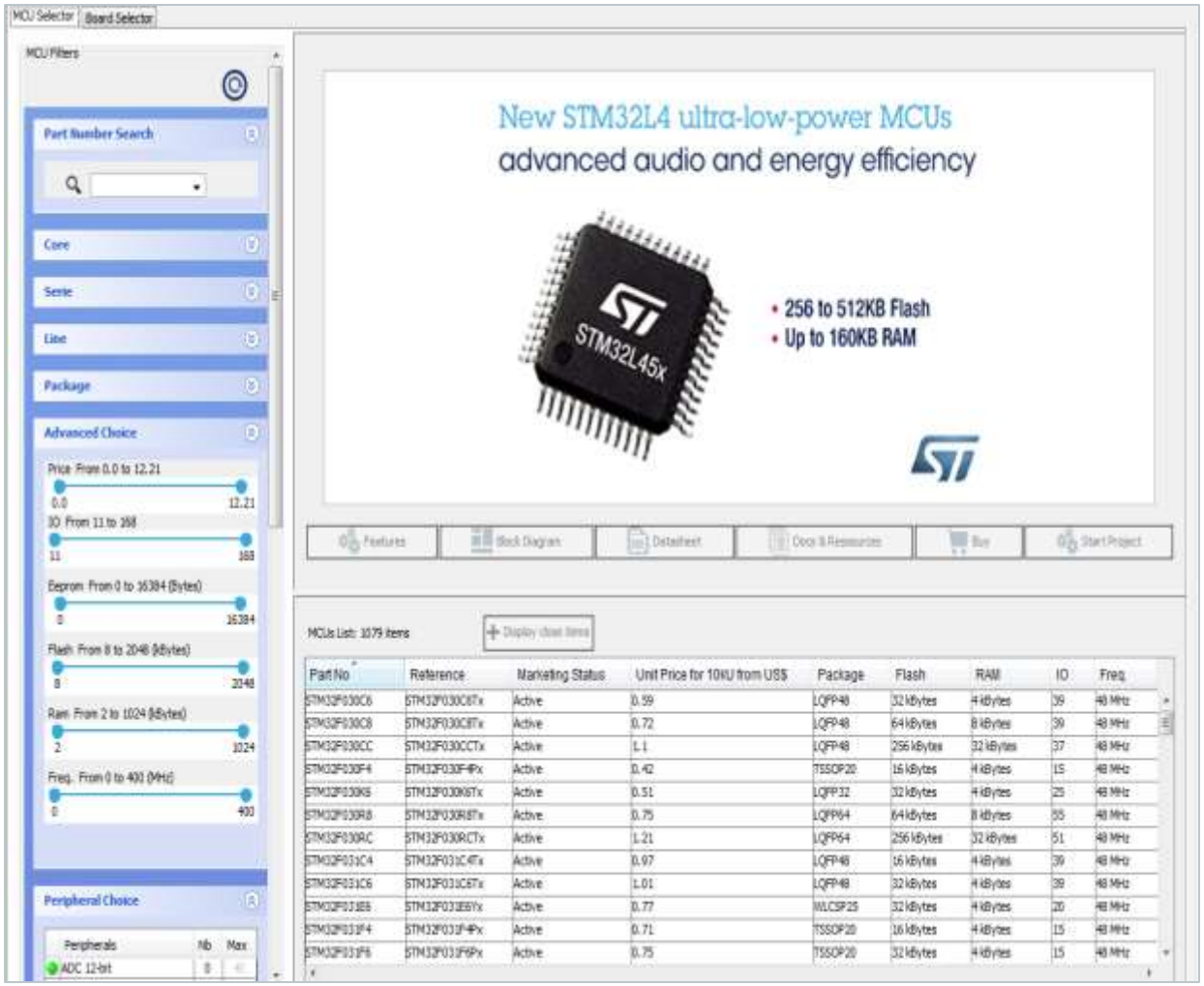


Generates Initialization C Code based on user choices !

STM32CubeMX: MCU Selector

Easy Optional filtering:

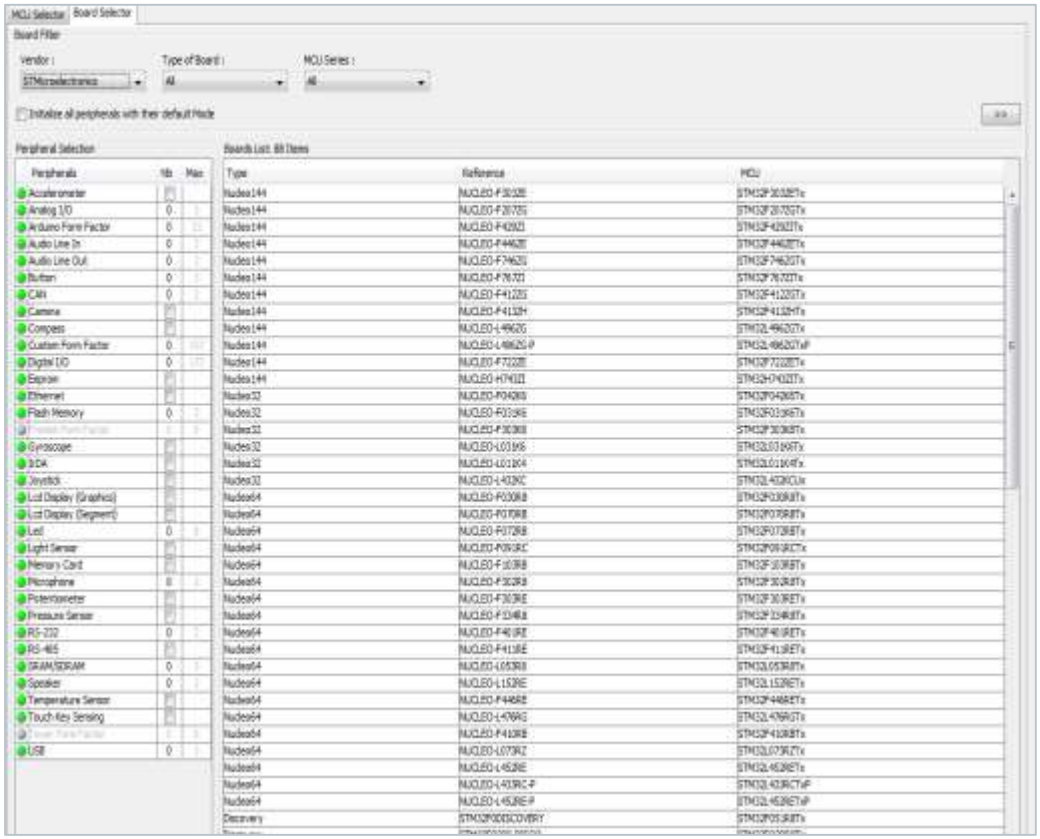
- Core
- Series
- Line
- Package
- Advanced choices…
- Peripherals choices…



life.augmented

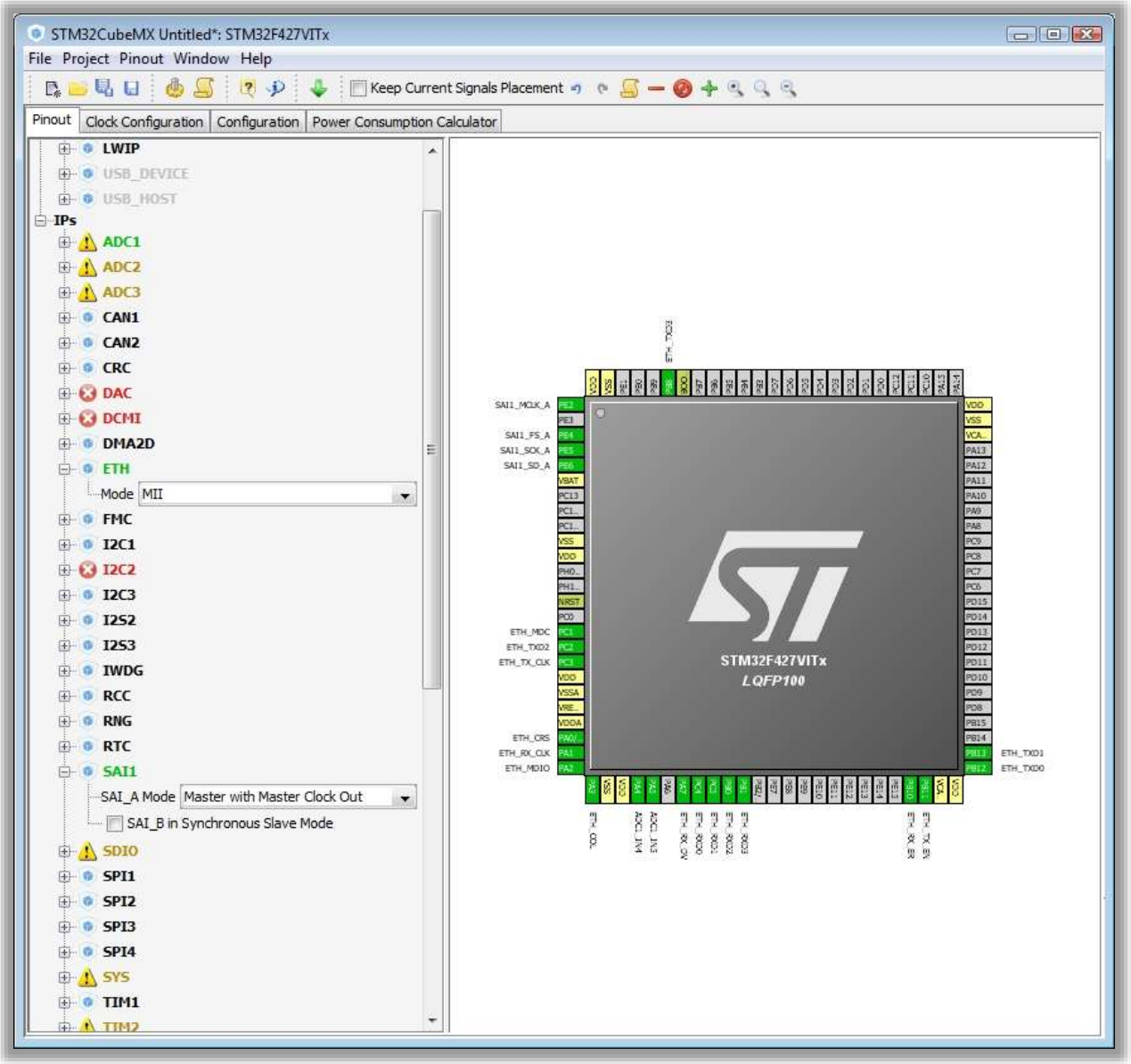
MCU selector continued

- Second tab provides shortcuts to predefined boards equipped with STM32 MCU.
- Predefined boards come with pinouts already assigned to use the connections and features of the particular board.
- Alternative board configurations are not covered.



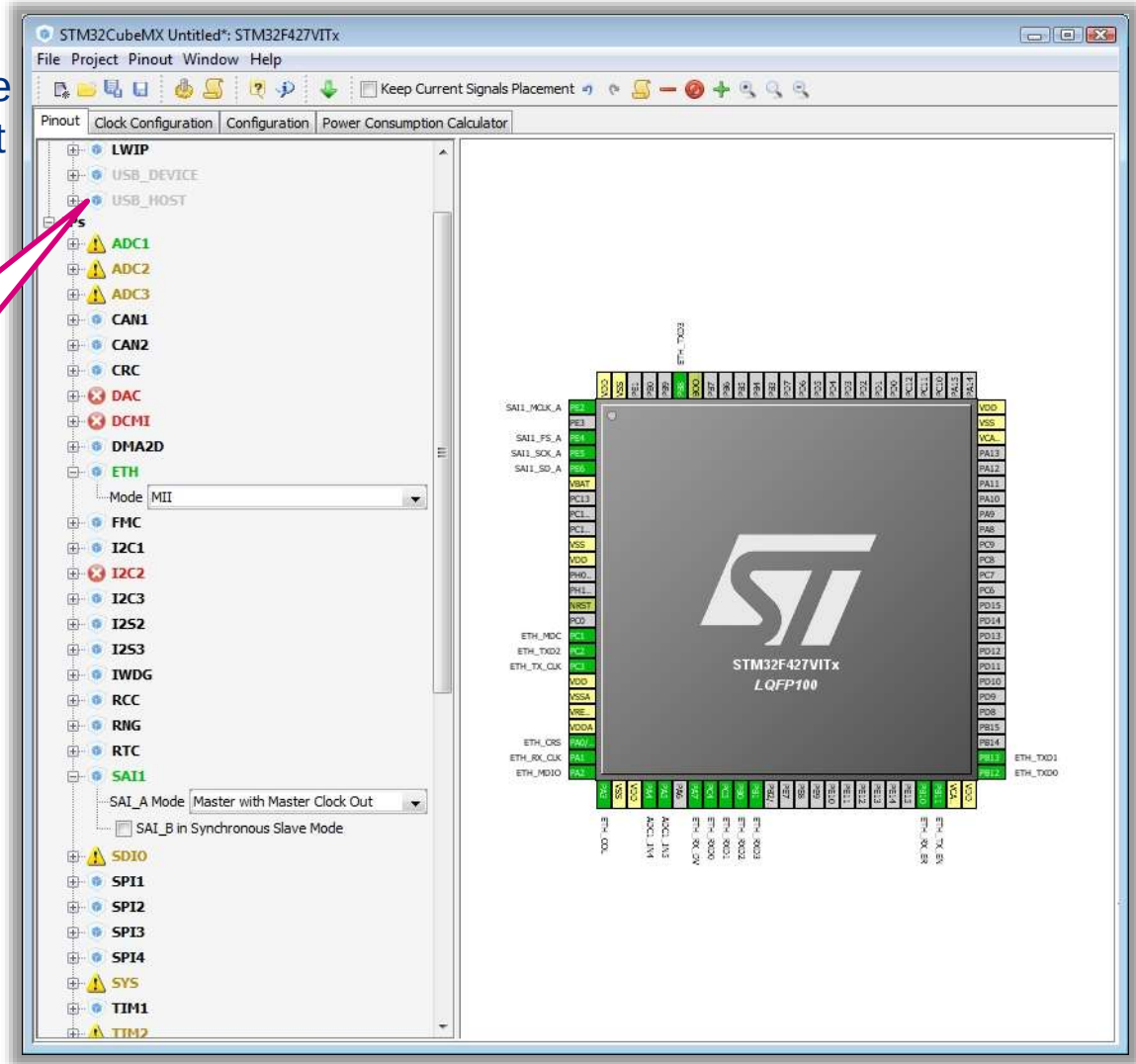
STM32CubeMX: Pinout configuration

- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals and/or middleware (FatFS, LWIP, FREERTOS, USB etc)



STM32CubeMX: Pinout configuration

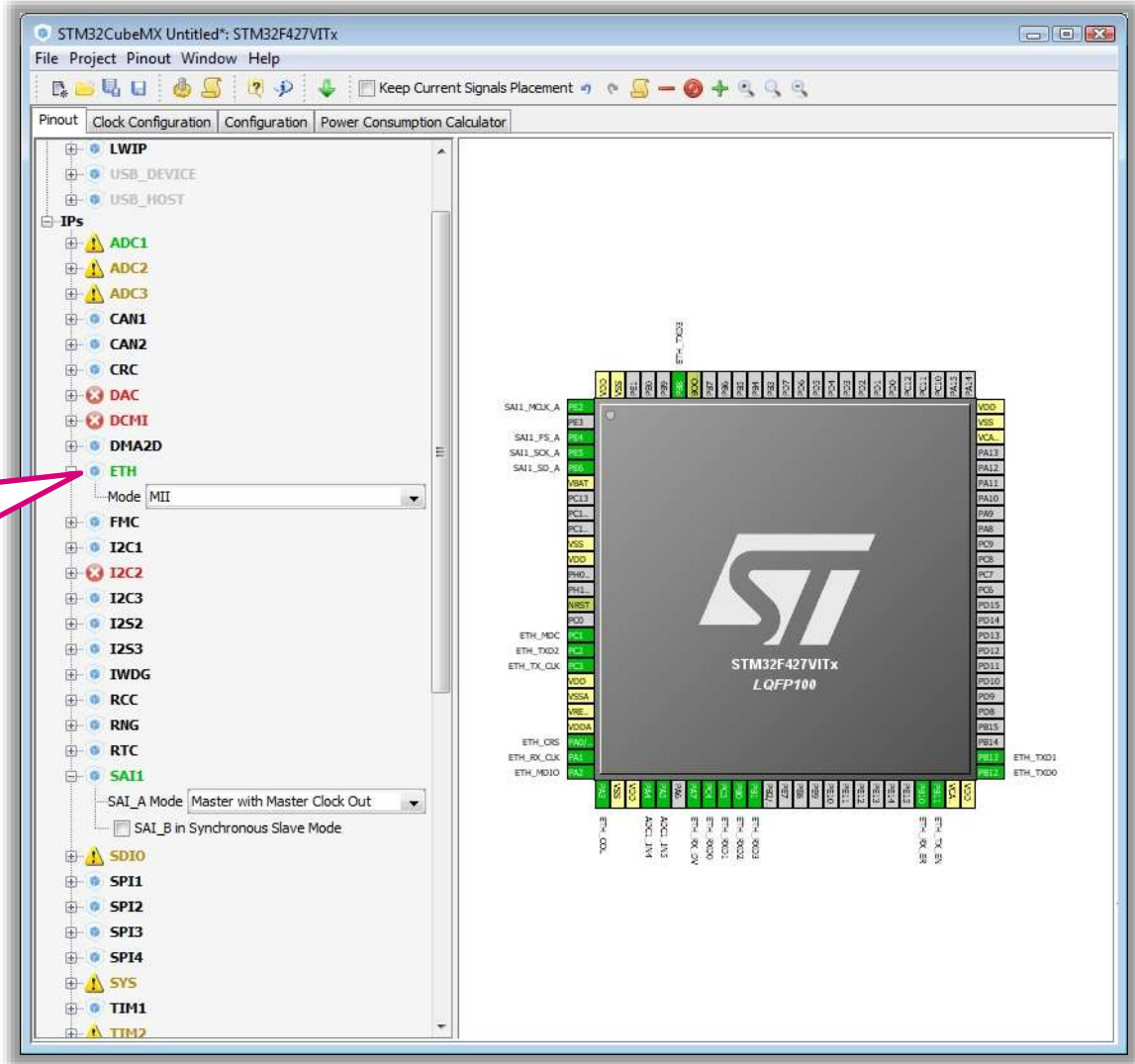
- Different possible states for a peripheral modes
  - Dimmed:
    the mode is not available because it requires another mode to be set (just put the mouse on top of the dimmed mode to see why)



Dimmed:
The additional periphery must be selected

STM32CubeMX: Pinout configuration

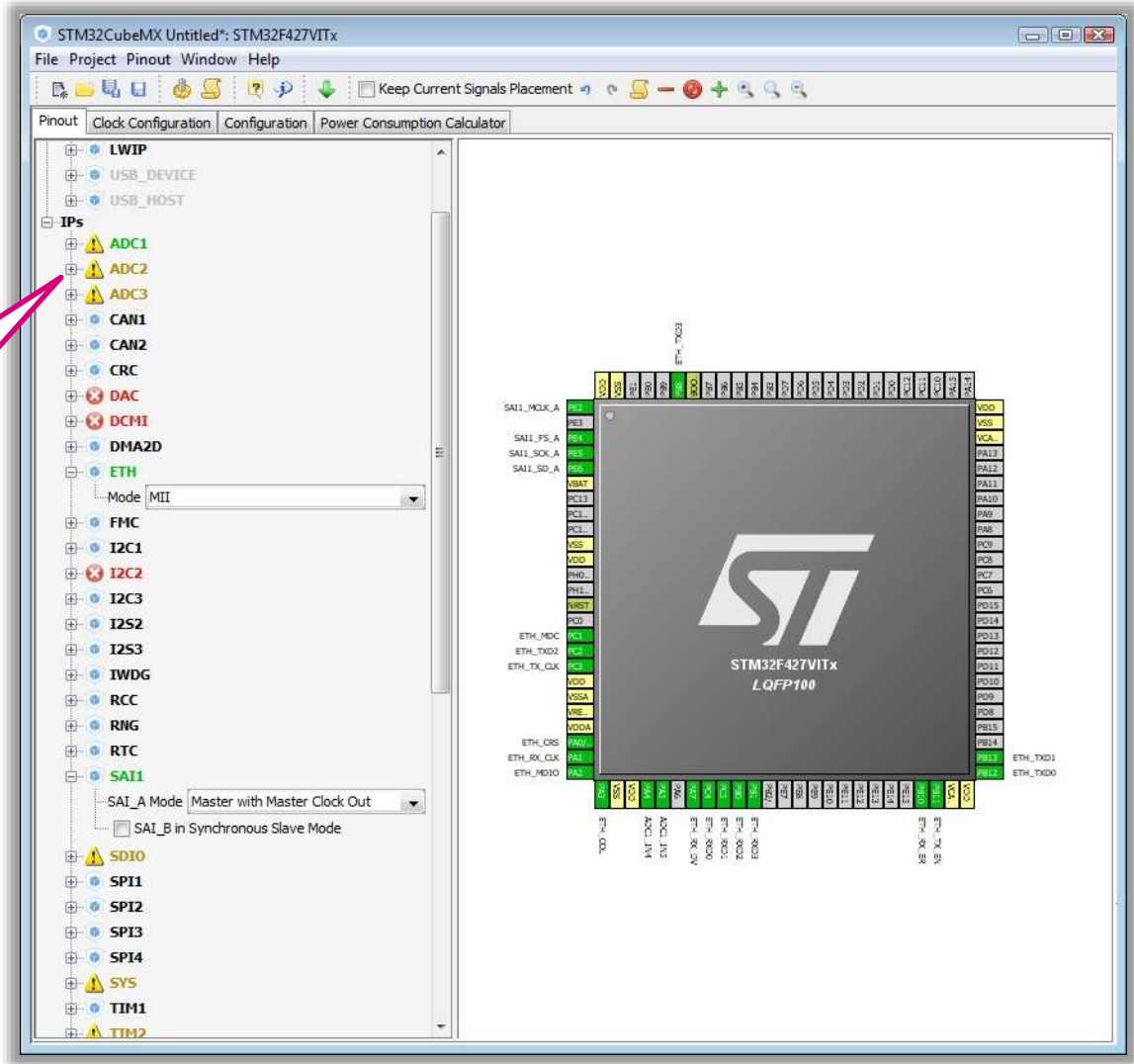
- Different possible states for a peripheral modes
  - Green:
    Periphery is assigned to pinout



