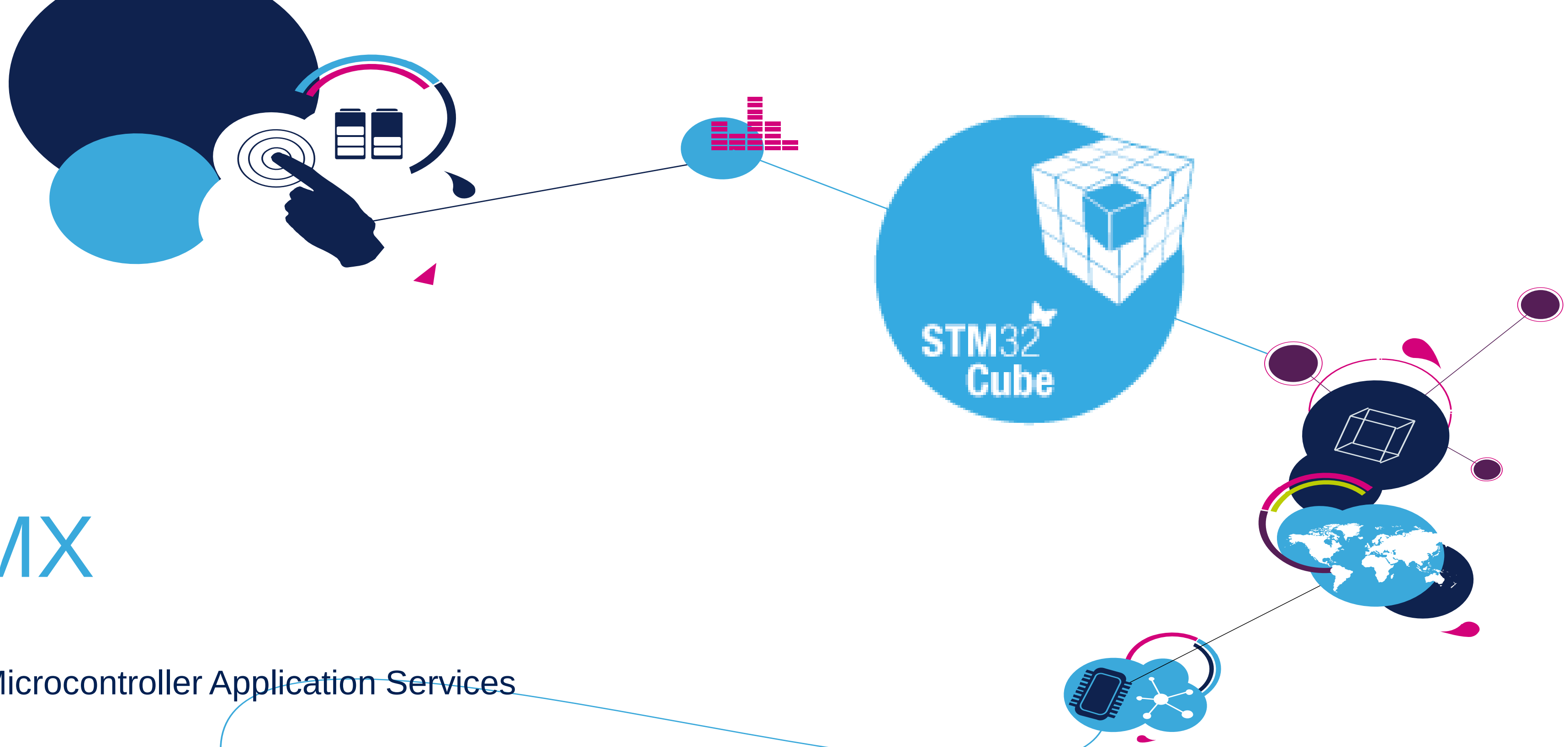


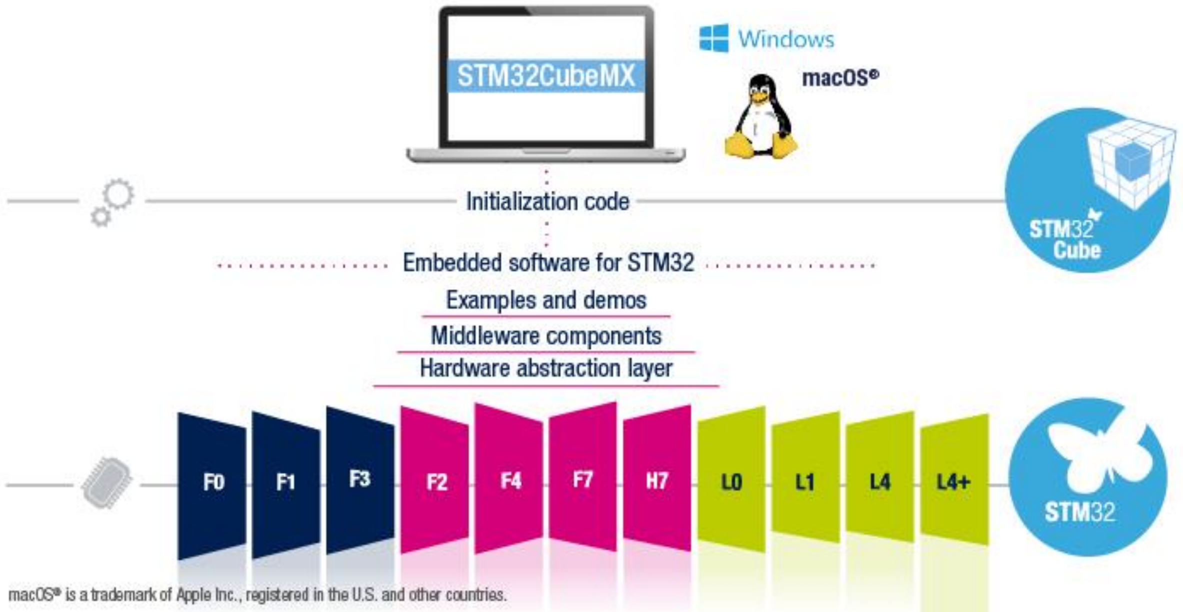
STM32CubeMX

T.O.M.A.S – Technically Oriented Microcontroller Application Services
v0.04MC 16:9

life.augmented

CubeMX install

- CubeMX tool
  - http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?s_searchtype=partnumber
- The CubeMX tool need java
  - Please check if you have last java on your pc, for sure 32bit and 64bit version
- How to solve problems with installation look into User manual UM1718, into FAQ section
  - http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?s_searchtype=partnumber