Green:
Periphery will be functional

STM32CubeMX: Pinout configuration

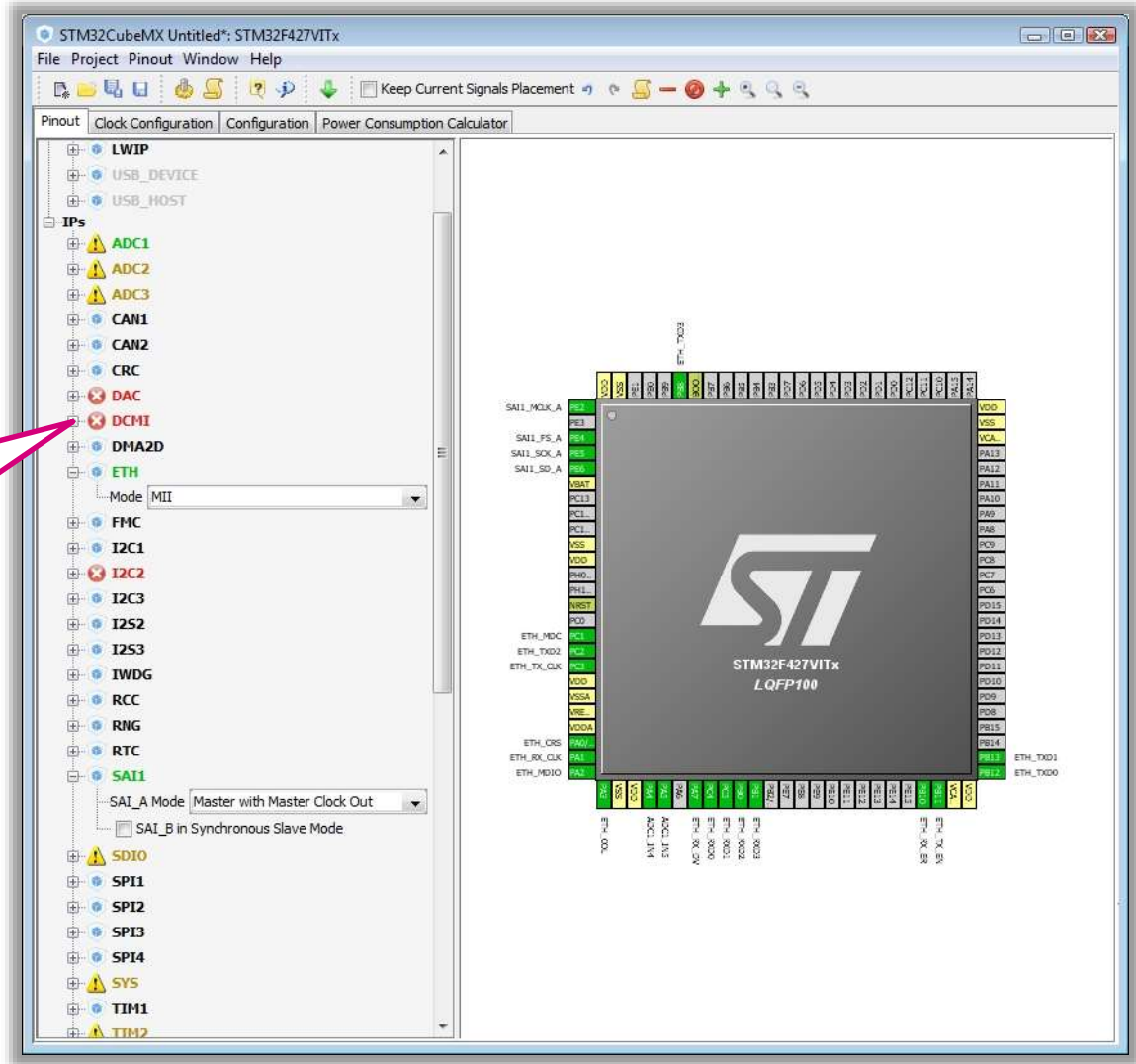
- Different possible states for a peripheral modes
  - Yellow:
    Only some functionalities of periphery can be used



Yellow:
On ADC only some channels can be used

STM32CubeMX: Pinout configuration

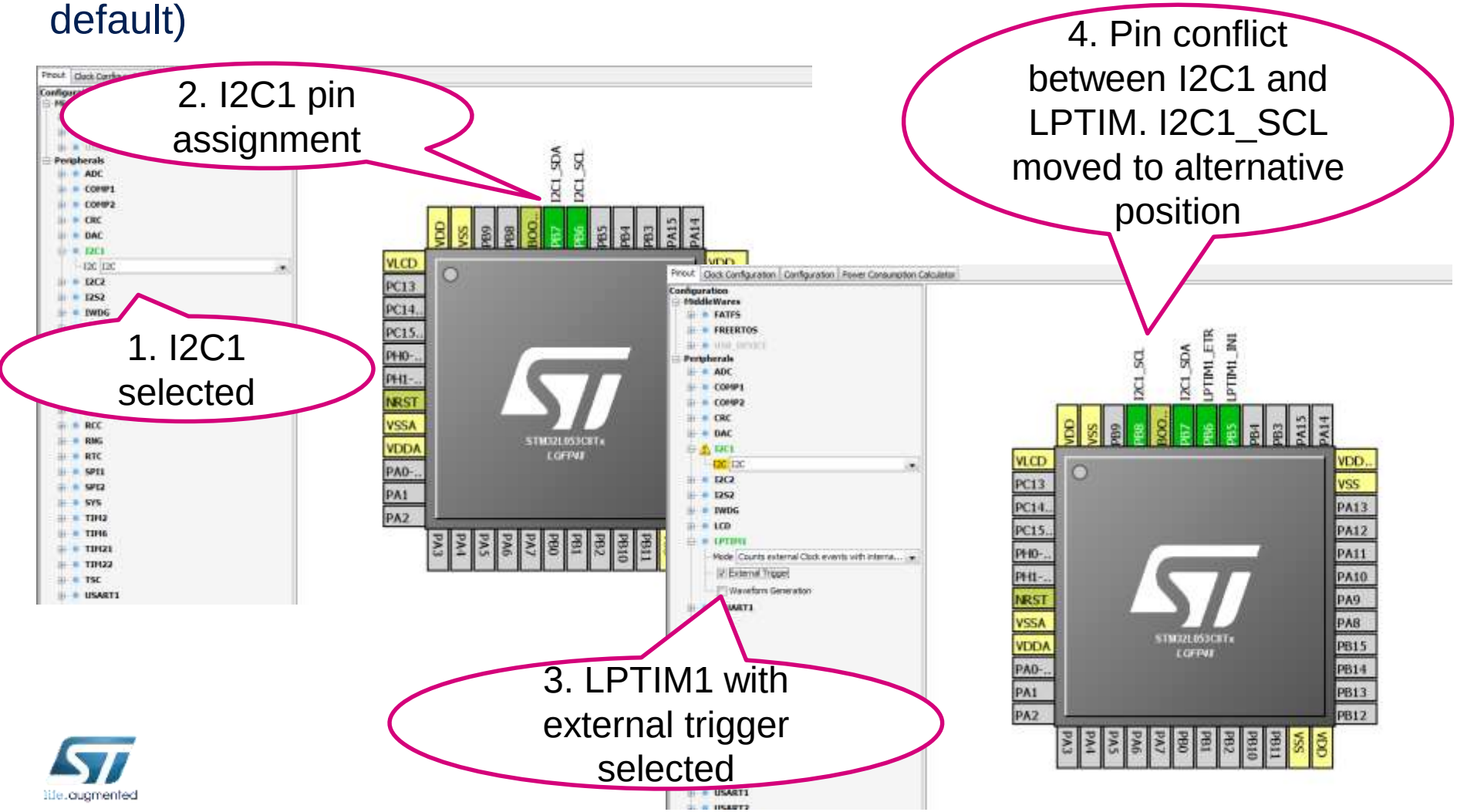
- Different possible states for a peripheral modes
  - Red:
    Signals required for this mode can't be mapped on the pinout (see tooltip to see conflicts)



Red:
Periphery cannot be used in this pinout setup

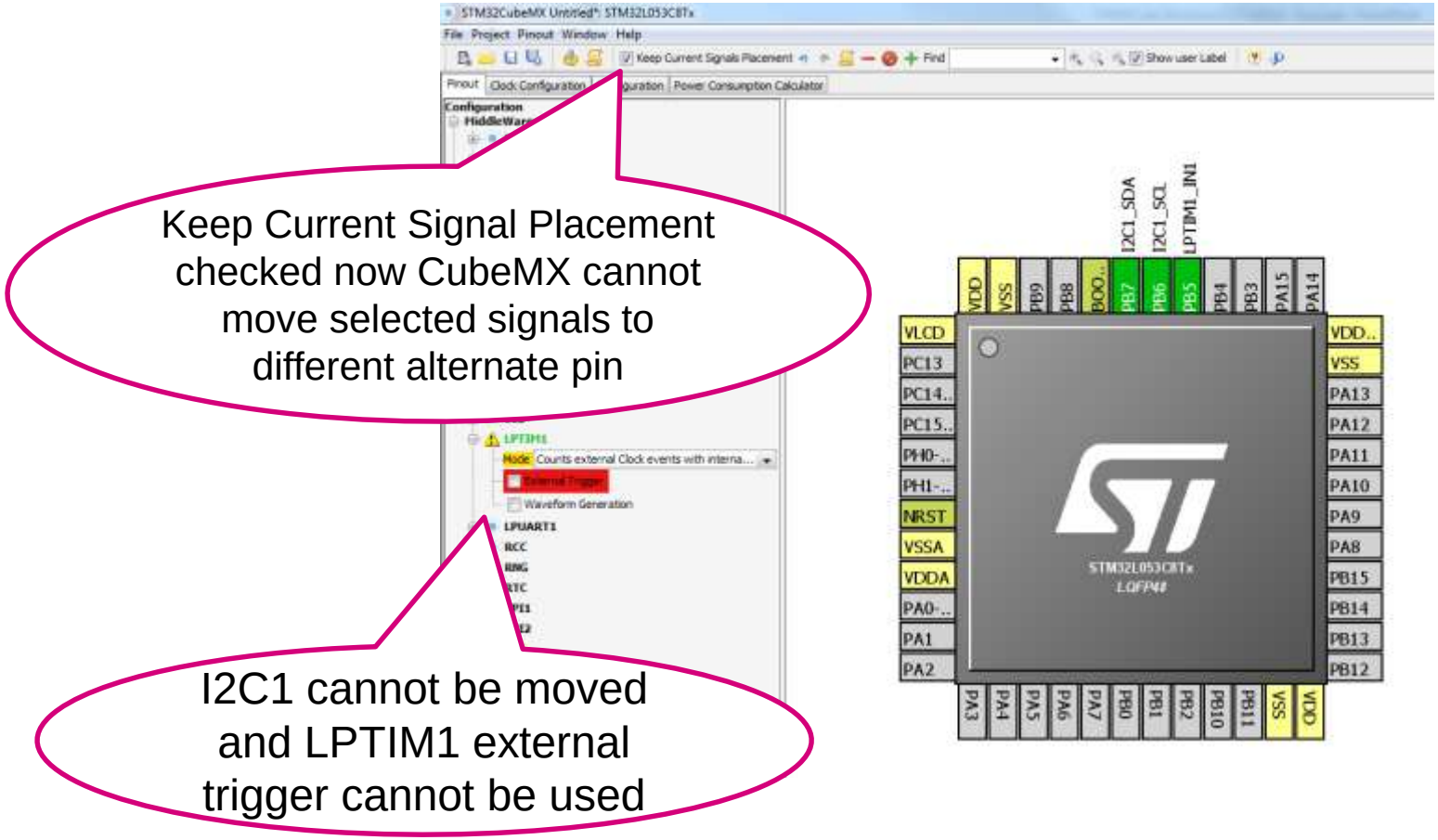
STM32CubeMX: Pinout configuration

- Ensure “Keep Current Signal Placement” is unchecked (by default)



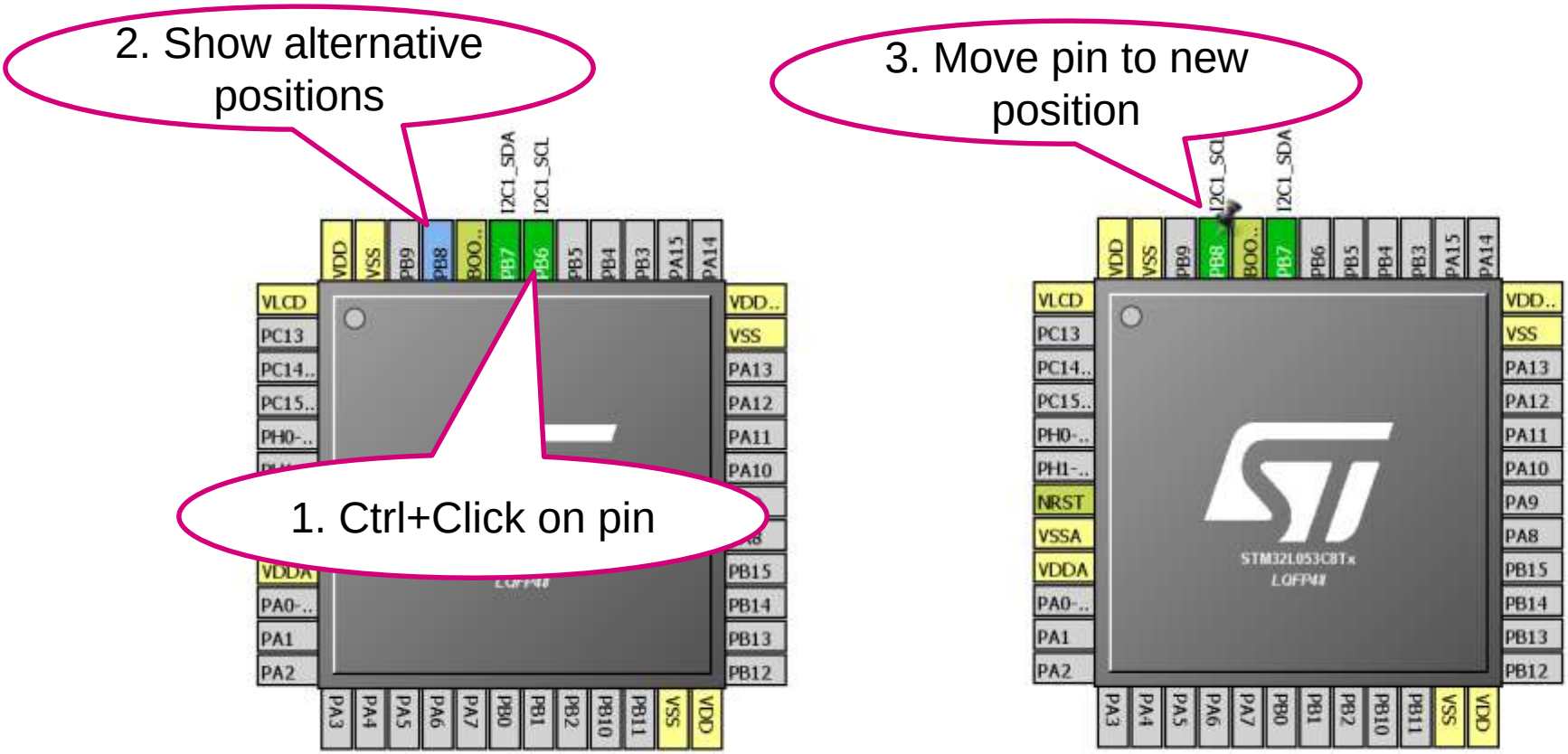
STM32CubeMX: Pinout configuration

- Keep User Placement renamed to Keep Current Signal Placement and is unchecked by default



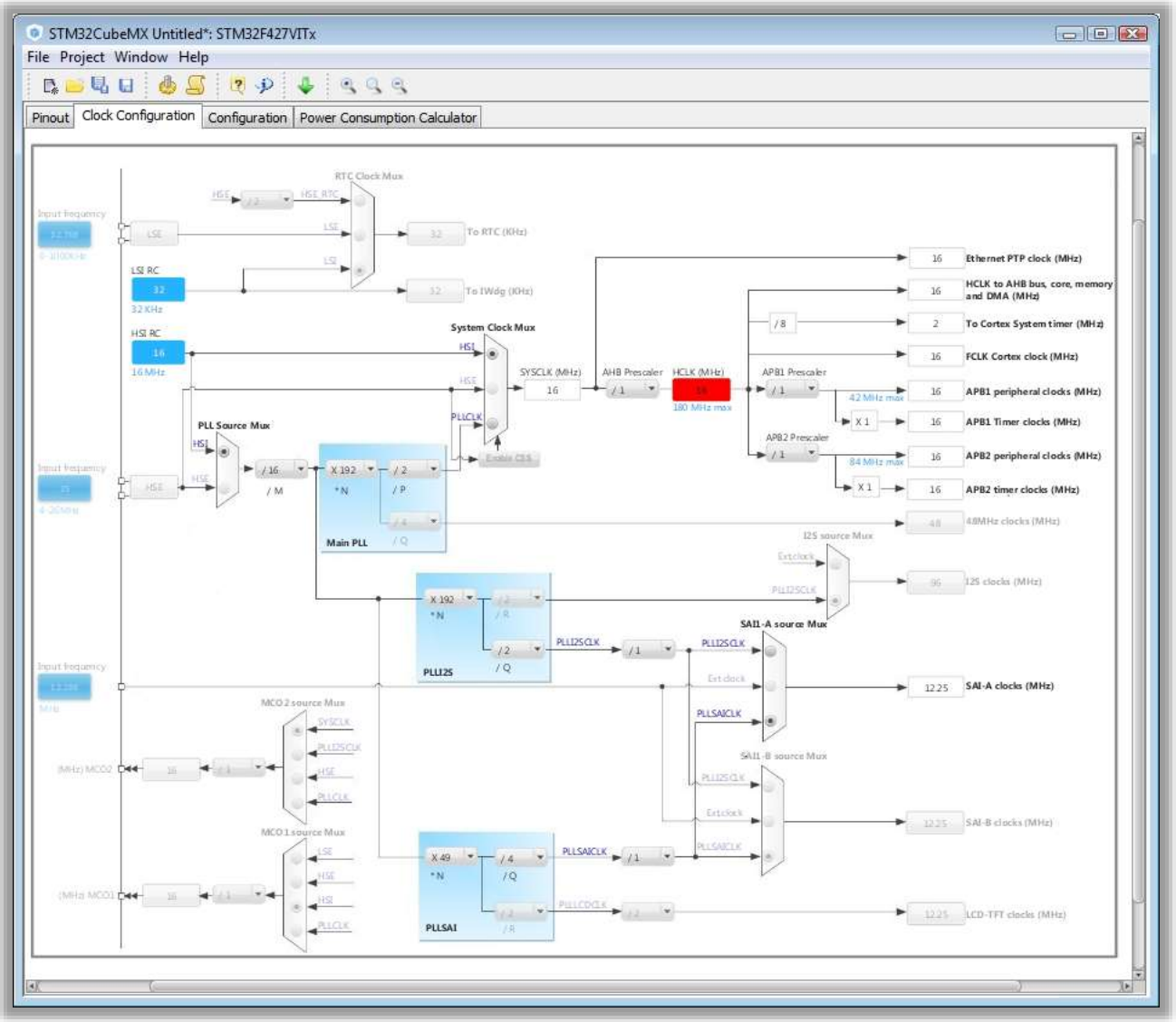
STM32CubeMX: Pinout configuration

- 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)



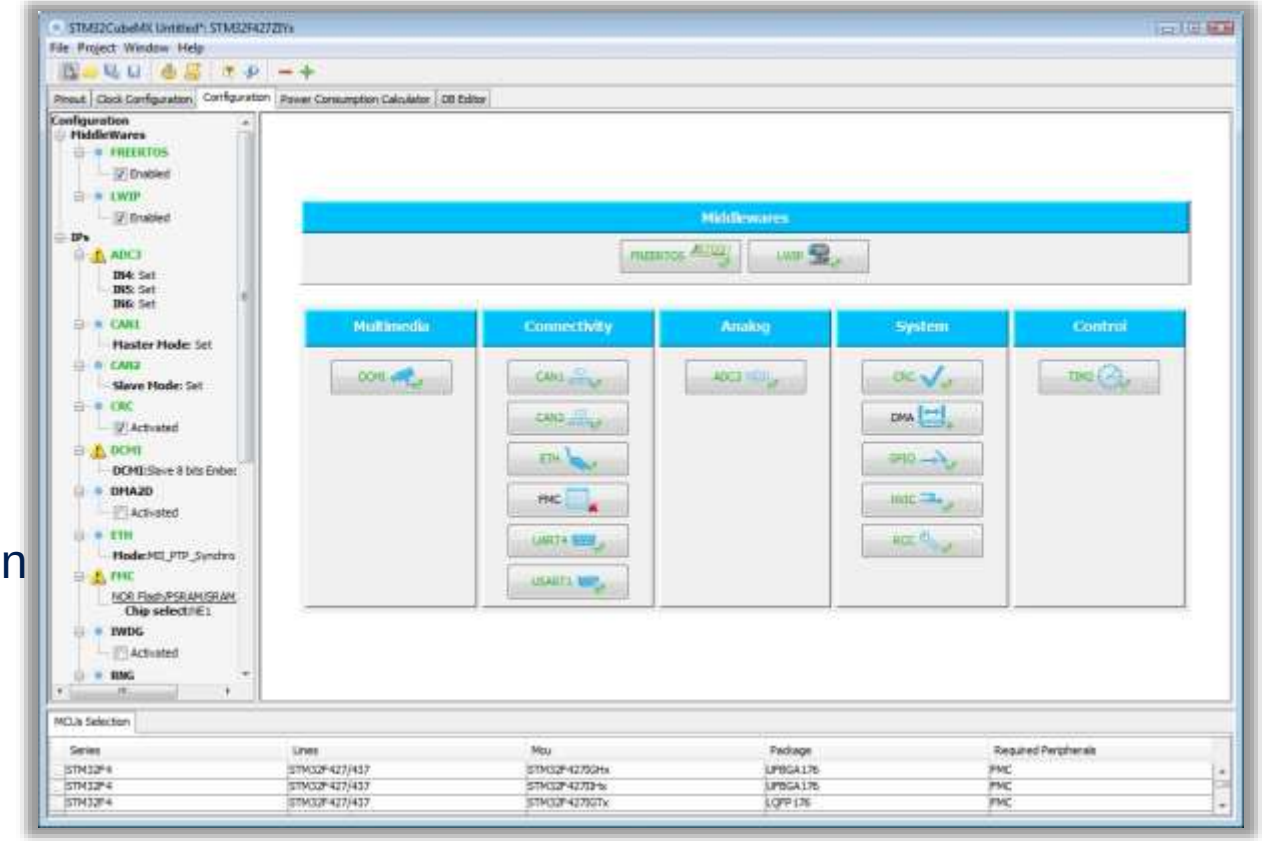
STM32CubeMX: Clock tree

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



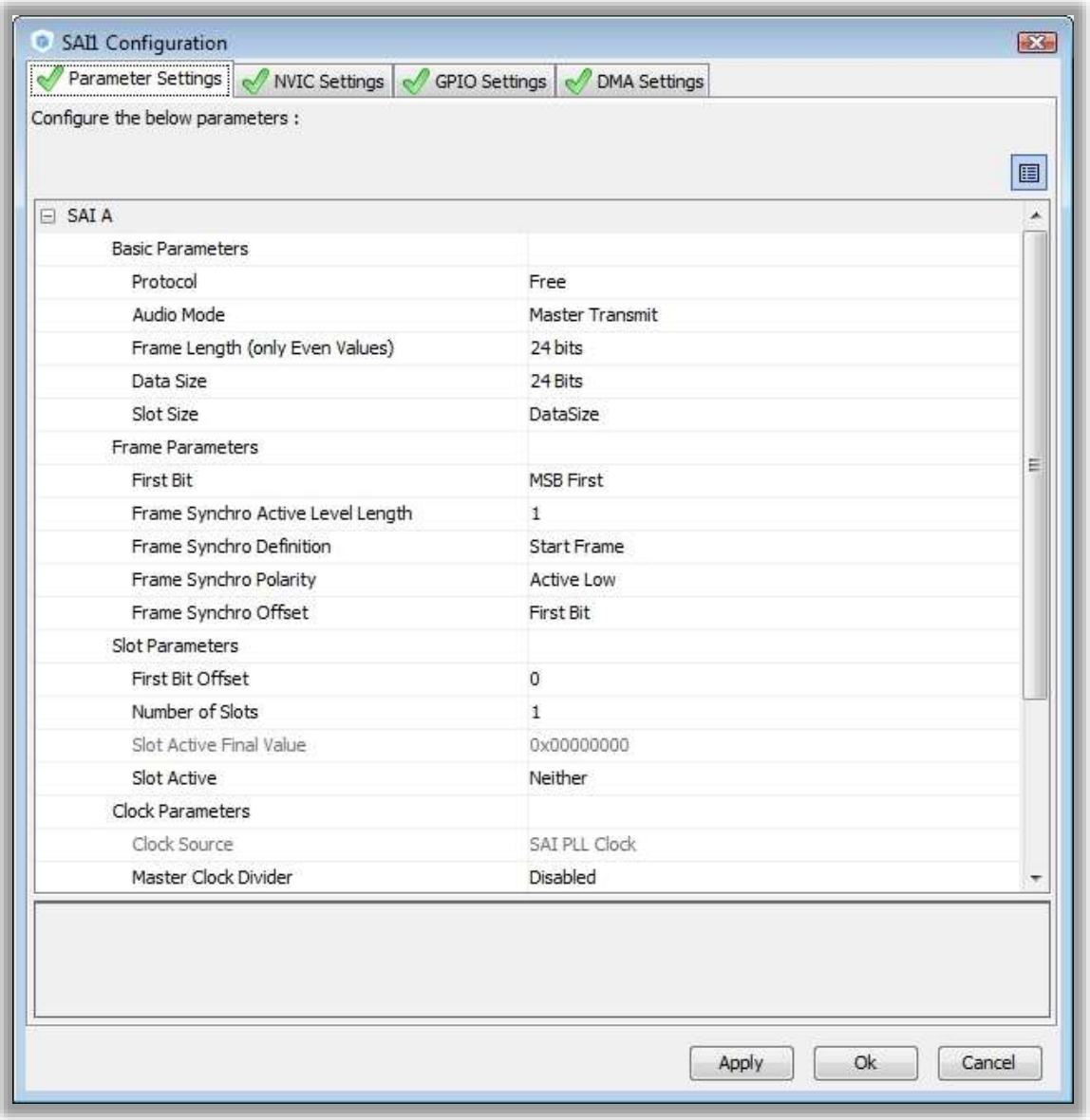
STM32CubeMX: Peripheral and middleware configuration

- 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



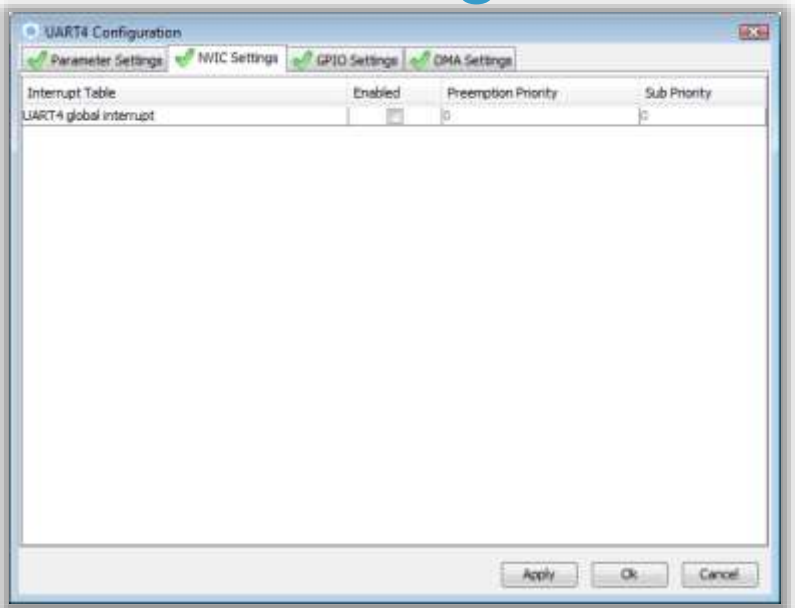
STM32CubeMX: Peripheral and middleware configuration

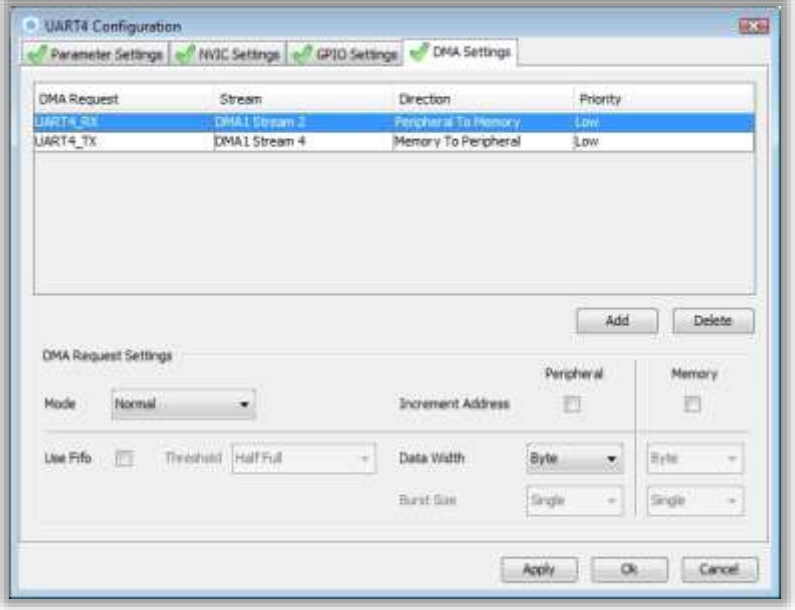
- Parameters with management of dependencies and constraints
- Interrupts
- GPIO
- DMA



STM32CubeMX: Peripheral and middleware configuration

- Manage Interruptions
  - priorities can only be set in the NVIC global view
- Manage GPIO parameters
- Manage DMA
  - Configure all the parameters of the DMA request
  - Runtime parameters (start address, …) are not managed

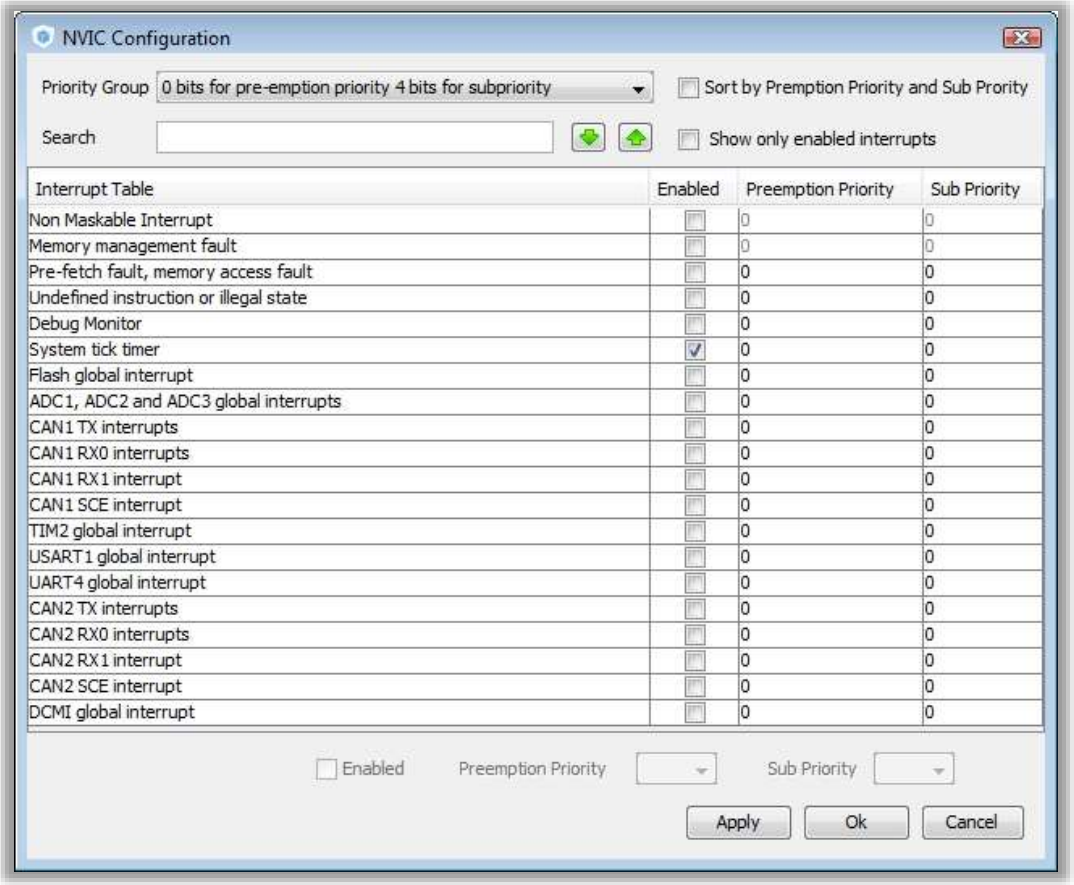




life.augmented

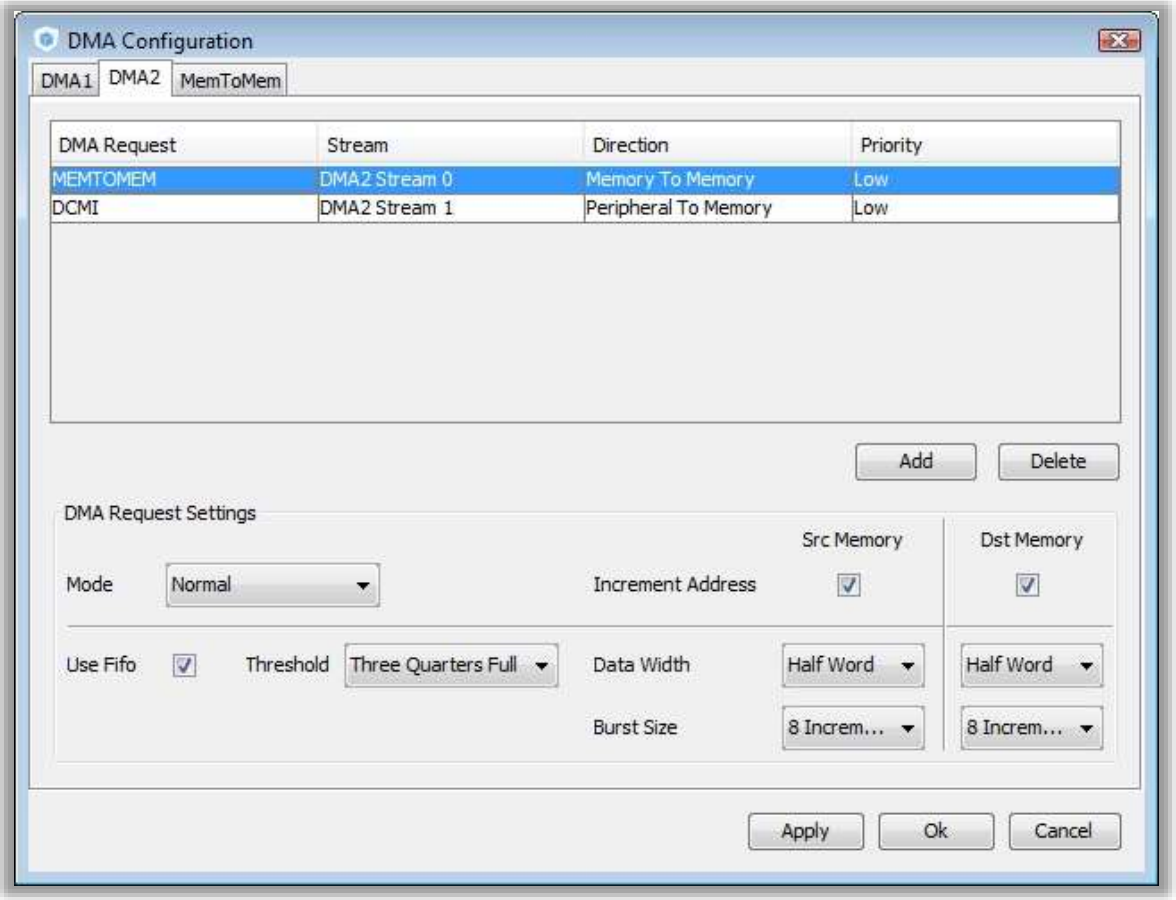
NVIC Panel

- Manage all interruptions
- Manage priorities and sort by priorities
- Search for a specific interrupt in the list



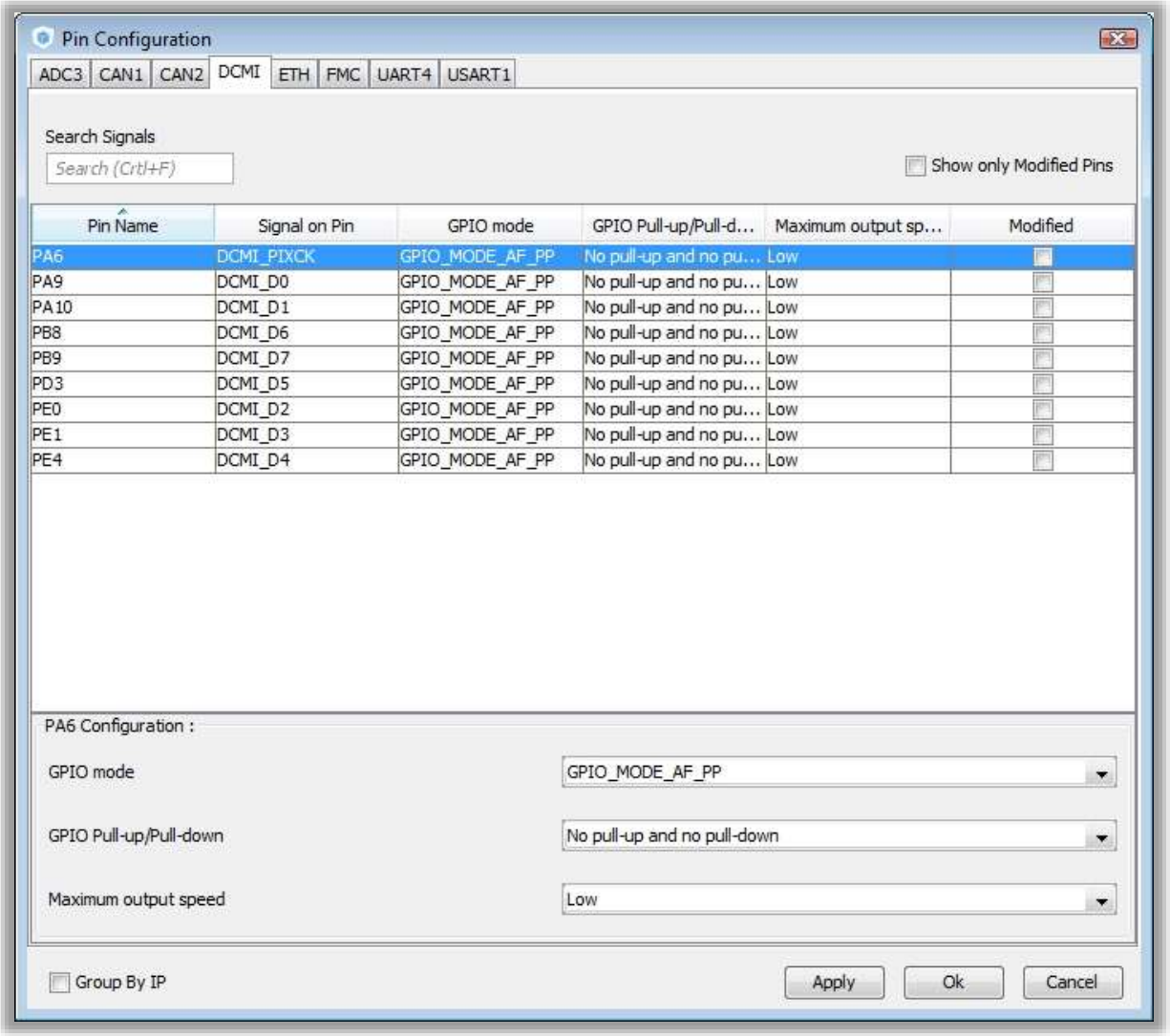
DMA Panel

- Manage All DMA requests including Memory to Memory
- Set Direction and priority
- Set specific parameters



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



STM32CubeMX: Code generation

- 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
- Required library code is automatically copied or referenced in the project (updater)

main.c

```
  ******************************************************************
  */
/* Includes ------------------------------------------------------
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
#include "lwip.h"
#include "usb_device.h"

/* Define structures */
ADC_HandleTypeDef hadc1;


/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
/* Private function prototypes -----------------------------------
static void SystemClock_Config(void);
static void StartThread(void const * argument);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_NVIC_Init(void);

int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
/* MCU Configuration----------------------------------------------
  /* Reset of all peripherals, Initializes the Flash interfa
  HAL_Init();
  /* Configure the system clock */
```

Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS

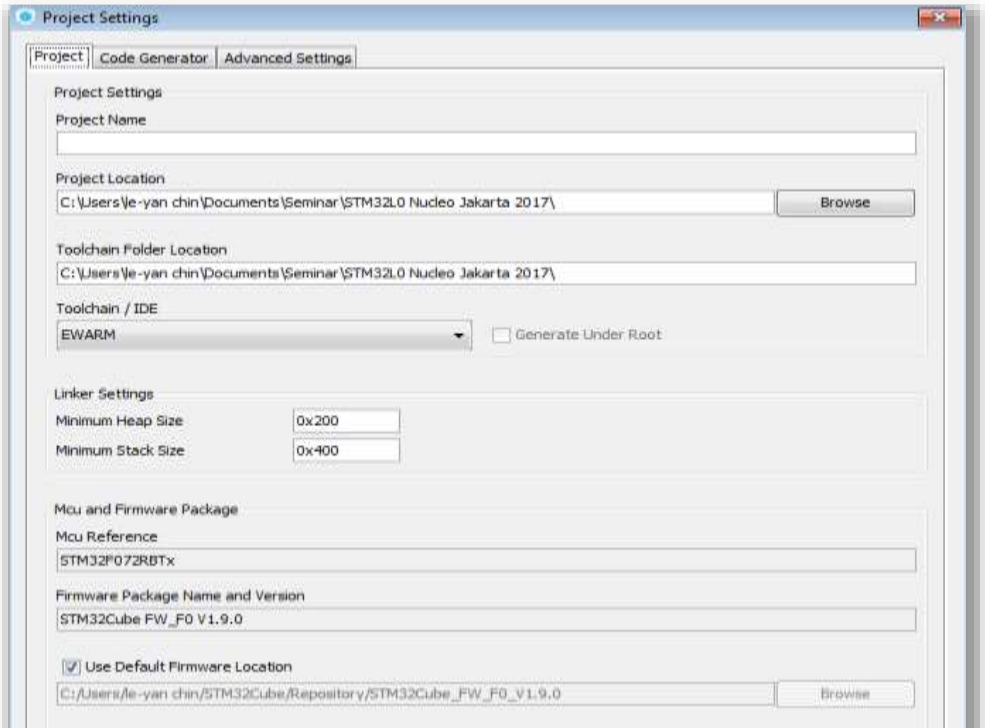
life.augmented

STM32CubeMX: Updater

- Help->Updater settings
  - Choose location of STM32CubeFx firmware libraries repository
  - Choose manual or automatic check
  - Configure connection parameters
    - Try to "Use System Proxy Parameters" first
    - If it doesn't work check with IT department
    - Alternatively, manually check and download from ST website
- Help->Install new libraries : Manage the content of the library repository
  - Click on the check button to see what is available
  - Select the library you want to install and click install now
    - The libraries will be automatically downloaded and unzipped

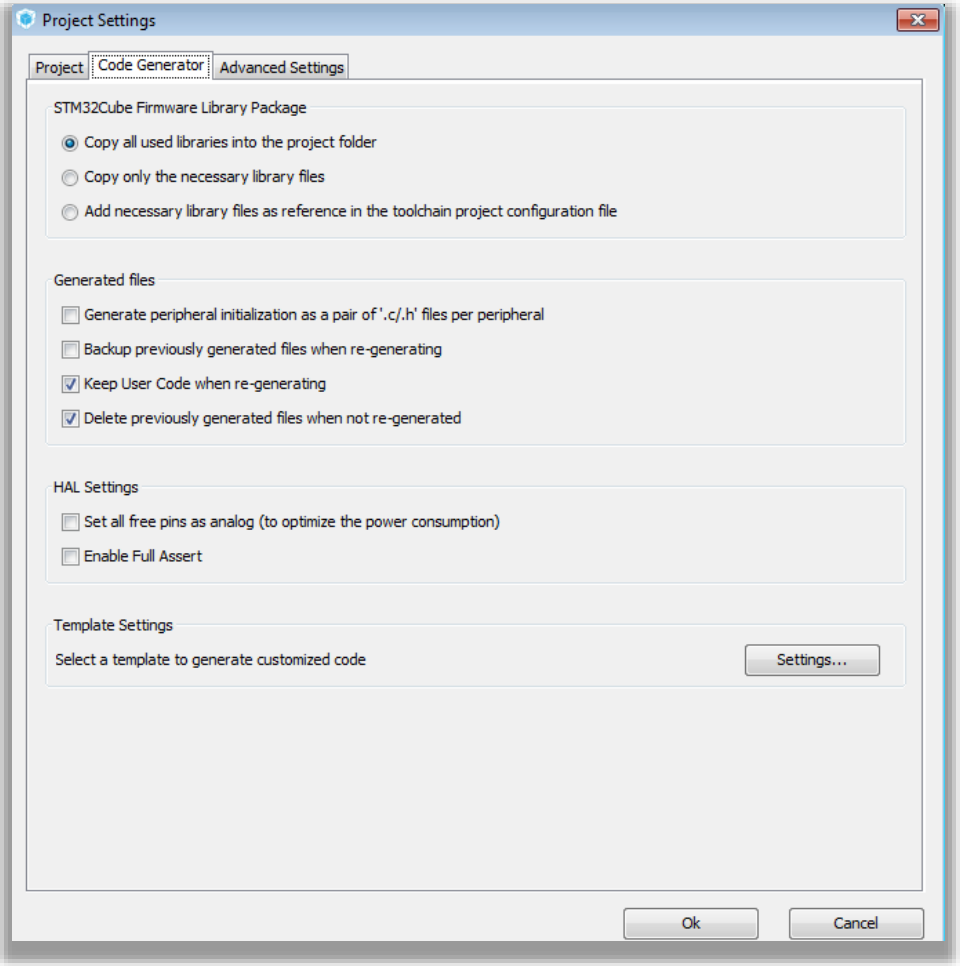
STM32CubeMX: Project settings

- Project -> Settings
  - Set project name and location
  - A full folder will be created named with the project name.
  - Inside this folder you'll find the saved configuration and all the generated code
  - Select toolchain (Keil, IAR, Atollic, SW4STM32)
  - You can choose to use the latest version of the firmware library or a specific one

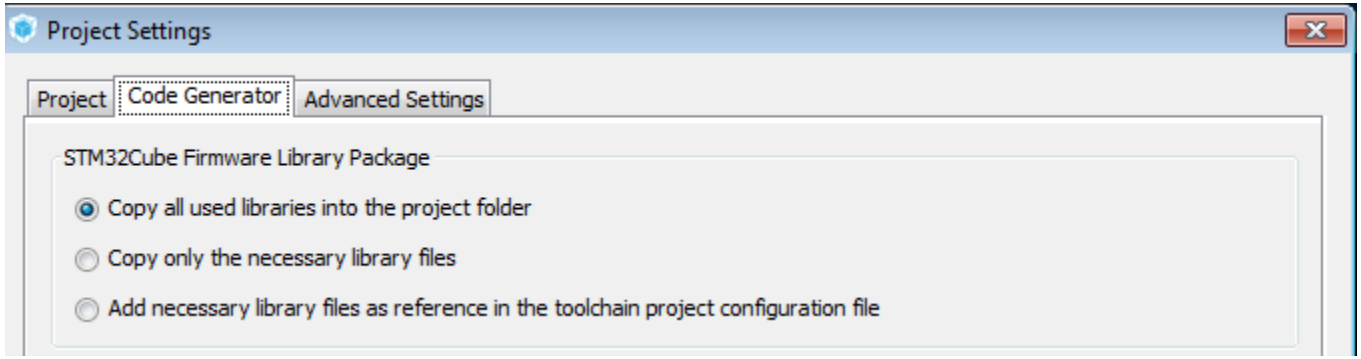


STM32CubeMX: Code Generator settings

- Code generator options
  - Either copy the full library or only the necessary files or just reference the files from the common repository
  - Generate all peripherals initialization in the stm32fYxx_hal_msp.c file or one file per peripheral
  - Keep user code or overwrite it (code between User code comment sections)
  - Delete or keep files that are not useful anymore
  - Set free pins as analog, this settings helps keep low consumption (**if SWD/JTAG is not selected in pinout, this option will disable it**)
  - Enable full assert in project, this help discover incorrect HAL function parameter used in user code



Code Generator options : STM32Cube Firmware Library package



- Copy all used libraries into the project folder

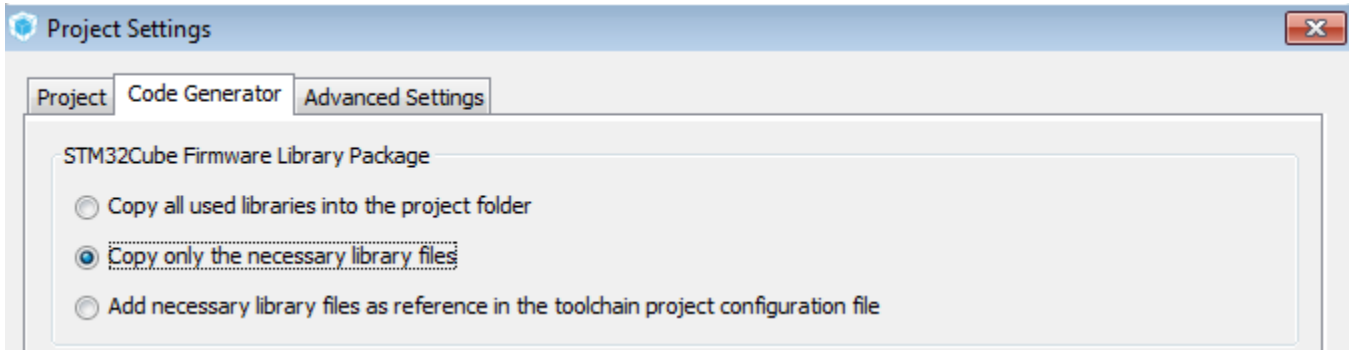
CubeMX repository → Project Driver Folder