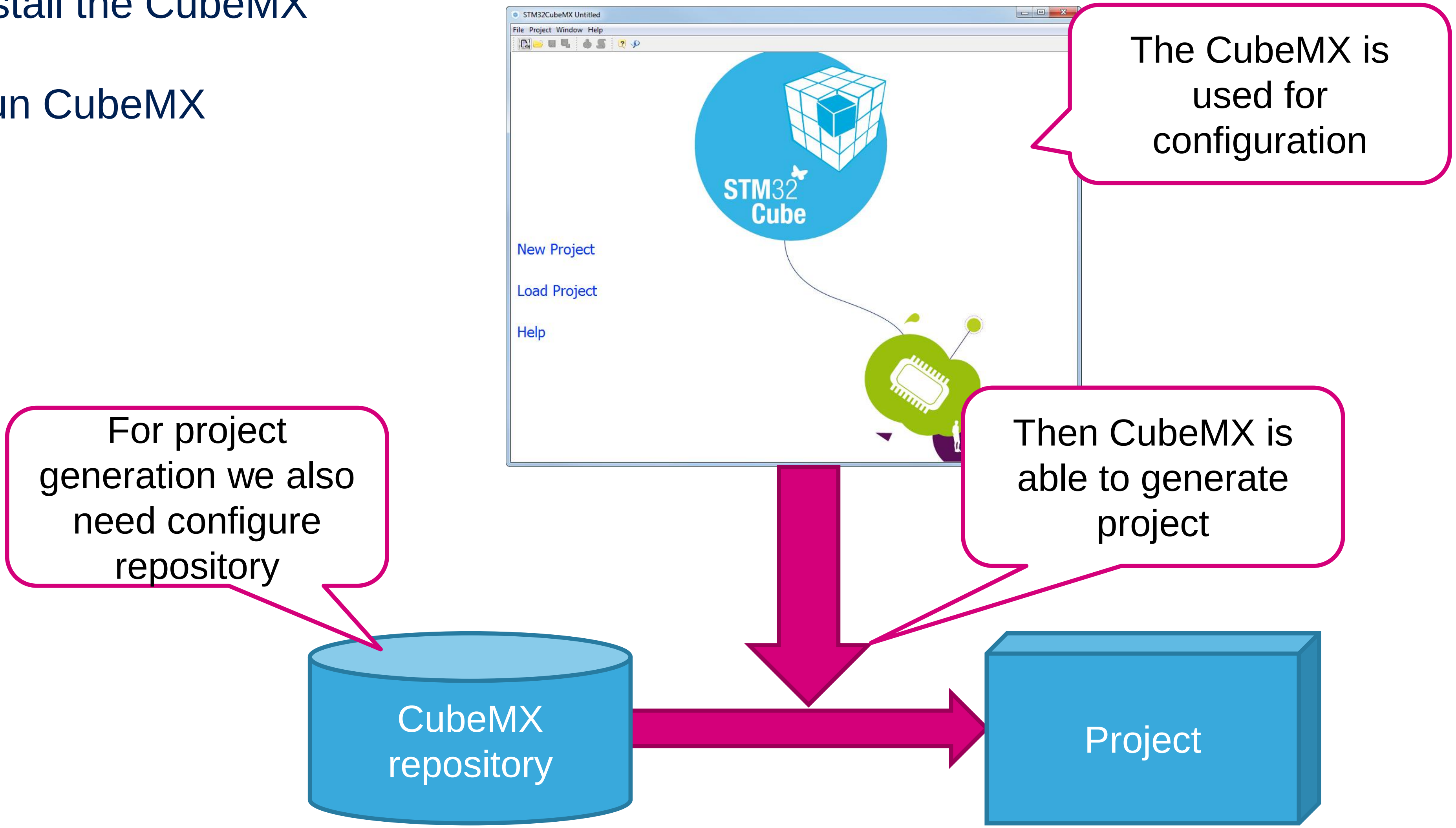


CubeMX install

- The CubeMX can generate the code for some IDE
  - Atollic TrueSTUDIO
  - IAR
  - Keil
  - System Workbench
- For the debugging is necessary to have the ST-Link drivers
  - STSW-LINK009 driver for Win XP/Vista/7/8/10
    http://www.st.com/web/en/catalog/tools/PF260219
- For driver installation you will need the **Admin rights** on your PC