Copy all driver files from CubeMX repository

In project are used only peripheral files selected in CubeMX

Stored in project_folder/Drivers/STM32XXXX_HAL_Driver folder

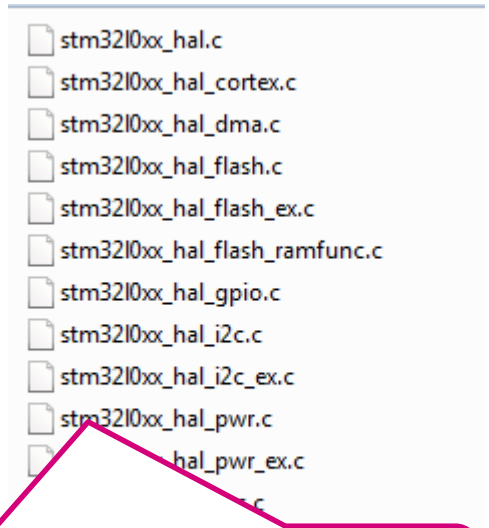
Code Generator options : STM32Cube Firmware Library package



- Copy only the necessary library files

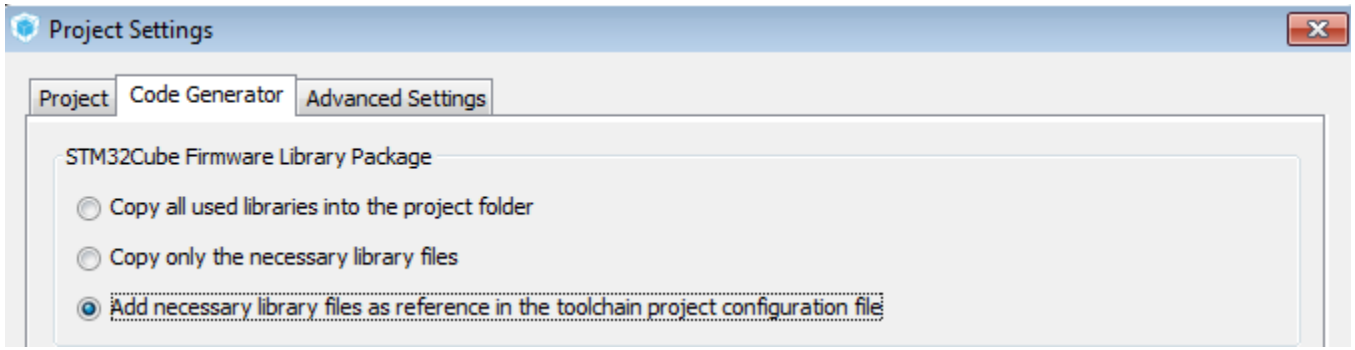
CubeMX repository → Project Driver Folder

Copy only files for peripherals selected in CubeMX



Stored in ***project_folder/Drivers/STM32XXXX_HAL_Driver*** folder

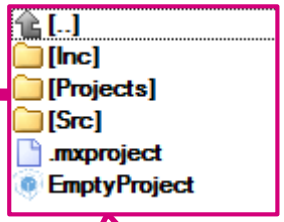
Code Generator options : STM32Cube Firmware Library package



- Add necessary library files as reference in the toolchain project configuration file

CubeMX repository

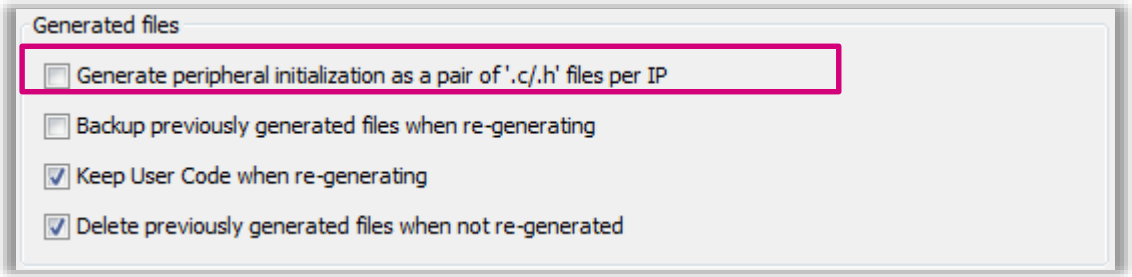
Project Driver Folder



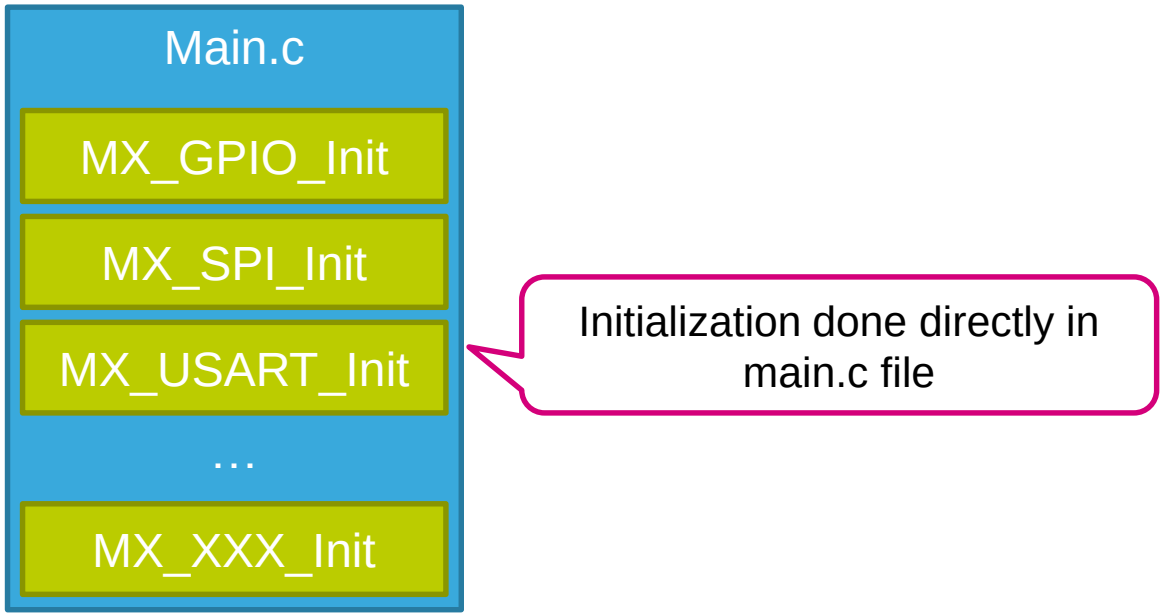
No ***Driver*** folder in project file

CubeMX not copy anything from repository.
Project use drivers from CubeMX repository

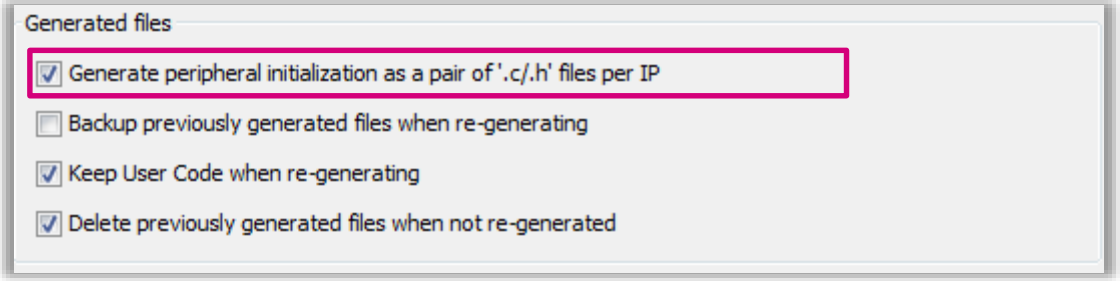
Code Generator options: Generate peripheral initialization as a pair of '.c/.h' files per IP



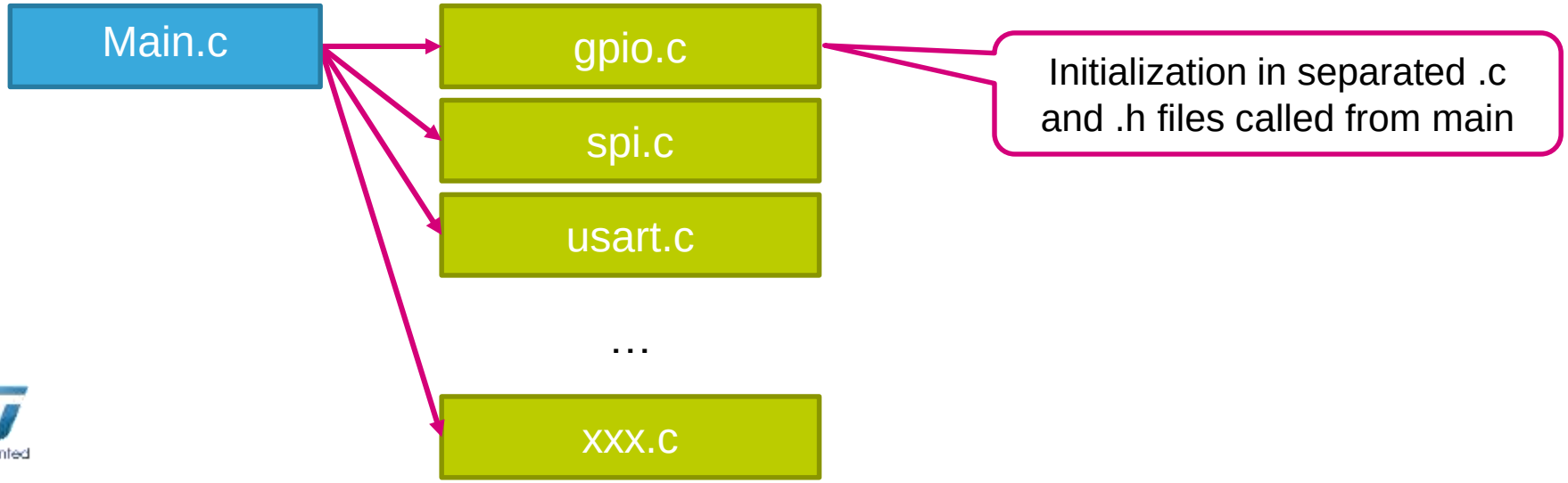
- By default this option is not used. All peripheral initialization code are generated in main.c



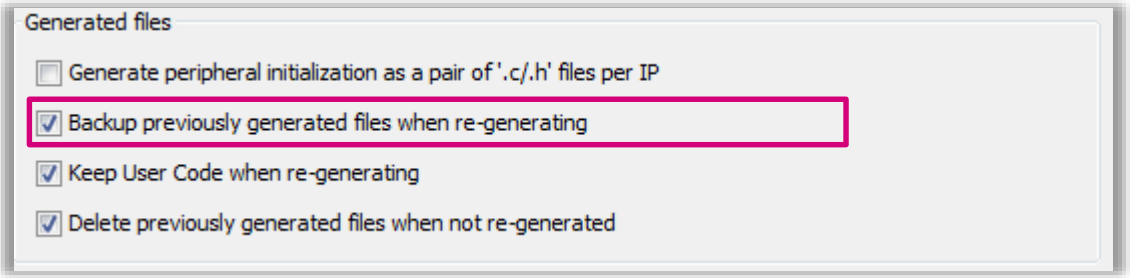
Code Generator optios: Generate peripheral initialization as a pair of '.c/.h' files per IP



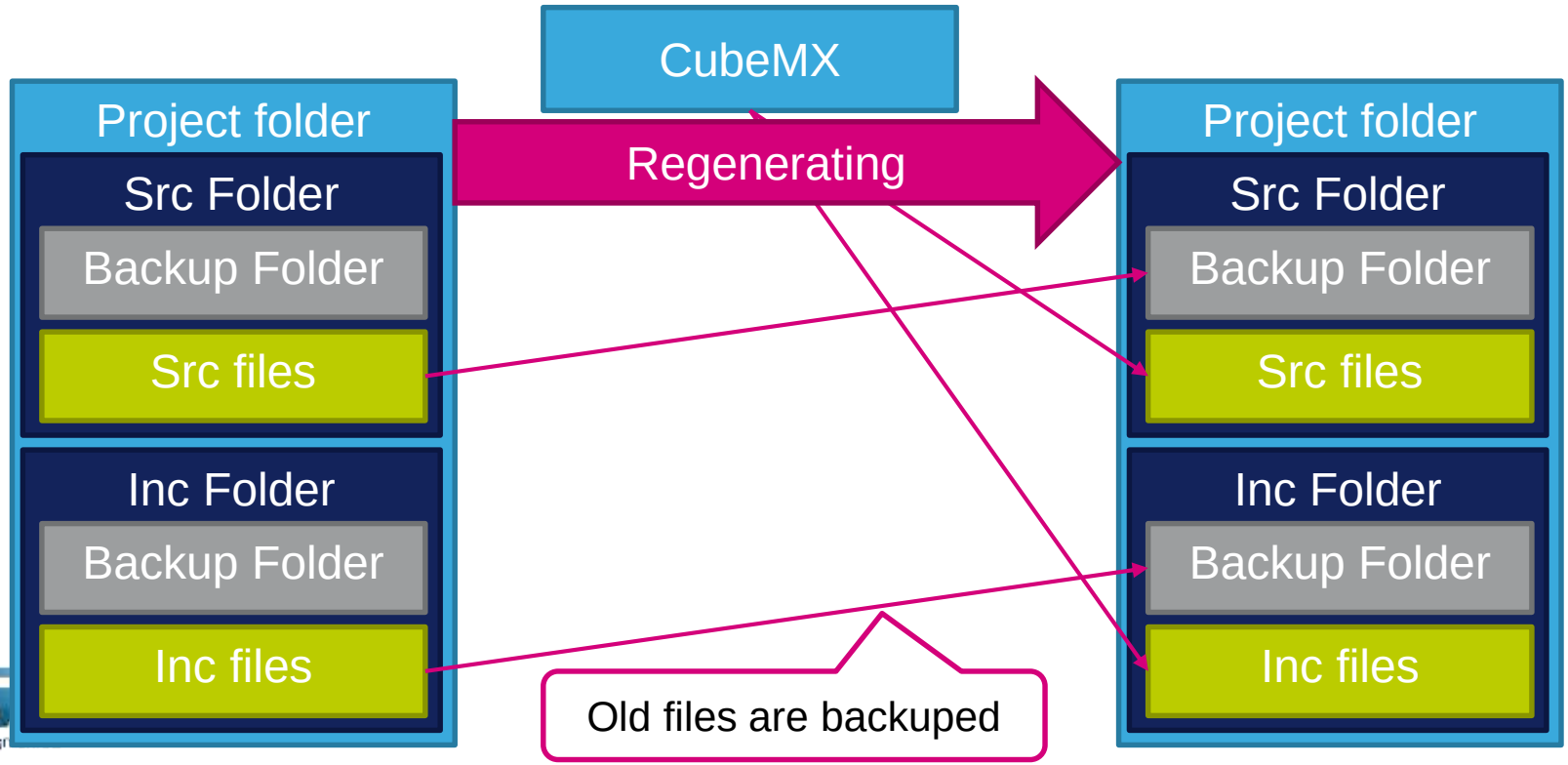
- Generate dedicated initialization .c and .h file for each periphery
- Advantage is that with .h file we can call MX_XXX init functions from every file in project not only from main.c



Backup previously generated files when re-generating



- Backup old files from **Src** and **Inc** folder into **Backup** folder

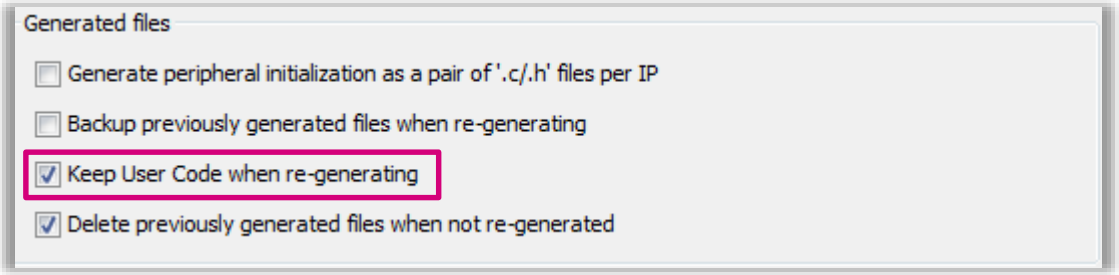


Keep User Code when re-generating

- Generated code contains USER CODE areas
- This areas are reserved in new code generation, if this option is selected

```
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
  /* USER CODE BEGIN 3 */
  /* Infinite loop */
  while (1)
  {

  }
  /* USER CODE END 3 */
}
```

Here can user put his code, code will be preserved during project generation

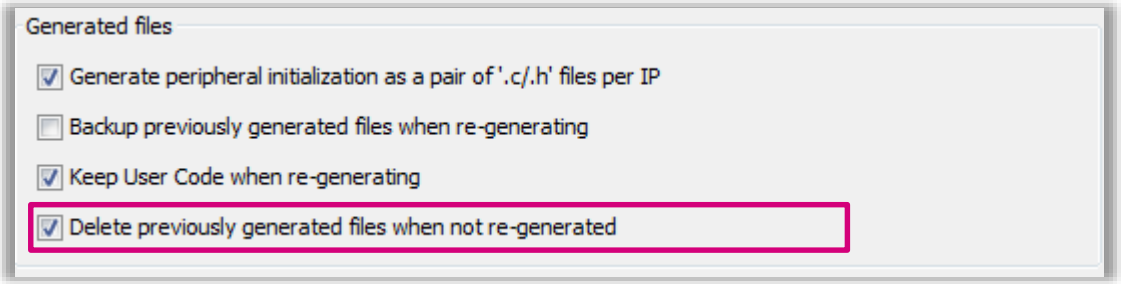


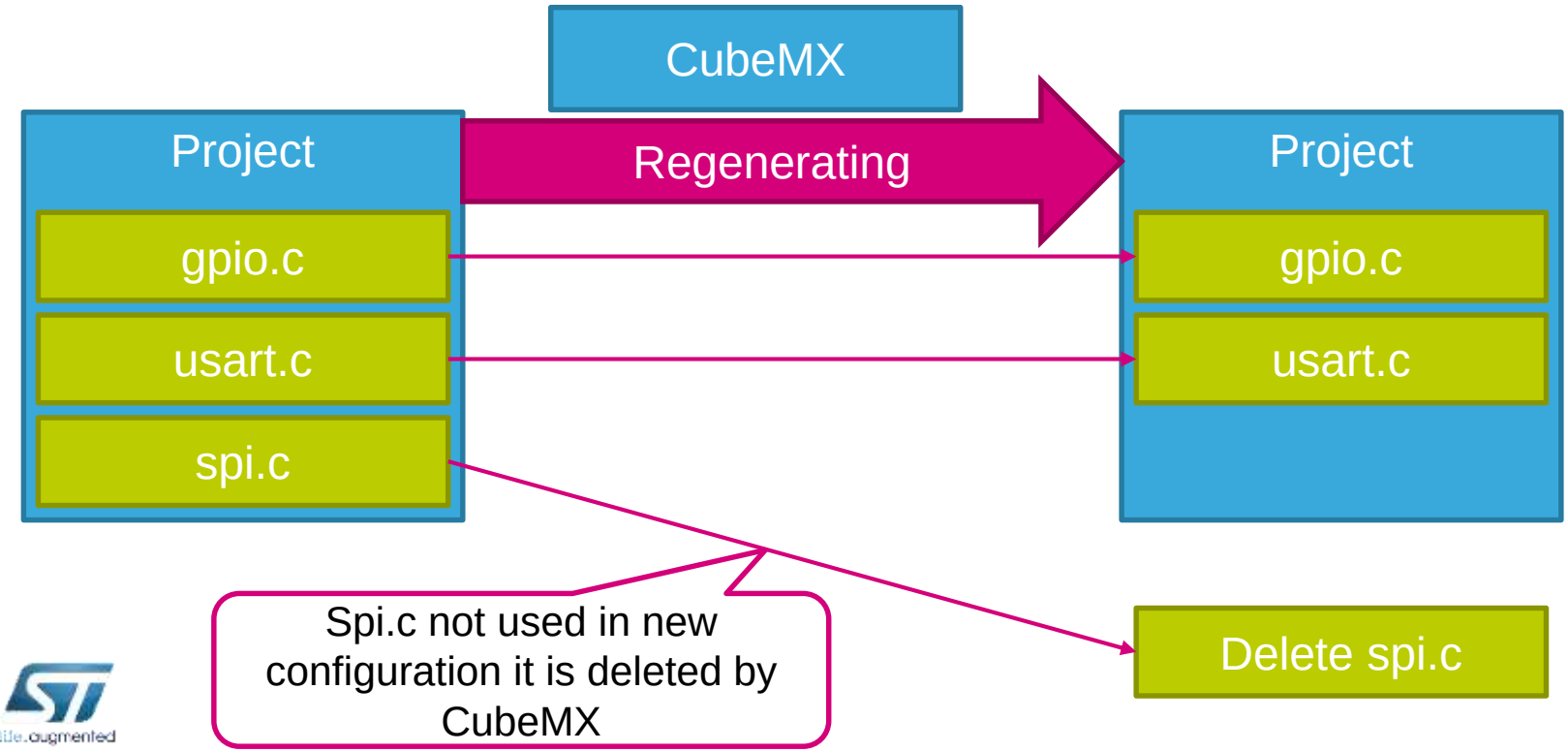
Keep User Code when re-generating

- Generated code contains USER CODE areas
- This areas are reserved in new code generation, if this option is selected
- Areas present in files generated by CubeMX
  - Main.c
  - Stm32l0xx_it.c
  - Stm32l0xx_hal_msp.c
- Areas cover important areas used for:
  - Includes
  - Variables
  - Function prototypes
  - Functions

```
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
```

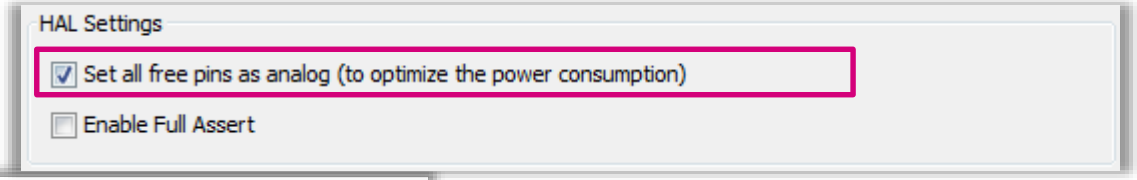
Code Generator options: Delete previously generated files when re-generating

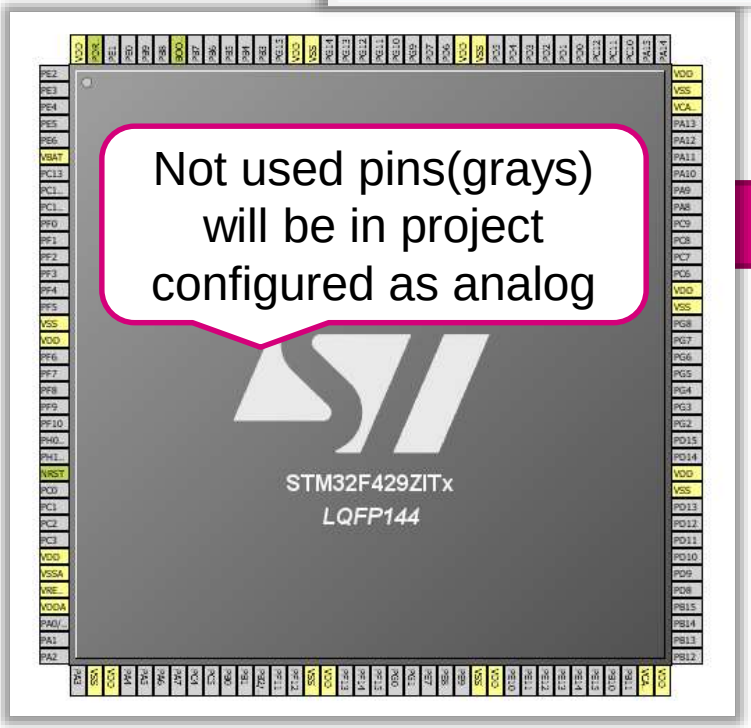




Set all free pins as analog

- This settings optimize power consumption of unused pins





CubeMX

Generating

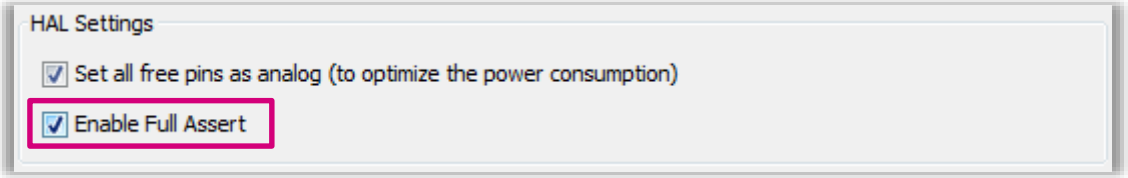
Project

MX_GPIO_Init

Initialization of unused pins is in MX_GPIO_Init

- If the **JTAG/SWD is not selected** in CubeMX, MX_GPIO_Init reconfigure JTAG/SWD pins to analog and this **disable debug possibilities**

Enable Full Assert



- Feature very useful during debugging
- Function input parameters are checked if they are in correct range, if not application jump into assert_failed function in main.c

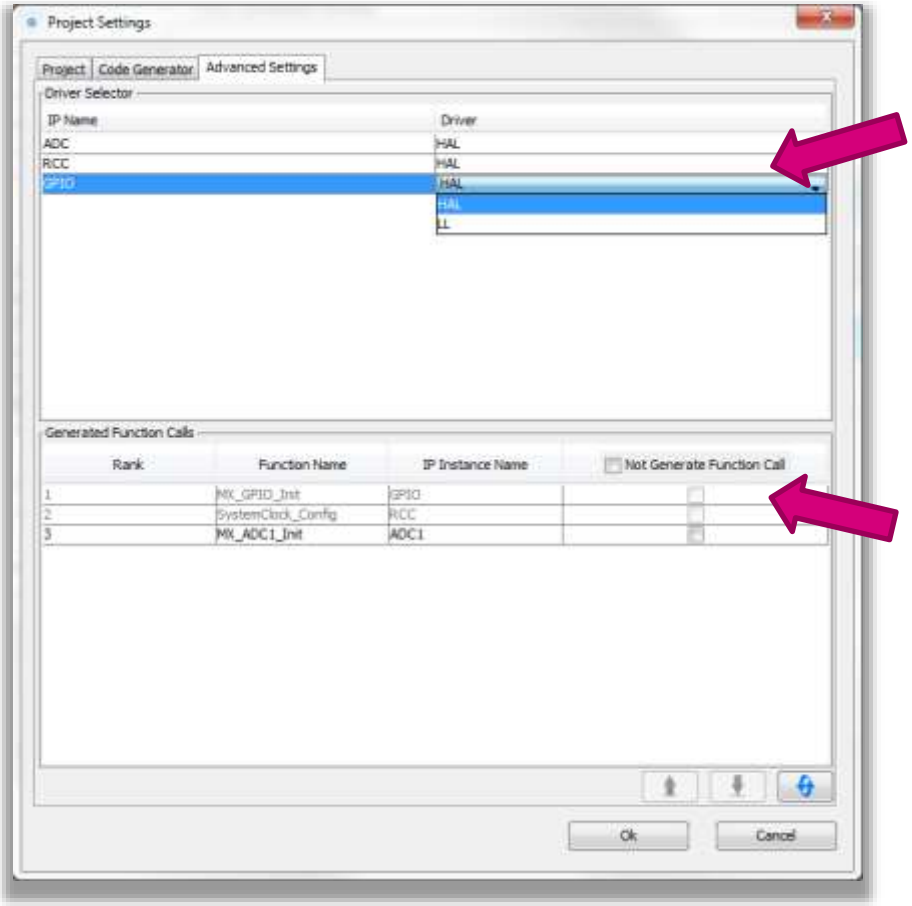
```
/* USER CODE BEGIN 2 */
 HAL_GPIO_TogglePin(GPIOA,(0x1<<17));
 /* USER CODE END 2 */
```

This function trying to configure not existing pin PA17

If parameter is not in valid range program jump into assert_failed function

```
void assert_failed(uint8_t* file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
```

Advanced options



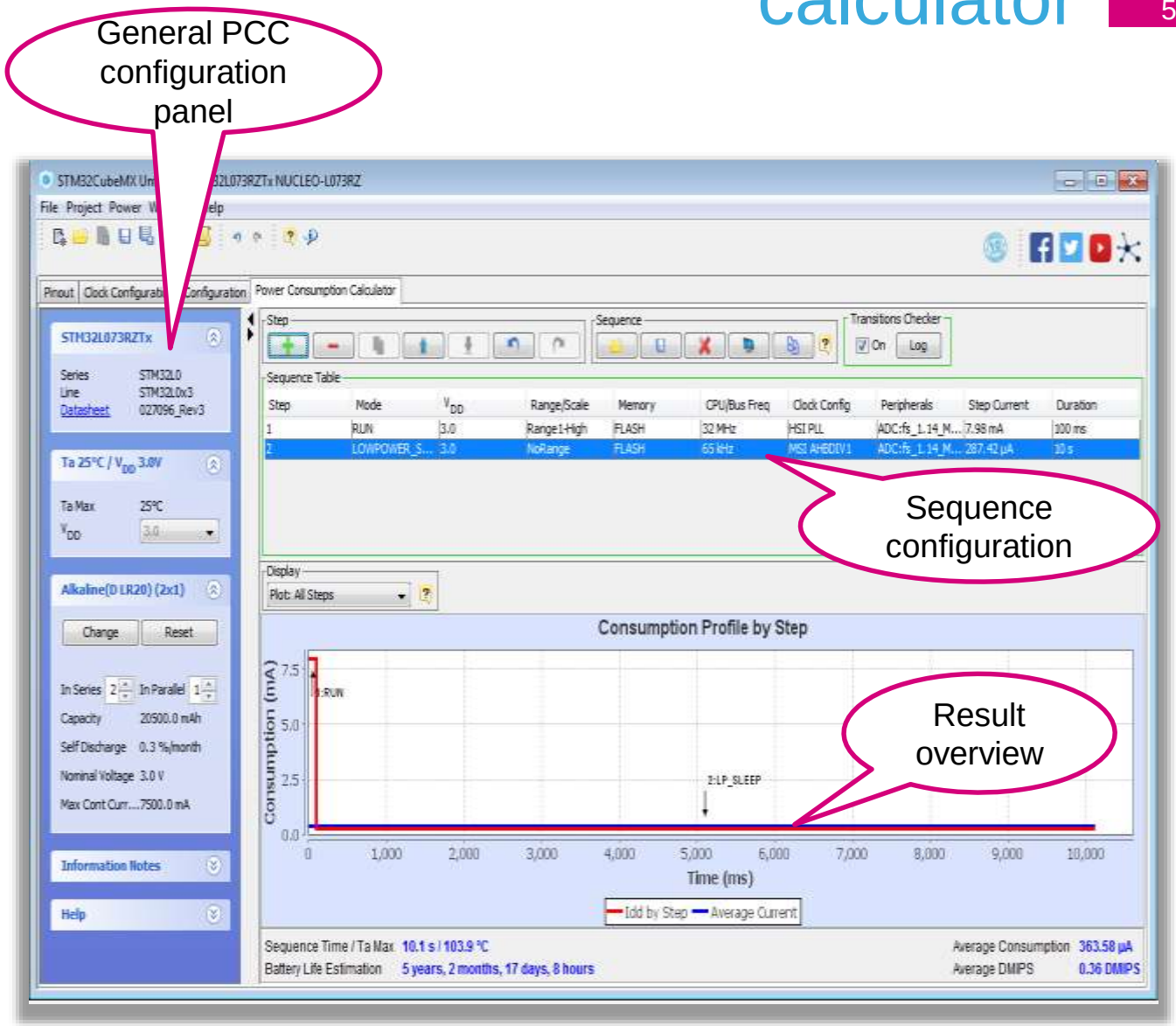
- API Driver Selector
  - Hardware Abstract Layer –use a high abstraction level based on standalone processes
  - Low Layer – deeply knowledge Hardware and process flow
- Generated function Call
  - The option to generate Initialization function Call of each periphery or not.

Warning and disclaimer

- Universal design for the entire STM32 family range at times prevents the tool from focusing on specific features of a particular product.
- **The STM32CubeMX GUI tool is not a replacement for the reference manual or datasheet**
  - Always refer to written documentation for further information!
  - Important features are often available on the product or in the HAL but not in the GUI.
- The GUI helps start a project and initialize a working starting configuration – but the configuration can be dynamically changed at runtime (i.e. GPIO, NVIC priority or clock settings).

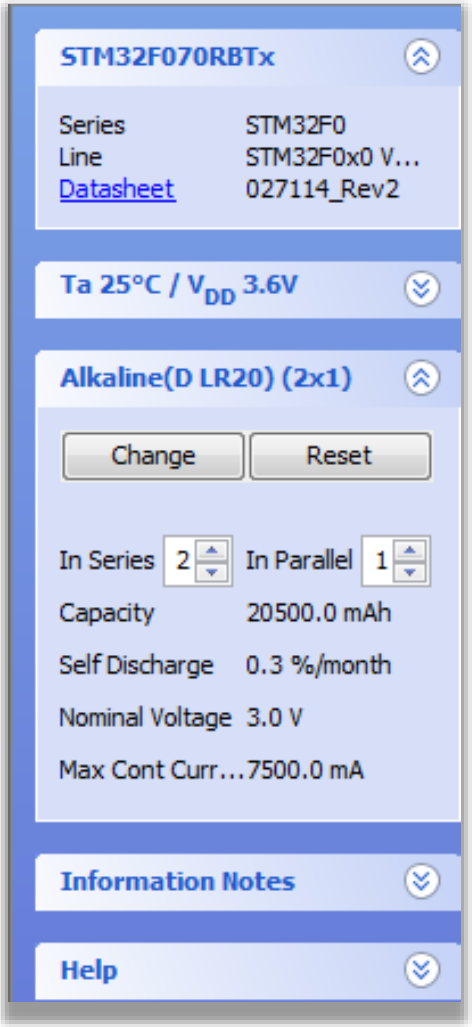
STM32CubeMX: Power consumption calculator

- Power step definitions
- Battery selection
- Creation of consumption graph
- Display of
  - Average consumption
  - Average DMIPS
  - Battery lifetime



General PCC parameters

- MCU selection inherited from STM32CubeMX
  - Use the direct link to the datasheet to get more detailed information.
- Parameter selection
  - Temperature and voltage choice may be limited, depending on the selected MCU.
- Battery selection – select typical or define your own
  - Battery is defined by capacity, voltage, self discharge and current limitations.
- Information notes
  - Purpose is to warn about estimation limitations.



Building a sequence

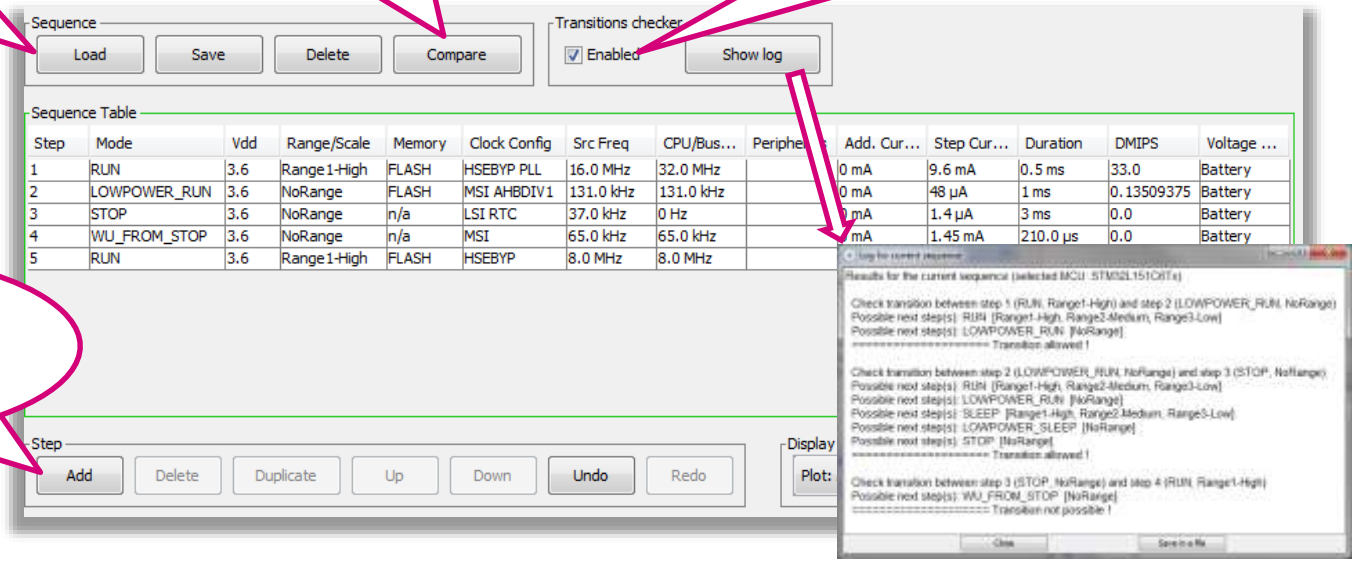
- A sequence is a set of ordered steps.

Load existing sequences and adapt them.

Compare sequences, even with different MCUs.

Check automatically if proposed power step transitions are valid.

Create new steps by adding or duplicating existingones.

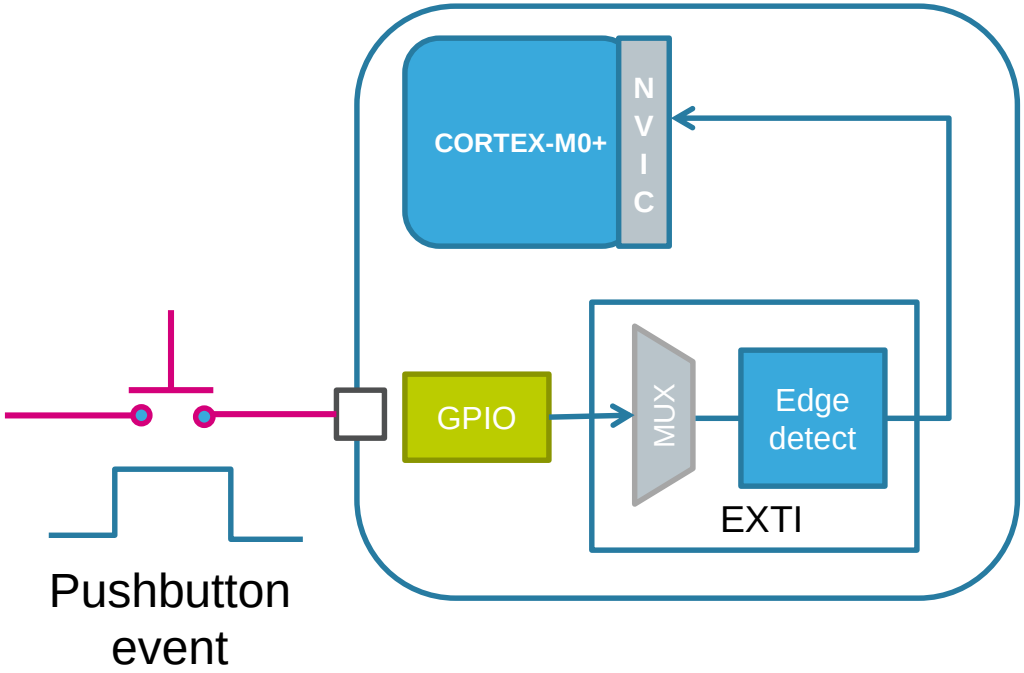


Hands-on Demo : GPIO and EXTI with STM32CubeMX

life.augmented

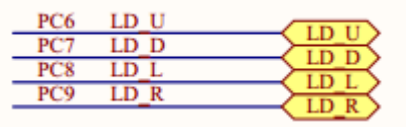
GPIO and EXTI Hands-on Demo

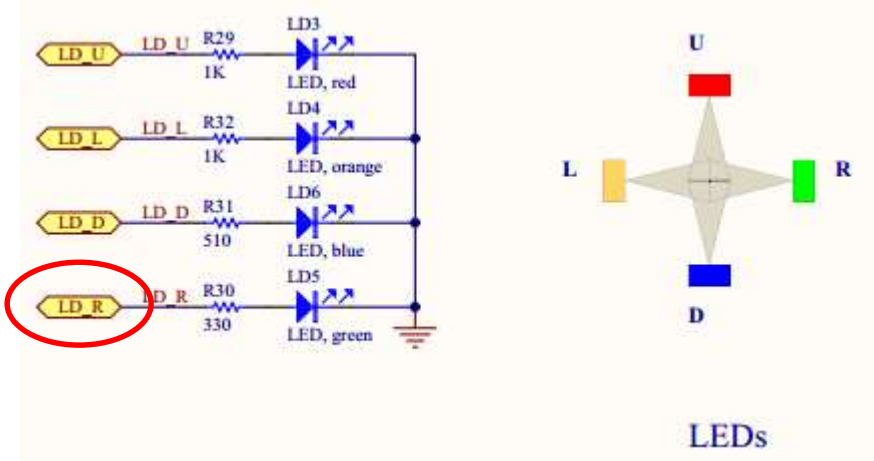
- Part 1: Configure a GPIO in External Interrupt mode.



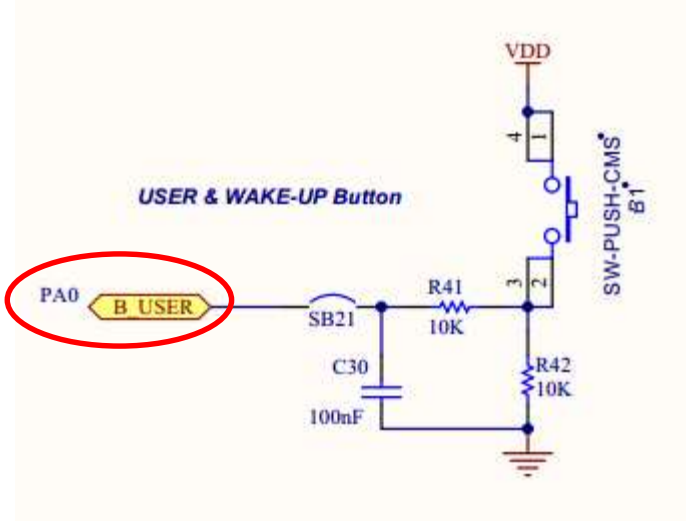
STM32 Nucleo User Pushbutton and LEDs

Active High





Active Low



GPIO_EXTI

- This hands-on describes how to use the GPIO HAL APIs.
  The User pushbutton, configured as input with interrupt, will be used to change the states of the LEDs.
- For this hands-on, the STM32CubeMX will be used to generate the initialization codes for the GPIOs and System clock. This process will speed up the development as the initialization codes are generated by the STM32CubeMX tool. The user then will only need to add the user codes as per application. Recommended especially for first time users of the STM32.
  - Create an new project
    - Target MCU: STM32F072RBT6
  - Save this project in folder below with project name: Lab-GPIO_EXTI
    - C:\..\STM32F0 Discovery Exercises***Lab-GPIO_EXTI***
  - In the Project->Settings, same as before, make sure that the following are set:
    - Project
      - Toolchain/IDE: MDK-ARM V5
      - Firmware Package Name and Version: STM32Cube FW_F0 V1.9.0 (or download the STM32CubeF0 latest version)
    - Code Generator
      - Copy only the necessary library files (to reduce the size of the project folder)
      - Keep User Code when re-generating
      - Delete previously generated files when not re-generated

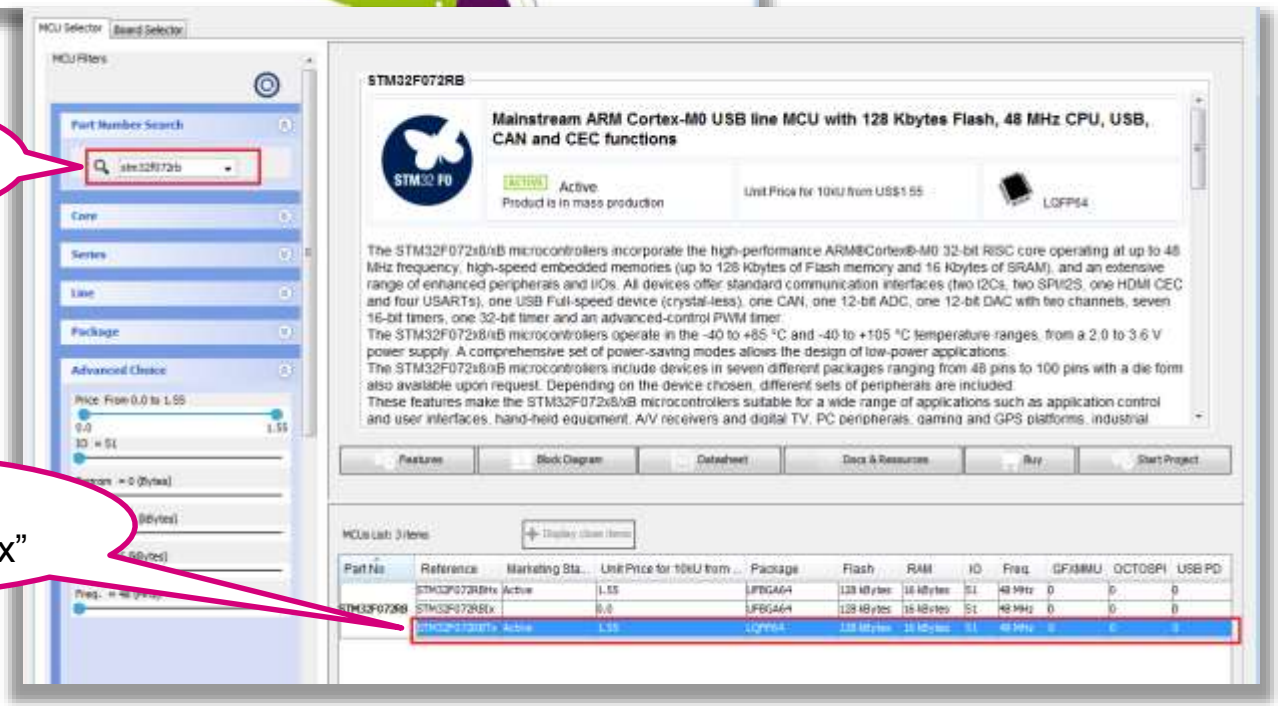
life.augmented

Screenshots

Step 1. Select New Project

Step 2. Type "STM32F072RB"

Select "STM32F072RBTx"



life.augmented

Lab-GPIO_EXTI: STM32CubeMX Config

- Objective:
  - Configure the GPIOS for LEDs
  - Configure the GPIO for the User pushbutton as input with interrupt(EXTI).
- Use the STM32CubeMX to configure the GPIOs accordingly:
  - Pinout Tab
    - GPIOs
      - PA0 (User pushbutton) – GPIO EXTI0
      - PC9 (LD_R - Green) – GPIO output
    - Sys (System)
      - Enable Serial Wire Debug (SWD)

      Although the SWD debug pins are active after reset, it is a good practice to make sure the debug pins are reserved for debug purposes while assigning pins for your application. This avoids assigning it for other alternate function by mistake while still in firmware development stage

Screenshots

Step 3. Type "pa0" in Find column

STM32CubeMX Untitled: STM32F072RBTx

File Project Pinout Window Help

Keep Current Signals Placement | Find pa0 | Show User Label

Pinout | Clock Configuration | Configuration | Power Consumption Calcu

PA0

Configuration
- MiddleWares
  - FATFS
  - FREERTOS
  - USB_DEVICE
- Peripherals
  - ADC
  - CAN
  - COMP1
  - COMP2
  - CRC
  - DAC
  - HDMI_CEC
  - I2C1
  - I2C2
  - I2S1
  - I2S2
  - IRTIM
  - IWDG
  - RCC
  - RTC
  - SPI1
  - SPI2

VDD VSS PB9 PB8 BOO.. PB7 PB6 PB5 PB4 PB3 PD2 PC12 PC11 PC10 PA15 PA14

VBAT PC13 PC14.. PC15.. PF0-.. PF1-.. NRST PC0 PC1 PC2 PC3 VSSA VDDA PA0 PA1 PA2

VDD.. VSS PA13

PA3 VSS VDD PA4 PA5 PA6 PA7

PA0
- Reset_State
- ADC_IN0
- COMP1_INM
- COMP1_OUT
- RTC_TAMP2
- SYS_WKUP1
- TIM2_CH1
- TIM2_ETR
- TSC_G1_IO1
- USART2_CTS
- USART4_TX
- GPIO_Input
- GPIO_Output
- GPIO_Analog
- GPIO_EXTI0