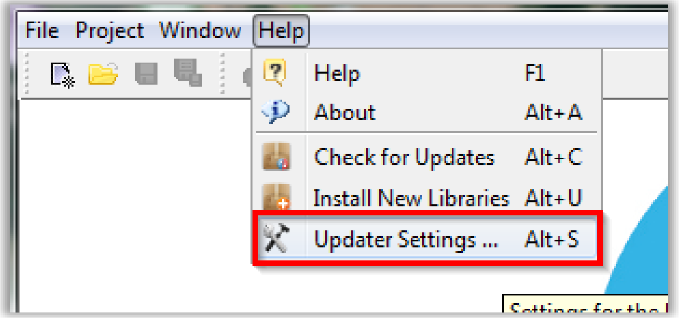
CubeMX repository configuration

- Install the CubeMX
- Run CubeMX

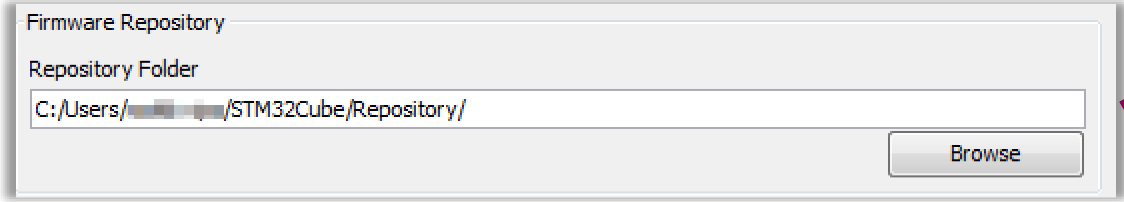


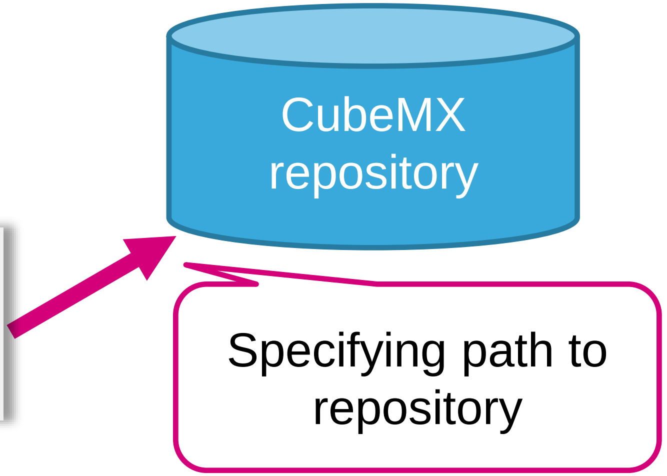
CubeMX repository configuration

- In case you download the package from web we need to find the place where they need to be stored



- MENU>Help>Updater Settings…
- You will see where is the repository folder
  - Default is C:/User/Acc_name/STM32Cube/Repository/
  - **In case that you have in your repository path diacritics, the CubeMX may not work properly, please change you repository path (ex: C:/Repository)**
- You need to download STM32 packages into this folder
- Or CubeMX automatically download them into this folder