GPIO_EXTI0

STM32F072RBTx
LQFP64

VDD VSS PB9 PB8 BOO.. PB7 PB6 PB5 PB4 PB3 PD2 PC12 PC11 PC10 PA15 PA14

VDD.. VSS PA13 PA12 PA11 PA10 PA9 PA8 PC9 PC8 PC7 PC6 PB15 PB14 PB13 PB12

PA1 PA2

PA3 VSS VDD PA4 PA5 PA6 PA7 PC4 PC5 PB0 PB1 PB2 PB10 PB11 VSS VDD

Step 4. click on PA0 & select "GPIO_EXT0"

life.augmented

Screenshots

Step 5. Type "pc9" in Find column

STM32CubeMX Untitled*: STM32F072RBTx

File Project Pinout Window Help

Keep Current Signals Placement | Find pc9 | Show User Label

Pinout | Clock Configuration | Configuration | Power Consumption Calcu

Configuration
- MiddleWares
  - FATFS
  - FREERTOS
  - USB_DEVICE
- Peripherals
  - ADC
  - CAN
  - COMP1
  - COMP2
  - CRC
  - DAC
  - HDMI_C
  - I2C1
  - I2C2
  - I2S1
  - I2S2
  - IRTIM
  - IWDG
  - RCC
  - RTC
  - SPI1
  - SPI2

STM32F072RBTx
LQFP64

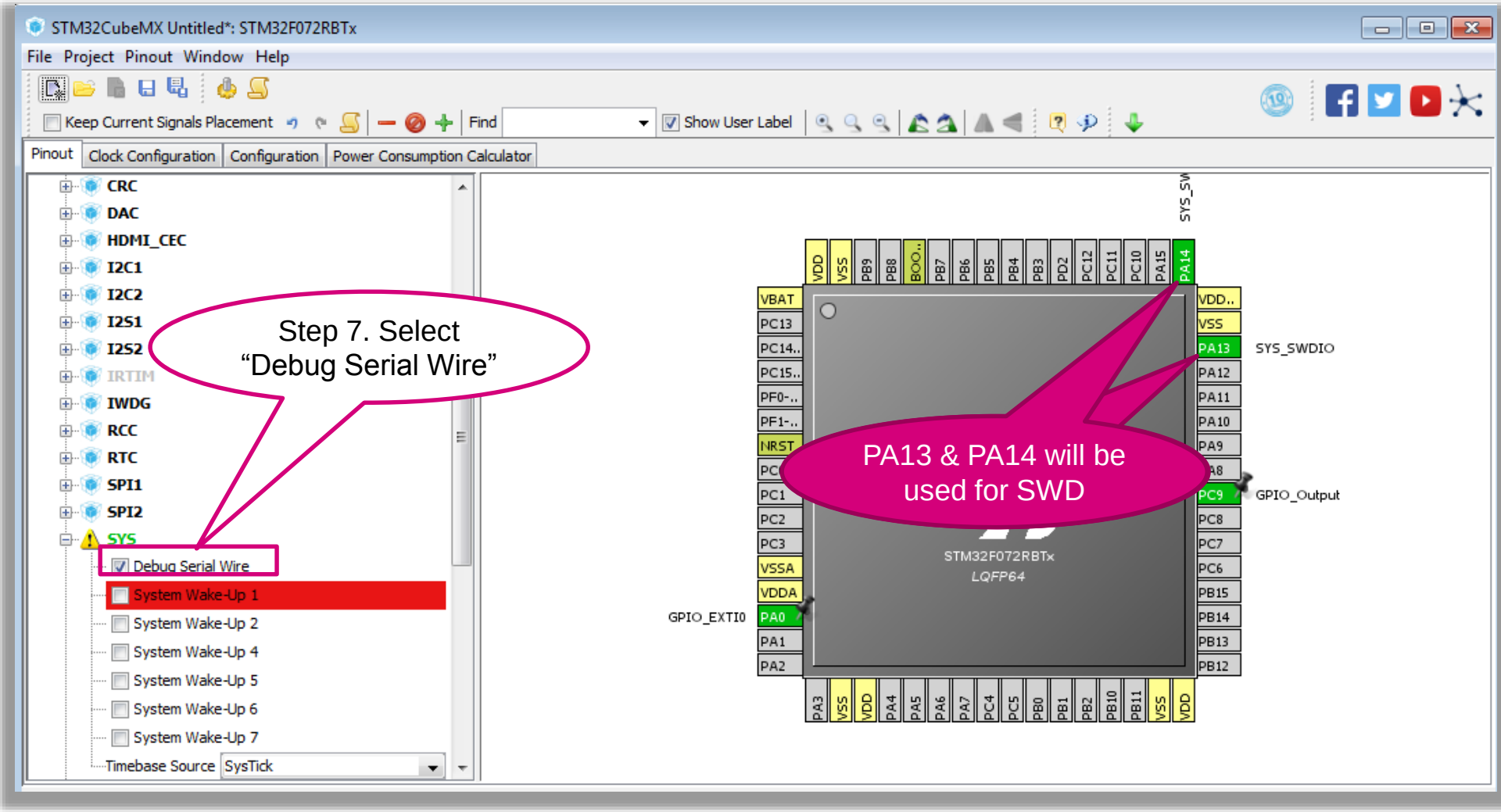
GPIO_EXTI0

PC9
- Reset_State
- DAC_EXTI9
- TIM3_CH4
- GPIO_Input
- GPIO_Output
- GPIO_Analog
- GPIO_EXTI9

Step 6. click on PC9 & select "GPIO_Output"

life.augmented

Screenshots



Lab-GPIO_EXTI: STM32CubeMX Config

- Clock Configuration Tab
  - System Clock using HSI (8MHz) as clock source:
    - HSI as SYSCLK clock source
    - SYSCLK = HCLK(AHB) = 8MHz
    - APB1(PCLK1) = 8MHz
    - APB2(PCLK2) = 8MHz

Screenshots

Step 8. Click on "Clock Configuration" tab

Verify the System Clock selection is "HSI" & 8MHz

STM32CubeMX Untitled* 072RBTx

File Project Clock Configuration Window Help

Pinout | Clock Configuration | Configuration | Power Consumption Calculator

RTC Clock Mux

I2C1 Clock Mux

Input frequency 32.768 0-1000 KHz

LSE

LSI RC 40 40 KHz

HSI RC 8 8 MHz

HSI48 RC 48 48 MHz

Input frequency 8 4-32 MHz

HSE

HSI14 RC 14

HSE / 32 HSE_RTC

LSE

LSI

40 To RTC

40 To IWDG (KHz)

8 To FLITF (MHz)

14 To ADC (MHz)

PLL Source Mux

HSI

HSI48

HSE

PREDiv / 1

VCOInput 8

*PLLMul X 2

PLL

System Clock Mux

HSI

HSI48

HSE

PLLCLK

Enable CSS

SYSCLK (MHz) 8

AHB Prescaler / 1

HCLK (MHz) 8 48 MHz max

APB1 Prescaler / 1

PCLK1 48 MHz max

X 1

USB Clock Mux

PLLCLK

HSI48

48 To USB (MHz)

CEC Clock Mux

USART1 Clock Mux

SYSCLK

HSI

LSE

PCLK1

8 To USART1 (MHz)

8 To I2C1 (MHz)

8 To I2S1/I2S2 (MHz)

8 HCLK to AHB bus, core, memory and DMA (MHz)

8 To Cortex System timer (MHz)

8 FCLK (MHz)

8 APB1 peripheral clocks (MHz)

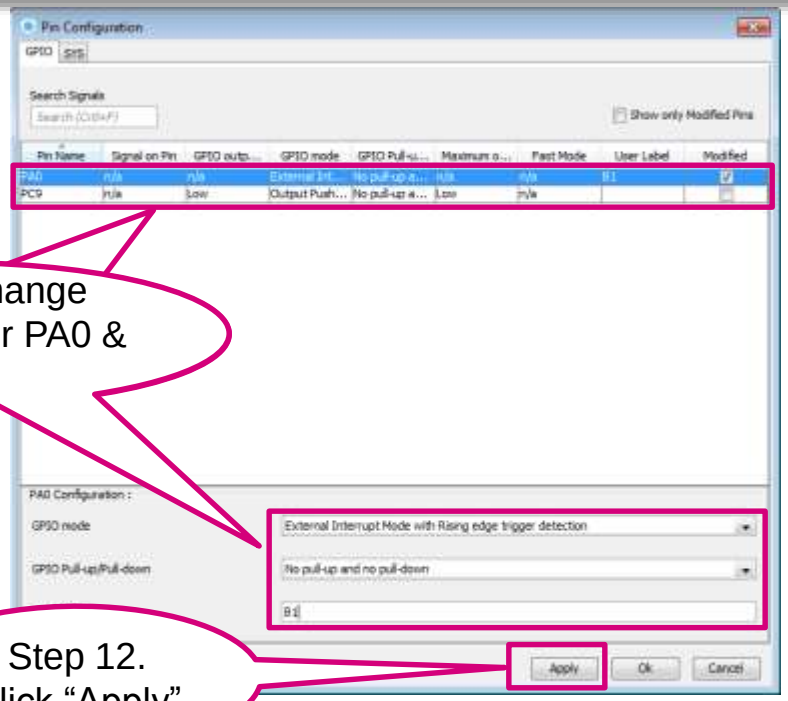
8 APB1 Timer clocks (MHz)

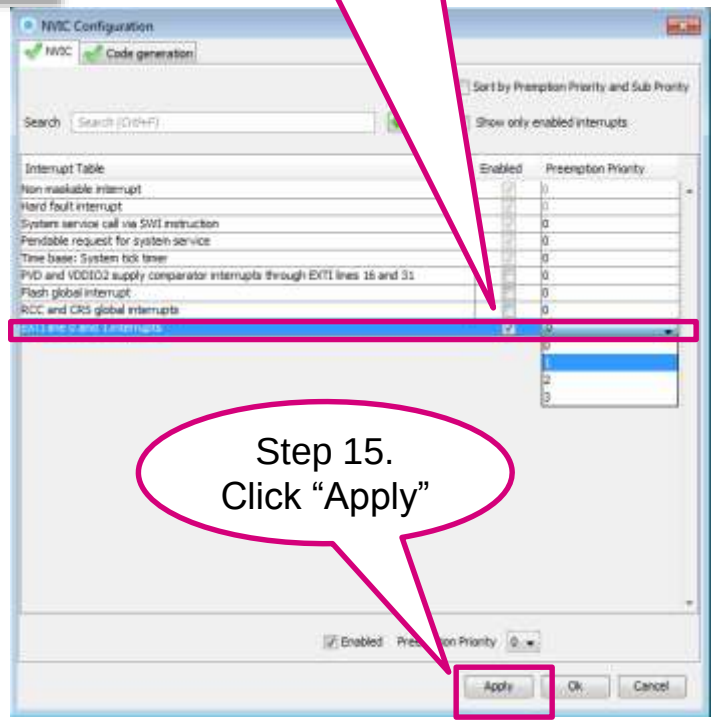
Lab GPIO_EXTI: STM32CubeMX Config cont.

- STM32CubeMX configuration cont.:
  - Configuration Tab
    - GPIO
      - PC9 (Output Push Pull mode, no pull-up/down, Fast output speed, User Label: LD_R (green))
      - PA0 (External Interrupt mode with the correct edge detection, no pull-up/down, User Label: B1 User) . Refer to schematics for the correct edge trigger.
      - PA13 (Set as SWD pins. No further action. User Label: SWDIO)
      - PA14 (Set as SWD pins. No further action. User Label: SWCLK)
    - NVIC
      - Enable External Line 4 to Line 15 interrupt with Software priority (Preemption Priortiy) set to 1.
      - System tick timer - Care must be taken when using HAL_Delay(), this function provides accurate delay (in milliseconds) based on variable incremented in SysTick ISR. This implies that if HAL_Delay() is called from a peripheral ISR process, then the SysTick interrupt must have higher priority (numerically lower) than the peripheral interrupt. Otherwise the caller ISR process will be blocked.
    - RCC – no further changes needed for this particular discovery board and exercise

life.augmented

Screenshots





life.augmented

Lab GPIO_EXTI: STM32CubeMX Project Setting

- Code Generator tab
  - STM32Cube Firmware Library Package
    - Select “Copy only the necessary library files”
  - Generated files
    - Select “ Keep User Code when re-generating” & “Delete previously generated files when not re-generated”
- Project tab
  - Project name
    - Enter the project name and the subdirectory
  - Toolchain/IDE
    - Select “MDK-ARM V5” for Keil V5 compiler

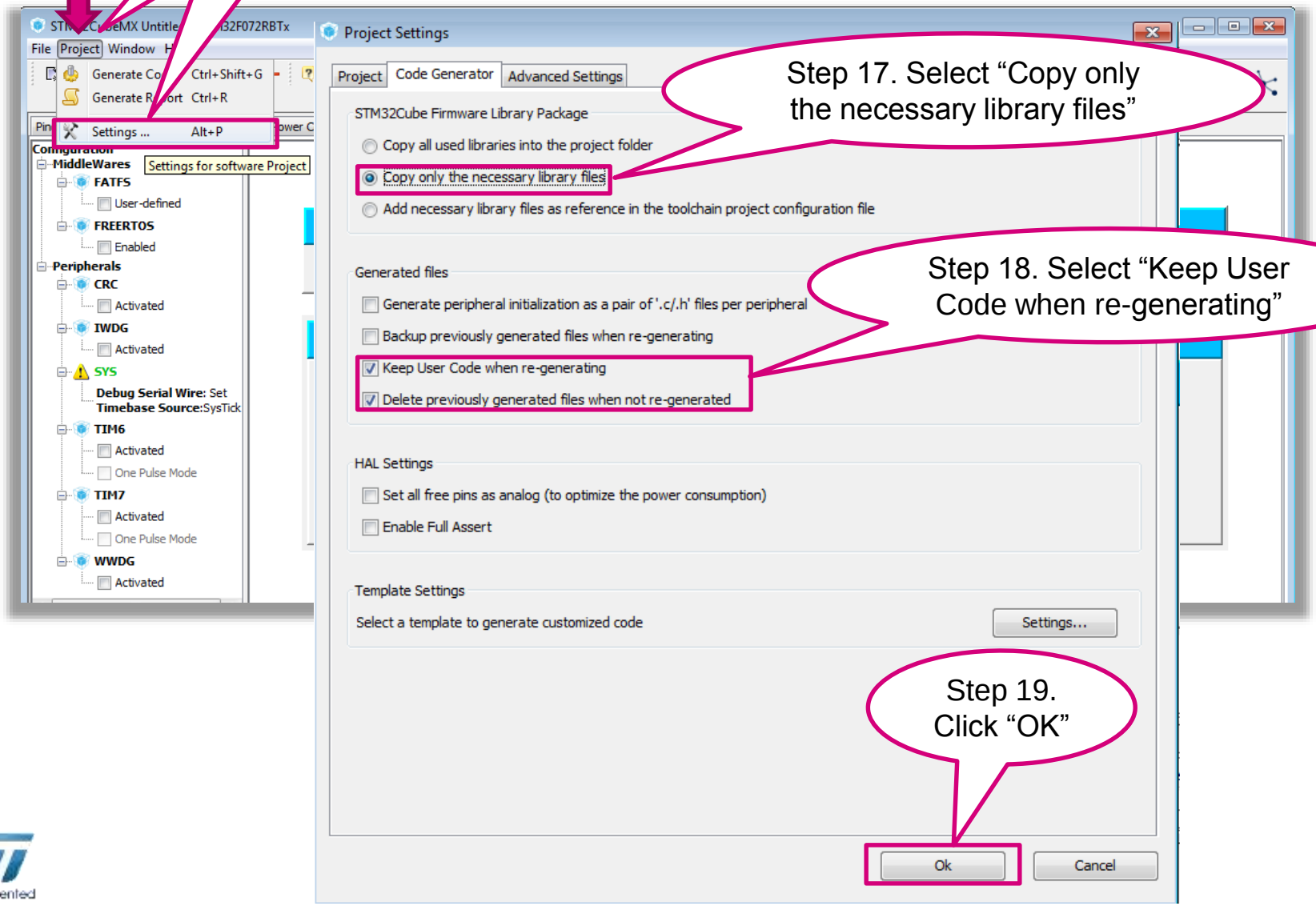
Screenshots

Step 16. Click "Project" and select "Setting"

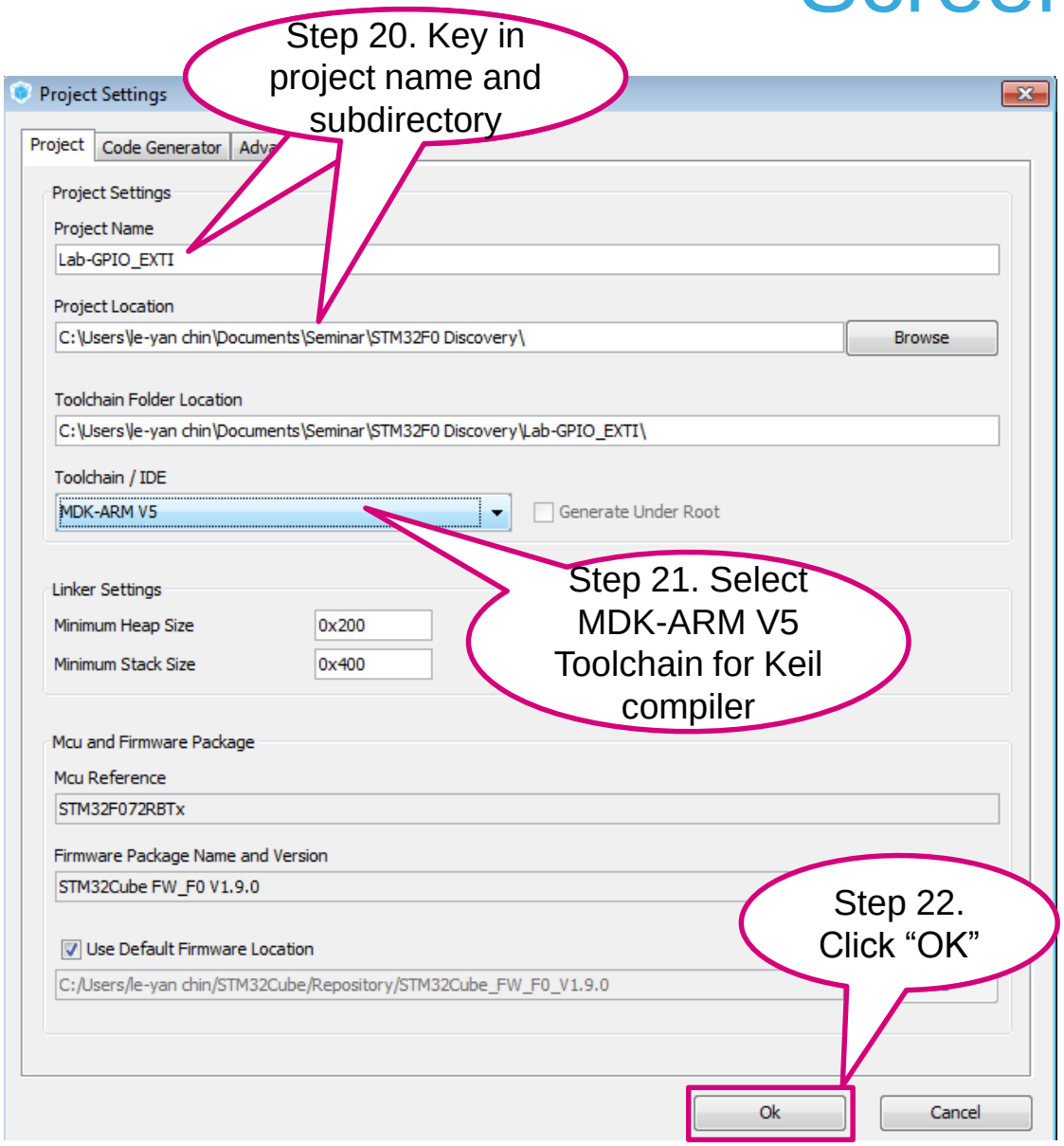
Step 17. Select "Copy only the necessary library files"

Step 18. Select "Keep User Code when re-generating"

Step 19. Click "OK"



Screenshots



Lab GPIO_EXTI: STM32CubeMX Project Setting

- Save the project once all configuration are done.
- To complete, perform the following:
  - Generate Report (optional)
    - This will create a .pdf, .txt, and .jpg file
  - Generate Code
    - This will generate a project based on the Toolchain/IDE selected and all the necessary user and library files.
- Open the KEIL MDK-ARM 5 Project (Lab-GPIO_EXTI.uvprojx)
  - When the Code Generation is done, just click “Open Project”. Or you can manually open from the specified folder.
  - \..\STM32F0 Discovery Exercises\Lab-GPIO_EXTI\MDK-ARM
- Now you are ready to write some codes.

Lab GPIO_EXTI: Firmware Modification

- In KEIL environment, open main.c file.
  - Study the generated GPIO configuration. Verify if the configurations done by the STM32CubeMX tool are correct.
    - In Main.c file -> MX_GPIO_Init ()
  - In the STM32CubeMX Pinout configuration, you have only configured PA0(User PB), PC9(LD_R), PA13(SWDIO) and PA14(SWCLK). How come there are configurations for the other pins? What are these configurations for?
- main.c
  - The STM32CubeMX tool only generates the initialization code, further modifications of the user files (e.g. main.c, stm32F0xx_hal_msp.c, stm32l0xx_it.c) are needed to complete the implementation.
  - In the main.c file, you will find sections for user code. Copy the highlighted codes below to the corresponding USER CODE sections in the main.c file.
  - It is important the codes are copied within the USER CODE sections. This will allow you to regenerate another initialization code using STM32CubeMX tool without deleting the user codes.

Lab GPIO_EXTI: Firmware modification cont.