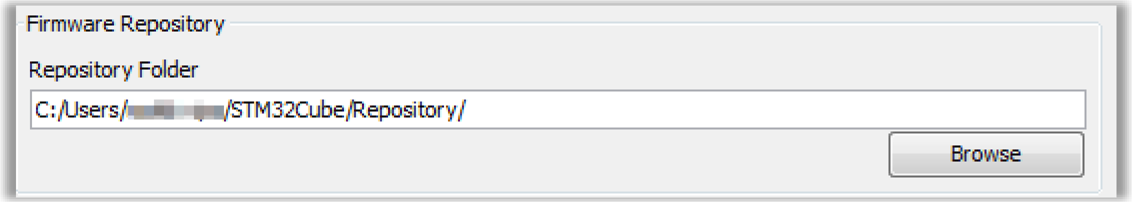


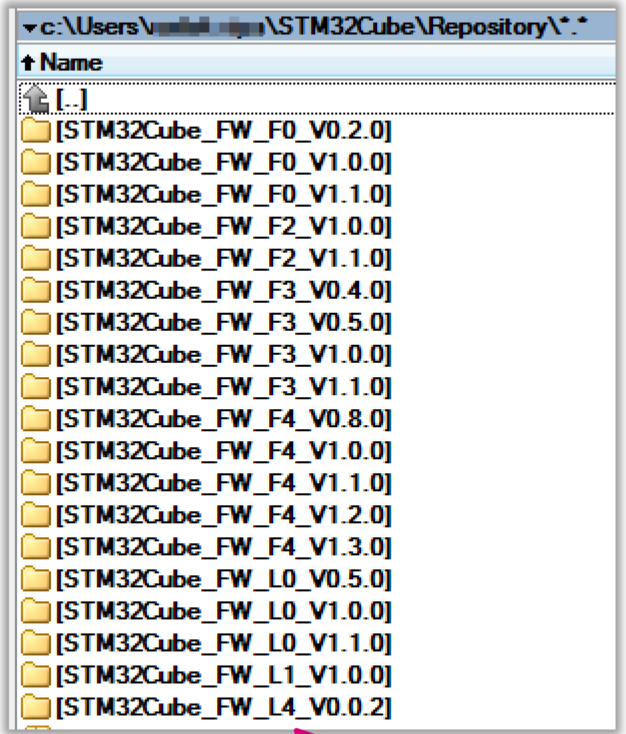


The "/" in the end of Repository is **necessary**

CubeMX repository configuration

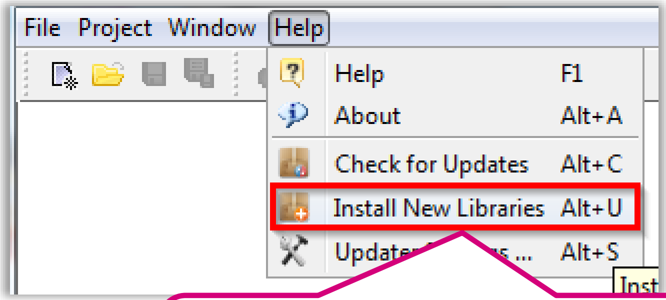
- The comparison of the CubeMX repository settings and structure in this folder





Example how the repository structure looks like

- In case you want to download this files automatically use in CubeMX
  - MENU>Help>Install New Libraries
  - Select libraries which you want
  - Force download with button Install Now



CubeMX can download for you the repository packages automatically

life.augmented

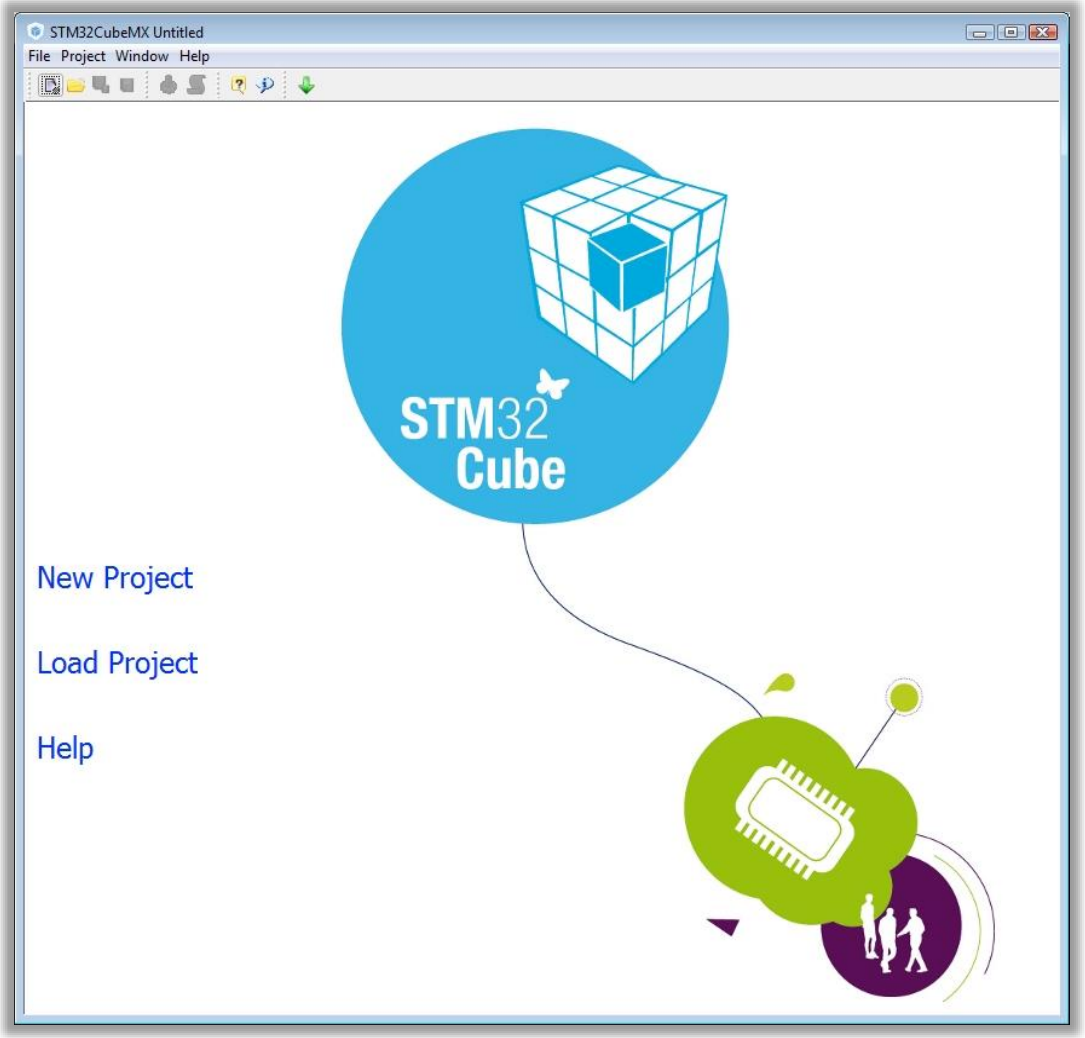
STM32 Cube

STM32CubeMX presentation

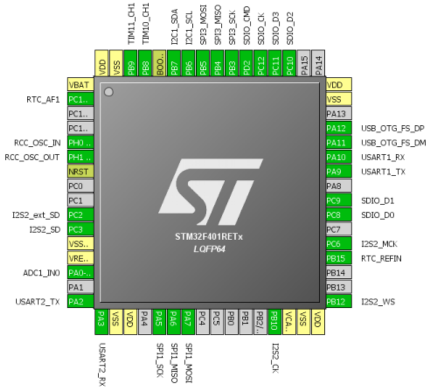
STM32Cube: STM32CubeMX

Step by step:

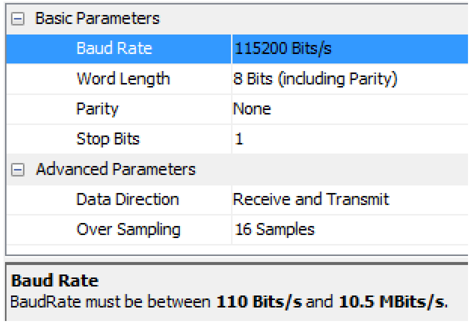
- MCU selector
- Pinout configuration
- Clock tree initialization
- Peripherals and middleware parameters
- Code generation
- Power consumption calculator