- For USER CODE BEGIN 0/USER CODE END 0

```
/* USER CODE BEGIN 0 */
uint8_t MODE_SELECTION;
/* USER CODE END 0 */
```

- For USER CODE BEGIN 3 / USER CODE END3

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
  if(MODE_SELECTION==0){
    /* Toggle LEDs - Use the HAL functions from stm32l0xx_hal_gpio.c file */
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_9);    //LD_R (green) – PC9
    HAL_Delay(100); //100ms
  }
  else if(MODE_SELECTION==1){
    /* Turn OFF the LEDs */
    /* - Use the HAL functions from stm32f0xx_hal_gpio.c file */
    /* Hint: Highlight + right click on the function and use “Go to definition…” */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_RESET); //Turn off LD_R(green)
    HAL_Delay(100); //100ms
  }
```

Lab GPIO_EXTI : Firmware modification cont.

- Cont. For USER CODE BEGIN 3 / USER CODE END3

```
  else if(MODE_SELECTION==2){
    /* Turn ON the LED */
    /* - Use the HAL functions from stm32f0xx_hal_gpio.c file */
    /* Hint: Highlight + right click on the function and use “Go to definition…” */
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_9, GPIO_PIN_SET);          //LD_R (green) – PC9

    /* Delay - Use the HAL delay function from stm32l0xx_hal.c file */
    HAL_Delay(1000);                //1secs
  }
}
 /* USER CODE END 3 */
```

Lab GPIO_EXTI : Firmware modification cont.

- For USER CODE BEGIN 4/ USER CODE END 4

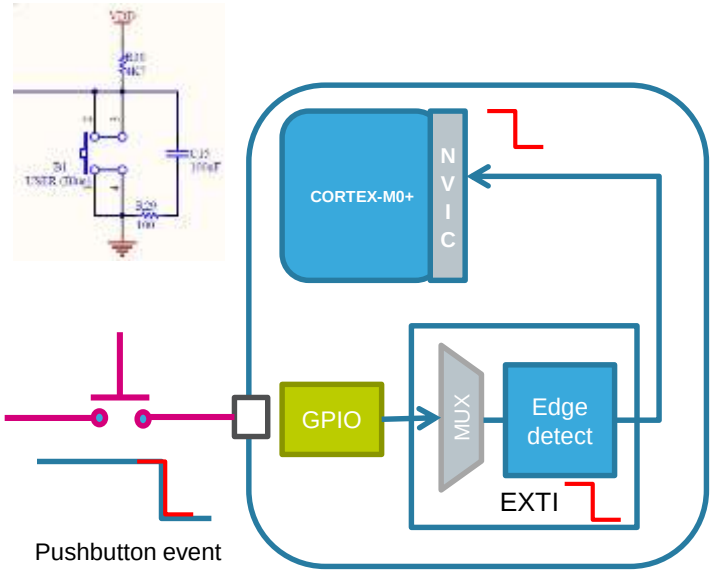
```
/* USER CODE BEGIN 4 */
/**
  * @brief EXTI line detection callback. The function will be call by EXTI0_IRQHandler in “stm32f0xx_it.c” .
  * @param GPIO_Pin: Specifies the pins connected EXTI line
  * @retval None
  */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin == GPIO_PIN_0)
  {
    MODE_SELECTION++;
    if(MODE_SELECTION > 2) MODE_SELECTION=0;
    /* Debounce - wait until the button is released . Read the GPIO to get the state. Refer to the schematics. */
    /* - Use the HAL functions from stm32l0xx_hal_gpio.c file */
    /* Hint: Highlight + right click on the function and use “Go to definition…” */
    while(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0) != GPIO_PIN_RESET);        //Blue pushbutton – PA0
  }

}
/* USER CODE END 4 */
```

Lab GPIO_EXTI: Verification

- Build, then Download and Debug
- Run the code
- Expected behavior:
  - When User button is pressed an interrupt is triggered and will call the EXTI IRQ handler in stm32l0xx_it.c file. The IRQ handler will then call the HAL_GPIO_EXTI_Callback() function in main.c file where the global variable (MODE_SELECTION) will be incremented.
  - MODE_SELECTION == 0 (Default), Green LED will toggle
  - MODE_SELECTION == 1, Green LED will turn off.
  - MODE_SELECTION == 2, Green LED will turn on.

Lab GPIO_EXTI: Discussion (Interrupts)

- Flow of interrupt
  - Pushbutton event occurs
  - EXTI detects valid edge
  - EXTI generates interrupt request
  - If the interrupt channel is enabled, the NVIC will acknowledge the interrupt request and checks the priority
  - When priority is higher, NVIC fetches EXTI Line interrupt vector. (Otherwise the interrupt will be set as pending until its priority becomes the highest compared to other pending interrupts)
  - Core executes EXTI IRQ Handler. Note that the handler will eventually call a callback function where the user will have to add and write the corresponding service routine.



In main.c

```
Main()
{
  …
  …
  while(1)
  {
    …
    …
    …
    …
  }
}
```

Interrupt trigger

In stm3fl0xx_it.c

```
void EXTI0_1_IRQHandler(void)
{
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

In stm32f0xx_hal_gpio.c

```
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if(__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != RESET)
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);
    HAL_GPIO_EXTI_Callback(GPIO_Pin);
  }
}
```

In main.c

```
HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  …  User Code to manage the
  …  interrupt
  …
  …
}
```

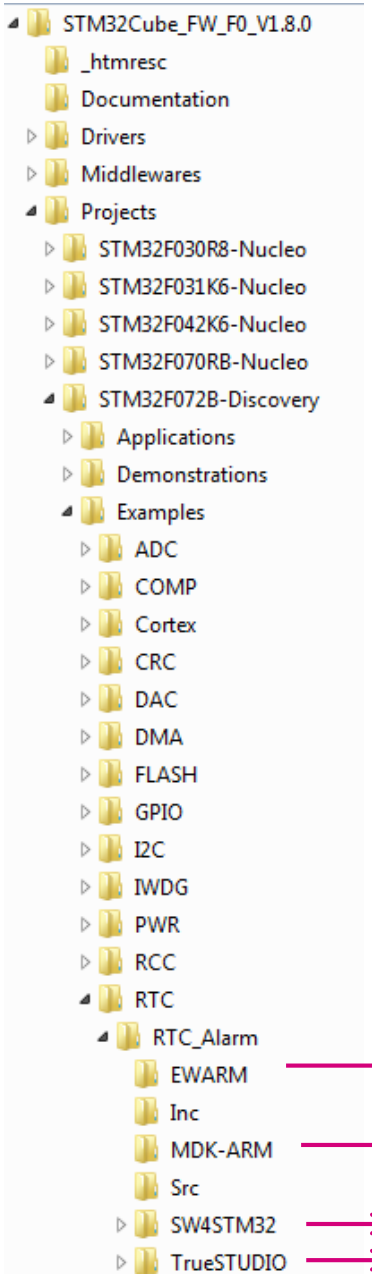
This flow of xxx_IRQhandler calls and xxx_Callback calls is similarly implemented for the other peripherals when interrupt request is enabled.

Demo 1: Running Sample code on the STM32F072 Discovery Board

life.augmented

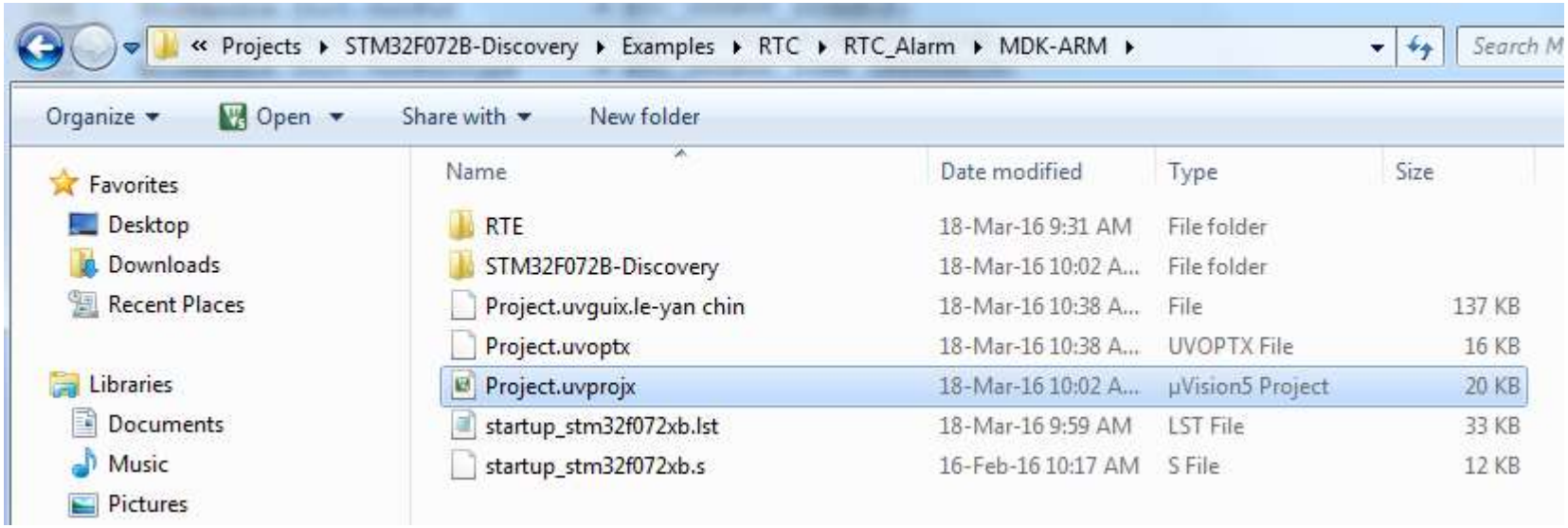
Hands-on sample code: RTC Alarm

- Objective:
  - Understand where to look for example codes in STM32Cube
  - Run “RTC_Alarm” example code on STM32F072 Discovery board which trigger turning on LED LD5 when the clock reaches the alarm setting
- Description:
  - LED LD5 is connected to PC9 and and RTC is set to 00:20:00 at initial
  - Alarm will be generated after 30 seconds on 02:20:30 and turns on LED LD5
- Procedure:
  - Use window explorer to locate \STM32Cube\Repository subdirectory
    - eg> c:\users\le-yan chin\STM32Cube\Repository
  - Check for the package F0 firmware
    - eg> STM32Cube_FW_F0_V1.8.0
  - Copy the whole directory content to other place to retain the original firmware package
  - Go to respective \Project subdirectory you copied and click into \STM32F072-Discovery subdirectory
    - eg> C:\Users\..\STM32Cube_FW_F0_V1.8.0\Projects\STM32F072B-Discovery
  - Select “RTC_Alarm” for Keil toolchain in \..\Examples\RTC\RTC_Alarm\MDK-ARM subdirectory
    - eg> C:\Users\..\Projects\STM32F072B-Discovery\Examples\RTC\RTC_Alarm\MDK-ARM



Hands-on sample code: RTC Alarm (cont'1)

- Procedure:
  - Click to run the "Project.uvprojx" for Keil toolchain platform



Hands-on sample code: RTC Alarm (cont'2)

- Procedure:
  - In Keil IDE, click the "Rebuild" () icon to rebuild all the target files in the project

C:\Users\le-yan chin\Documents\Training\STM32F0 RTOS seminar 6May 2016\STM32Cube_FW_F0_V1.5.0\Projects\STM32F072B-Discovery\Examples\RTC\RTC_Alarm\MDK-

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

aShowTime

STM32F072B-Discovery

Project

Rebuild
Rebuild all target files

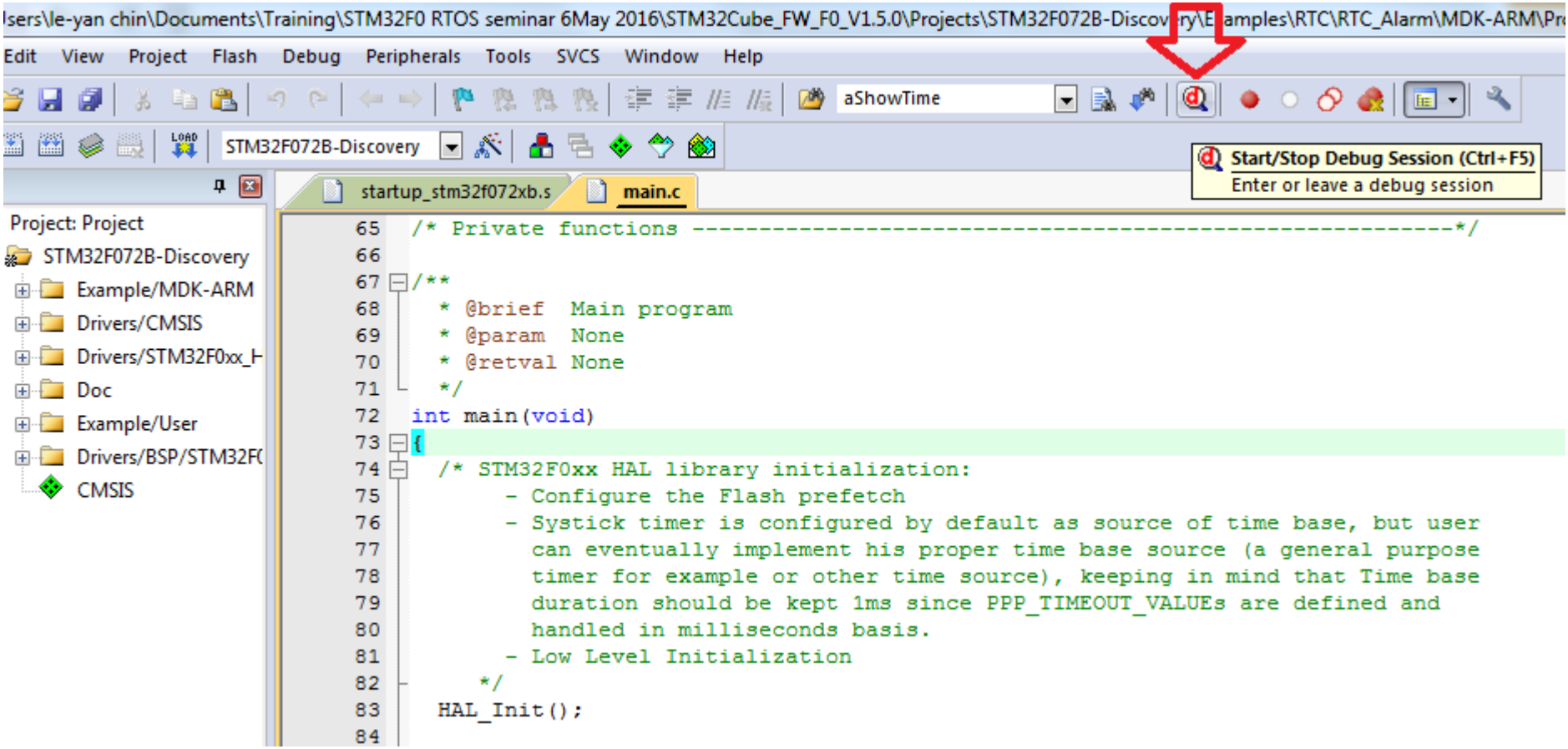
Proj
STM32F072B-Discovery
Example/MDK-ARM
Drivers/CMSIS
Drivers/STM32F0xx_H
Doc
Example/User
Drivers/BSP/STM32F0
CMSIS

startup_stm32f072xb.s | main.c

```
/* Private functions ---------------------------------------------------------*/

/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* STM32F0xx HAL library initialization:
       - Configure the Flash prefetch
       - Systick timer is configured by default as source of time base, but user
         can eventually implement his proper time base source (a general purpose
         timer for example or other time source), keeping in mind that Time base
         duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
         handled in milliseconds basis.
       - Low Level Initialization
     */
  HAL_Init();
```

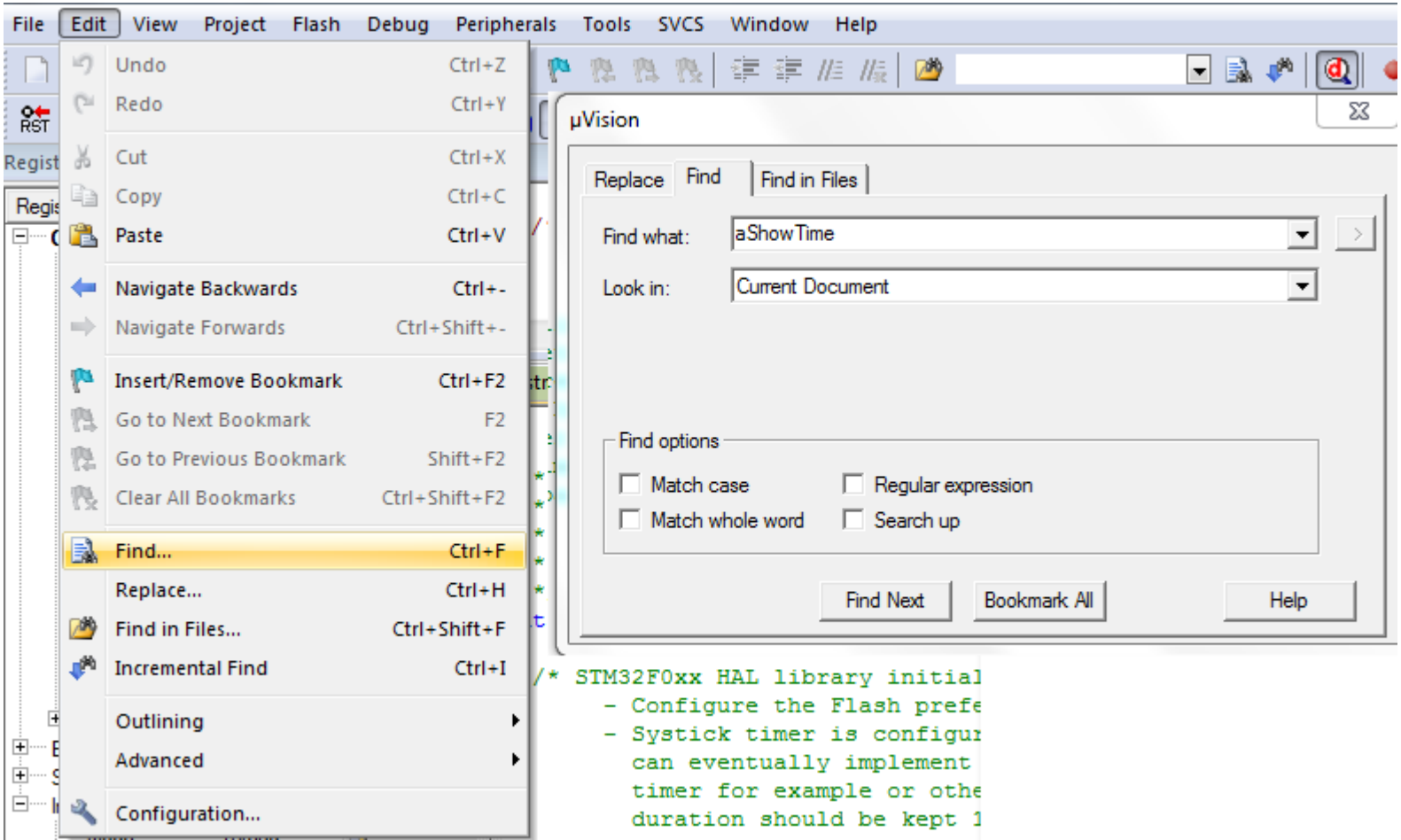
Hands-on sample code: RTC Alarm (cont'3)

- Procedure:
  - When build finish without error, click the "Start/Stop Debug" () icon to go into debug mode



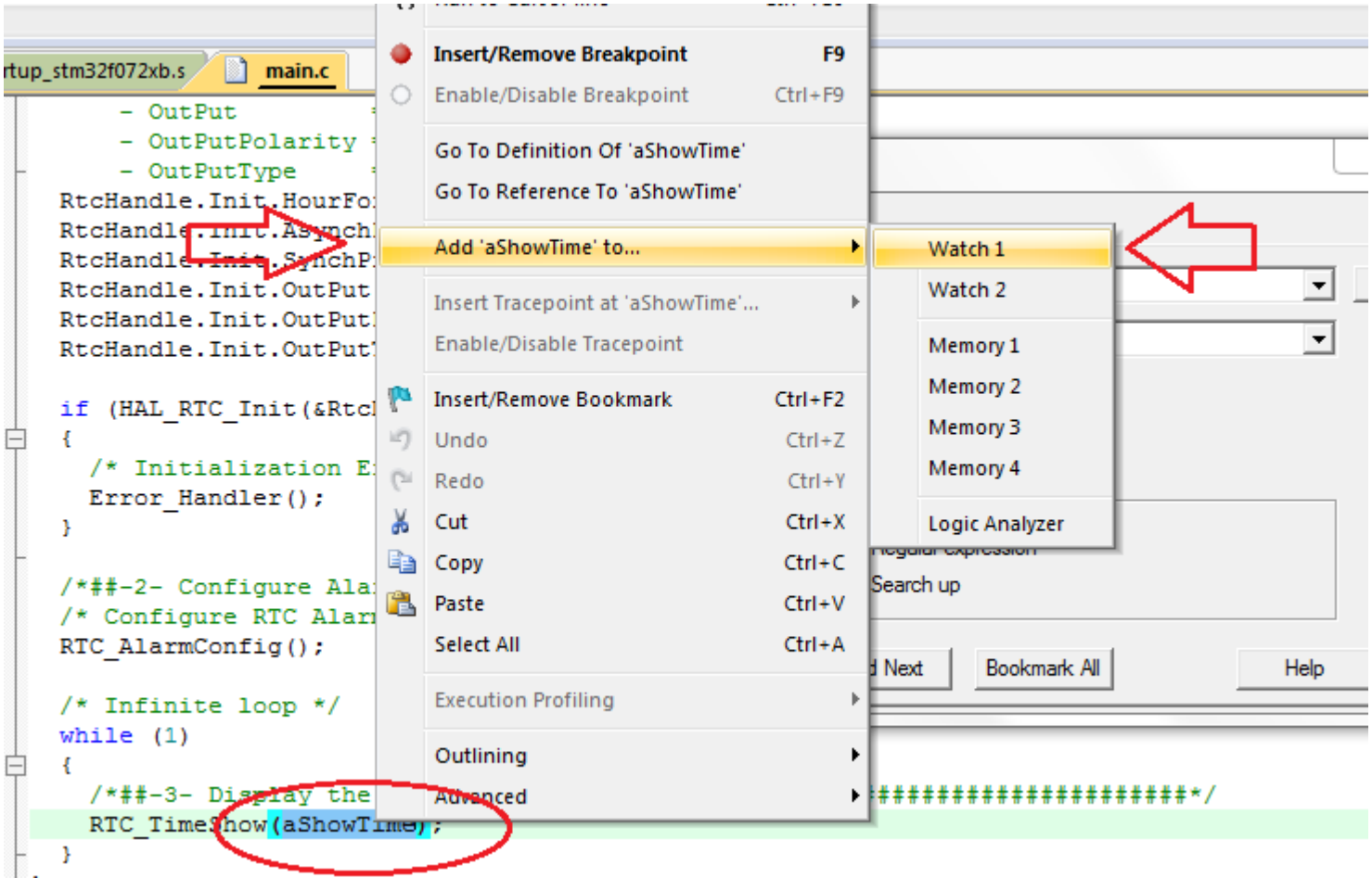
Hands-on sample code: RTC Alarm (cont'4)

- Procedure:
  - In Debug mode use the "Find" command to locate "aShowTime" variable



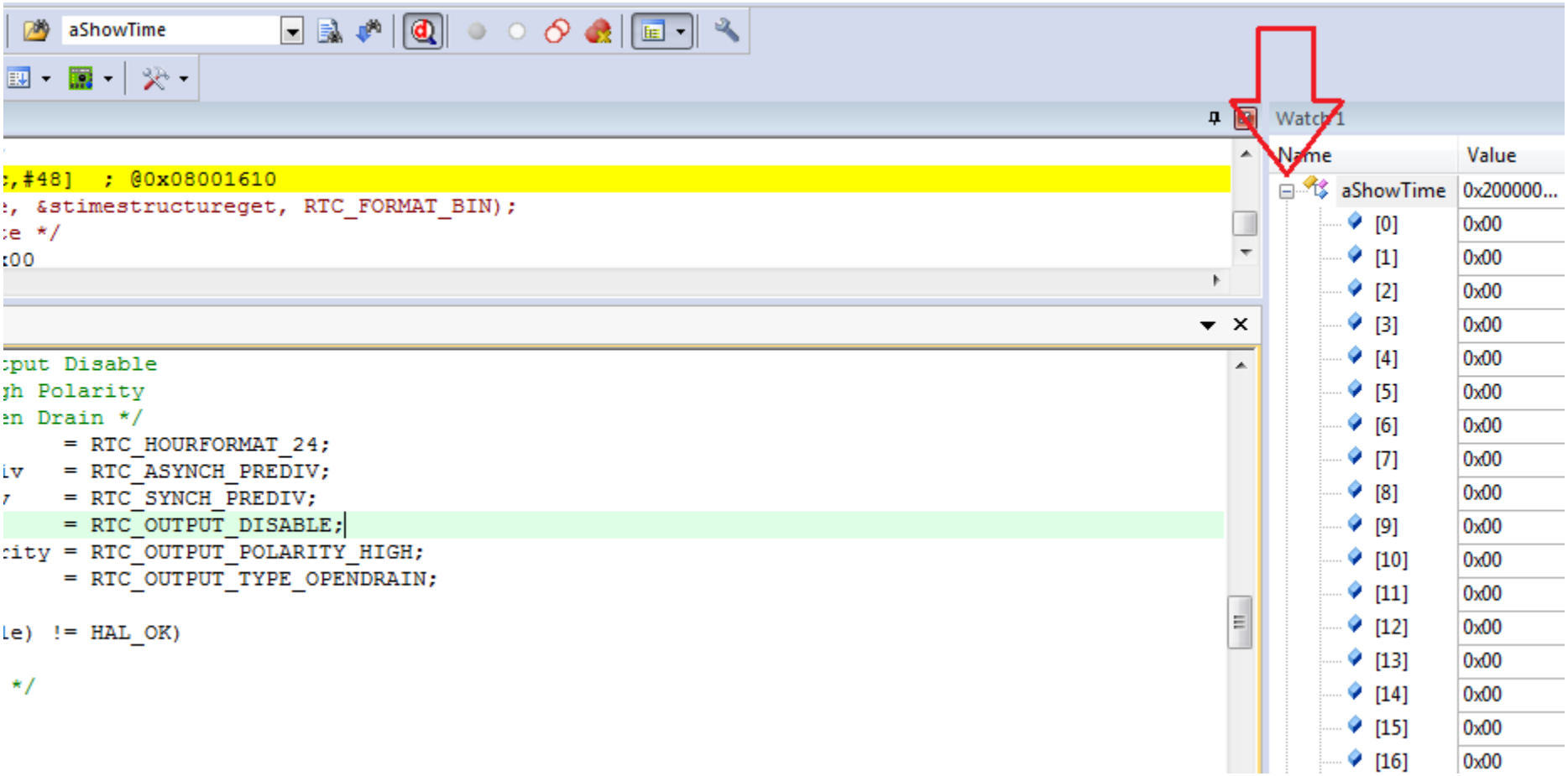
Hands-on sample code: RTC Alarm (cont'5)

- Procedure:
  - Move the cursor pointer to the variable "aShowTime", right click the mouse and select {Add "aShowTime" } to "Watch 1"



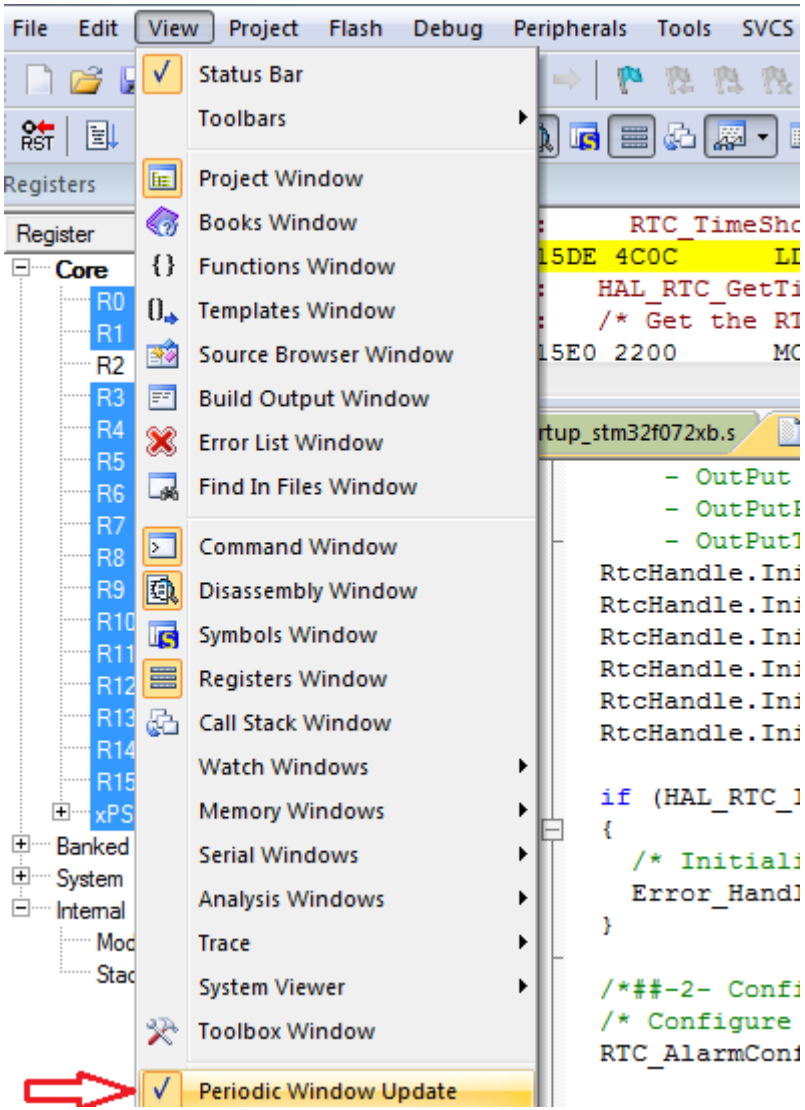
Hands-on sample code: RTC Alarm (cont'6)

- Procedure:
  - Expand the variable "aShowTime" in the Watch 1 to see detail value



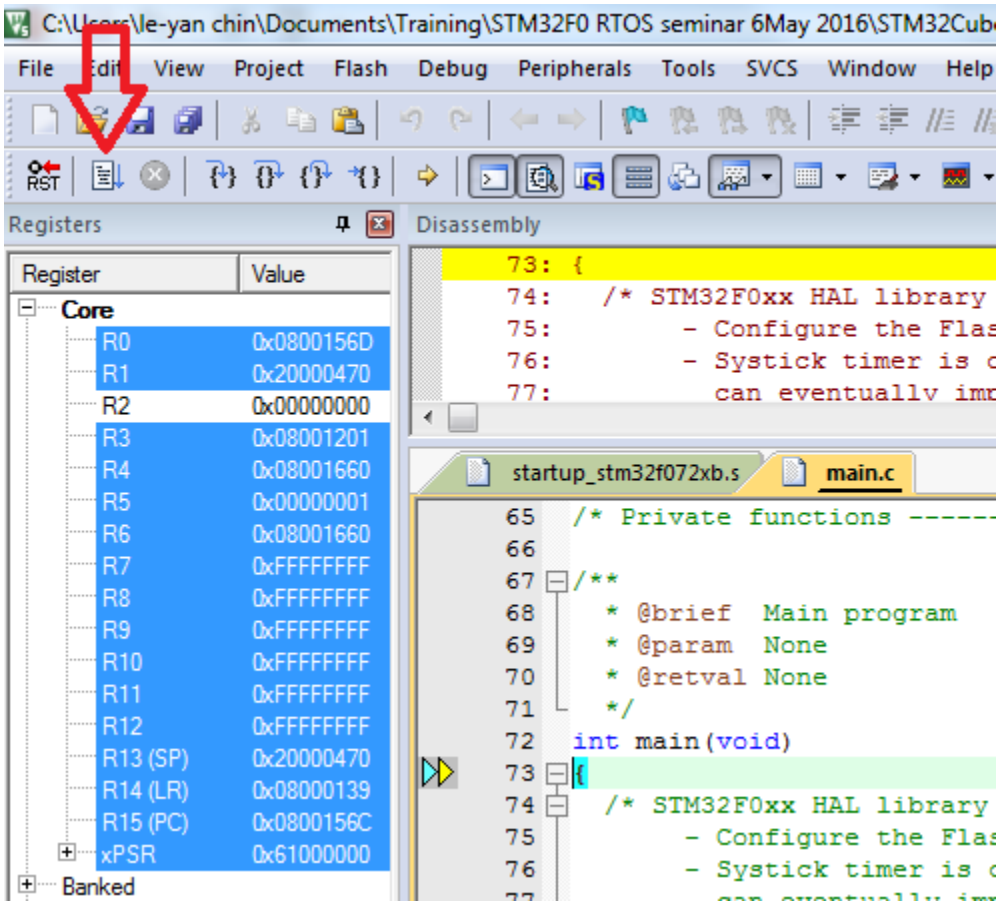
Hands-on sample code: RTC Alarm (cont'7)

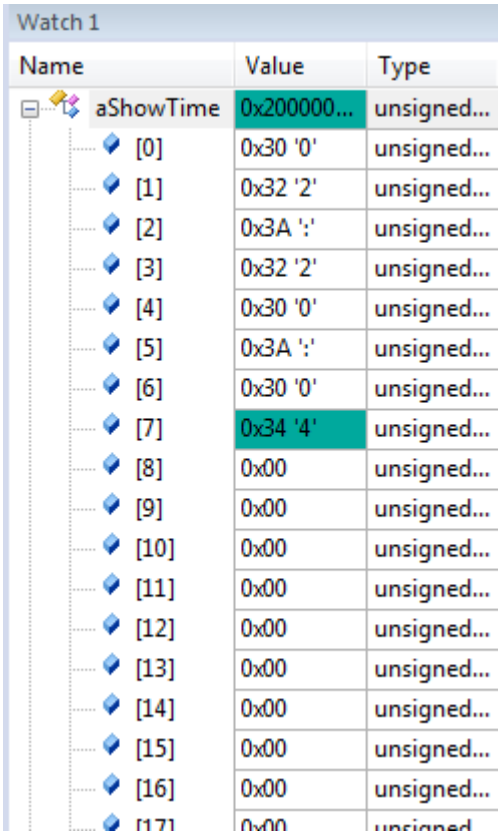
- Procedure:
  - Click on "View" menu and check the "Periodic Window Update"



Hands-on sample code: RTC Alarm (cont'8)

- Procedure:
  - Click the "Run" () icon to run the project, observe the eShowTime variable change in Watch 1 window





Hands-on sample code: RTC Alarm (cont'9)

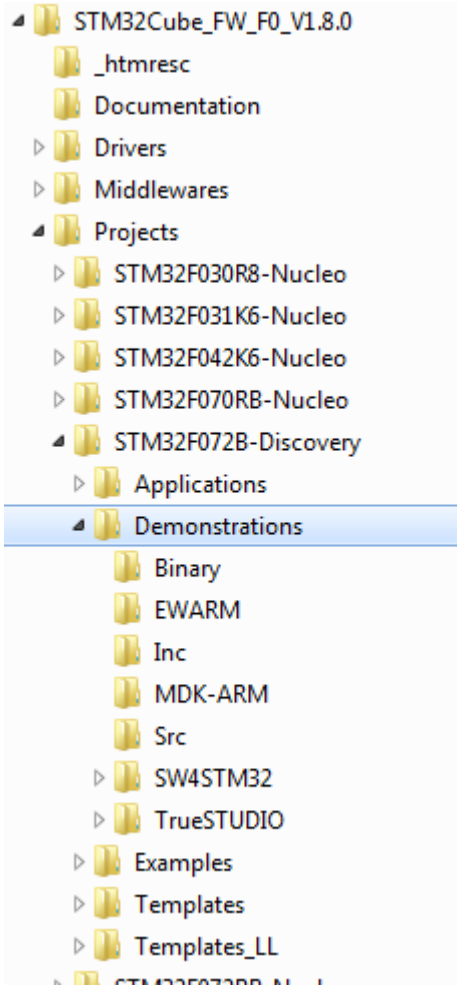
- Procedure:
  - a) Try to modify the code only to turn on the LED LD5 when eShowTime is 02:20:20
    - Hint : look for "salarmstructure.AlarmTime.Seconds"
  - b) Try to change the start up time from 12:00:00
    - Hint : look for "stimestructure"

Demo 2: Running the Demonstration project on the STM32F072 Discovery Board

life.augmented

Hands-on sample code: Demostrations

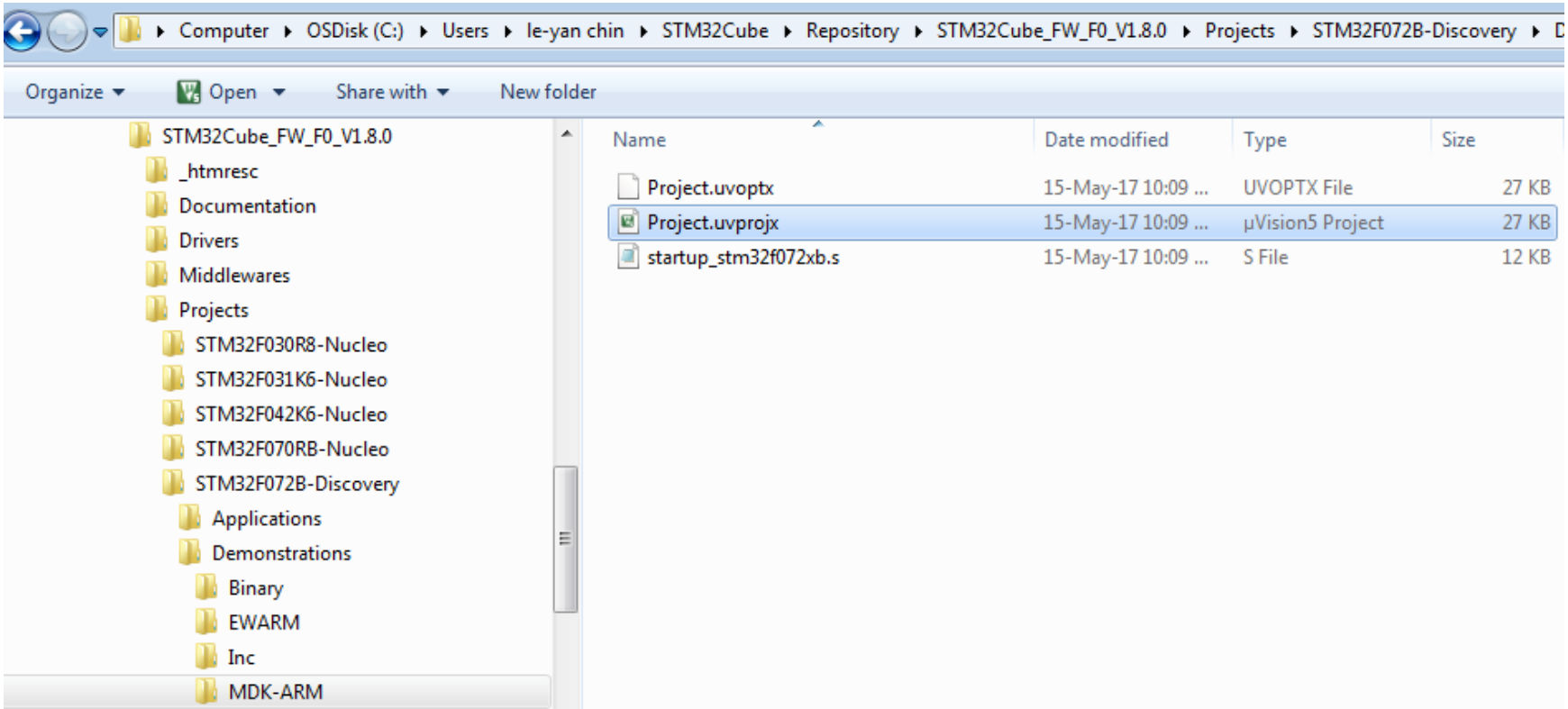
- Objective:
  - Understand where to look for example codes in STM32Cube
  - Run “Demostrations” example code of STM32F072 on Discovery board to test the touchsense button, USB HID interface and Gyro sensor.
- Description:
  - LED LD5 is connected to PC9 and 4 touchsense buttons are connected to TS_G1_IO3,TS_G1_IO4, TS_G2_IO3,TS_G2_IO4,TS_G3_IO2, TS_G3_IO3
  - Using User button to select different application options in the project such as Sliding position, HID, Gyro movement.
- Procedure:
  - Use window explorer to locate \STM32Cube\Repository subdirectory
    - eg> c:\users\le-yan chin\STM32Cube\Repository
  - Check for the package F0 firmware
    - eg> STM32Cube_FW_F0_V1.8.0
  - Copy the whole directory content to other place to retain the original firmware package
  - Go to respective \Project subdirectory you copied and click into \STM32F072-Discovery subdirectory
    - eg> C:\Users\..\STM32Cube_FW_F0_V1.8.0\Projects\STM32F072B-Discovery
  - Select “Demonstrations” subdirectory in \STM32F072-Discovery subdirectory
    - eg> C:\Users\..\STM32Cube_FW_F0_V1.8.0\Projects\STM32F072B-Discovery\Demonstrations



Hands-on sample code: Demostrations (cont'1)

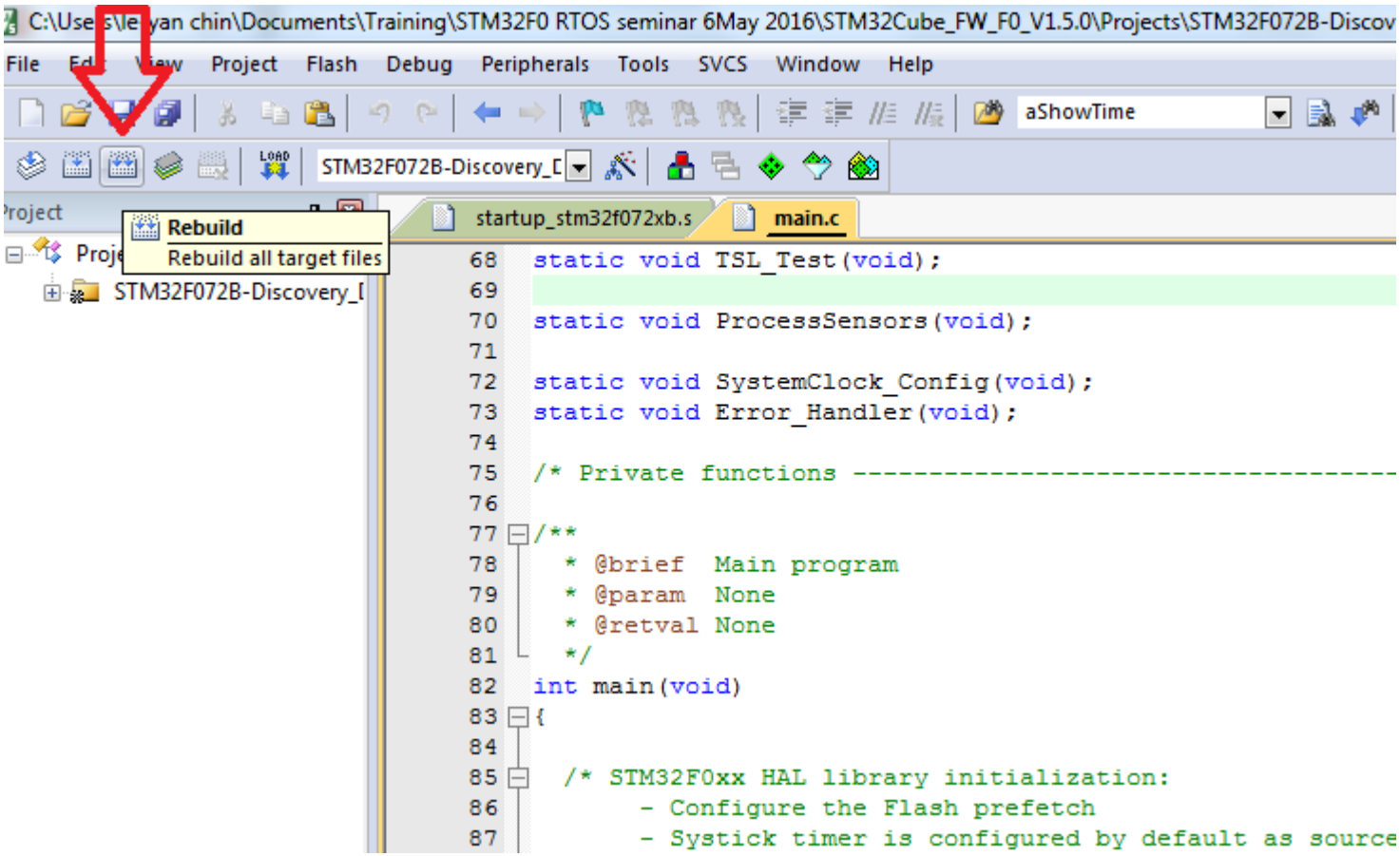
- Procedure:
  - Select project file "Project.uvprojx" to run on Keil toolchain

    eg> C:\Users\..\STM32Cube_FW_F0_V1.8.0\Projects\STM32F072B-Discovery\Demonstrations\MDK-ARM\project.uvprojx



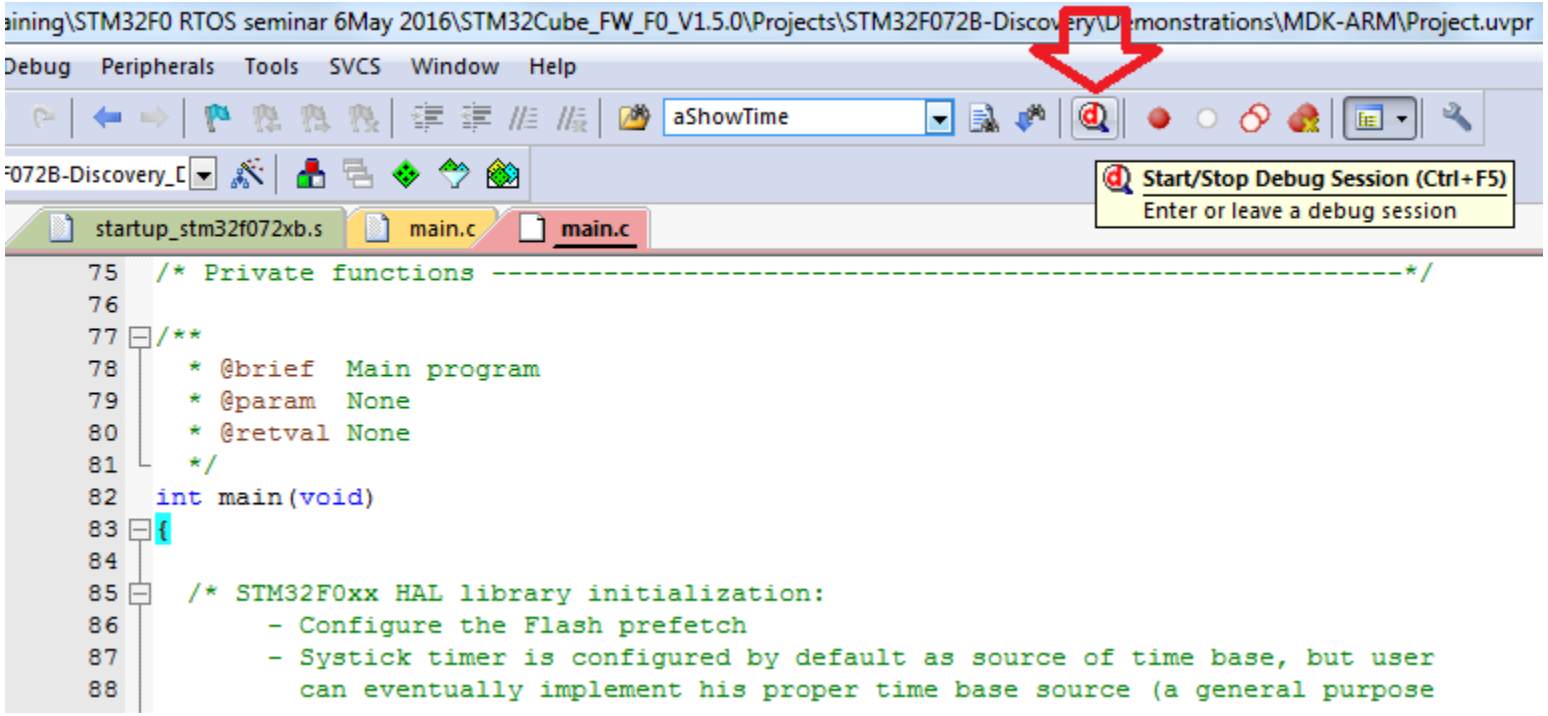
Hands-on sample code: Demostrations (cont'2)

- Procedure:
  - In Keil IDE, click the "Rebuild" () icon to rebuild all the target files in the project.



Hands-on sample code: Demostraions (cont'3)

- Procedure:
  - When build finish without error, click the "Start/Stop Debug" () icon to go into debug mode



Hands-on sample code: Demostrations (cont'4)

- Procedure:
  - Click the "Run" () icon to run the project.

C:\Users\le-yan chin\Documents\Training\STM32F0 RTOS seminar 6May 2016\STM32Cube_FW_F0_V1.5.0\Projects\STM32F072B-Discovery\De

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

aShowTime

Registers

Run (F5)
Start code execution

Disassembly

```
83: {
84:
85:   /* STM32F0xx HAL library initialization:
86:        - Configure the Flash prefetch
87:        - Systick timer is configured by default as source of
```

Register

Core	
R0	0x08004749
R1	0x20000FF0
R2	0x00000000
R3	0x0800473B
R4	0x080049E4
R5	0x00000001
R6	0x080049E4
R7	0xFFFFFFFF
R8	0xFFFFFFFF
R9	0xFFFFFFFF
R10	0xFFFFFFFF
R11	0xFFFFFFFF
R12	0xFFFFFFFF
R13 (SP)	0x20000FF0
R14 (LR)	0x08000345
R15 (PC)	0x08004748
xPSR	0x61000000
Banked	

startup_stm32f072xb.s | main.c | main.c

```
75  /* Private functions ---------------------------------------
76
77  /**
78    * @brief  Main program
79    * @param  None
80    * @retval None
81    */
82  int main(void)
83  {
84
85    /* STM32F0xx HAL library initialization:
86         - Configure the Flash prefetch
87         - Systick timer is configured by default as source of
```

Hands-on sample code: Demostrations (cont'5)

- Procedure:
  - Press User button B1 1st time
  - Change the Discovery board level position to see the LEDs on change.
    - eg LED LD4 turns on shows the board level swing to left, opposite swing direction will turn on LED LD5; similarly LED LD3 turns on shows board level swing to front and opposite direction will turn on LED LD6
  - Press User button B1 2nd time, LED LD3 and LD6 turns on
  - Connect a mini-USB cable at the USB User socket
  - The movement of the Discovery board will take control of the PC mouse pointer
  - Press User button B1 3rd time, all LED turns off
  - Slide the touchsense buttons and observe the LEDs on/off changes; indicating the position of the finger sliding

life.augmented