STM32CubeMX



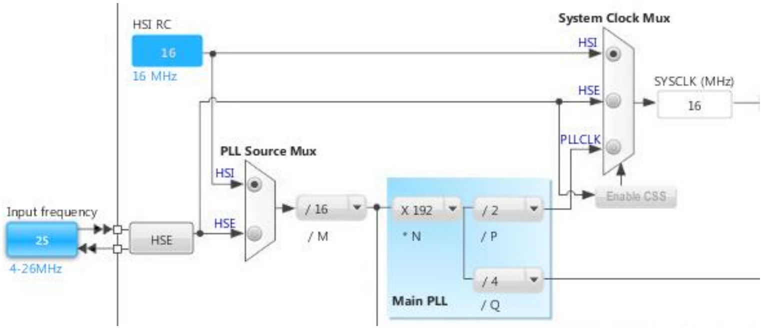
Pinout Wizard



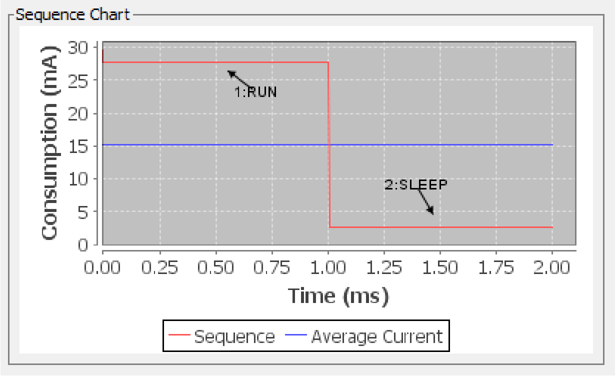
Peripherals & Middleware Wizard

STM32 Cube

Clock Tree wizard



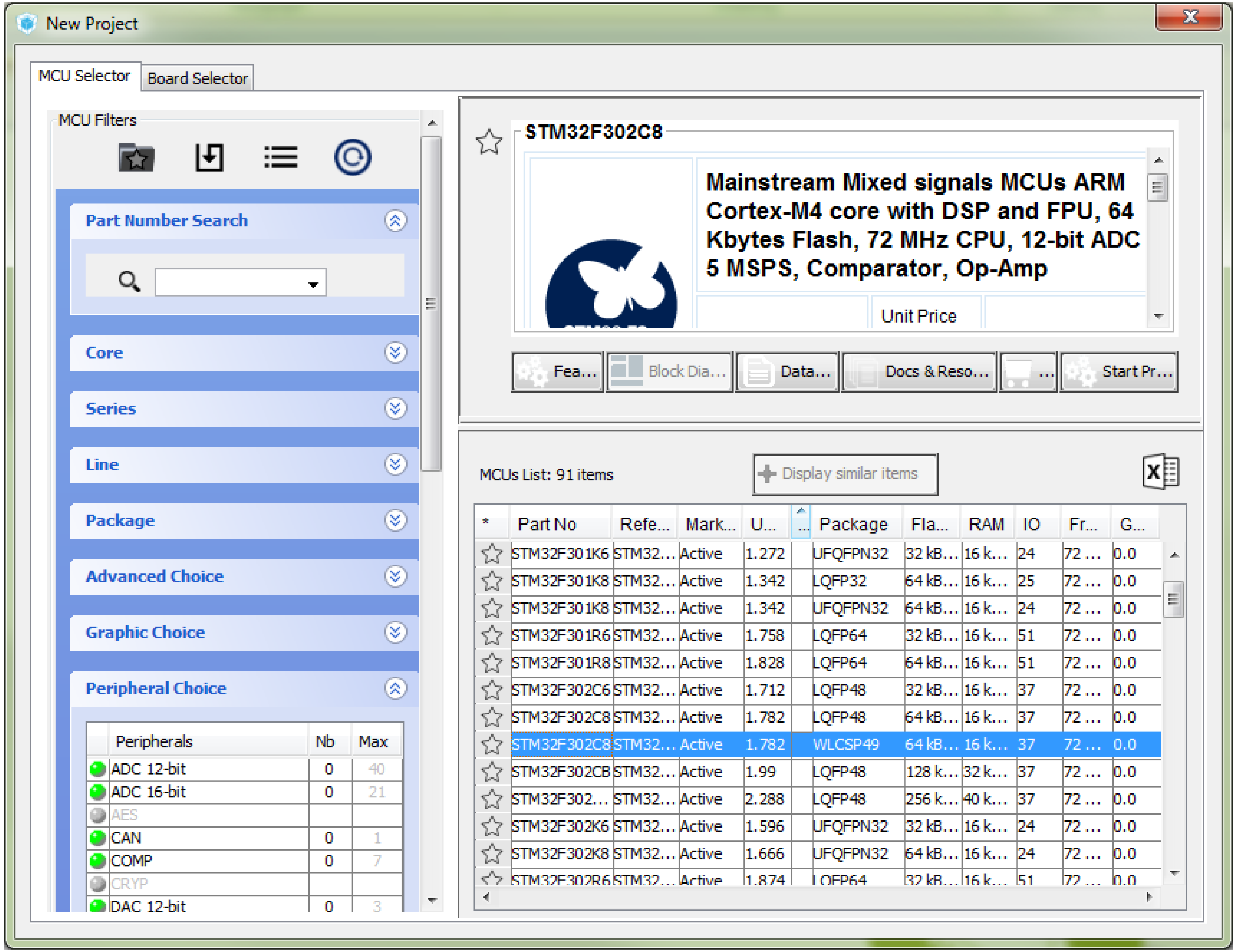
Power Consumption Wizard



STM32CubeMX: MCU Selector

Easy Optional filtering:

- Series
- Line
- Package
- Peripherals

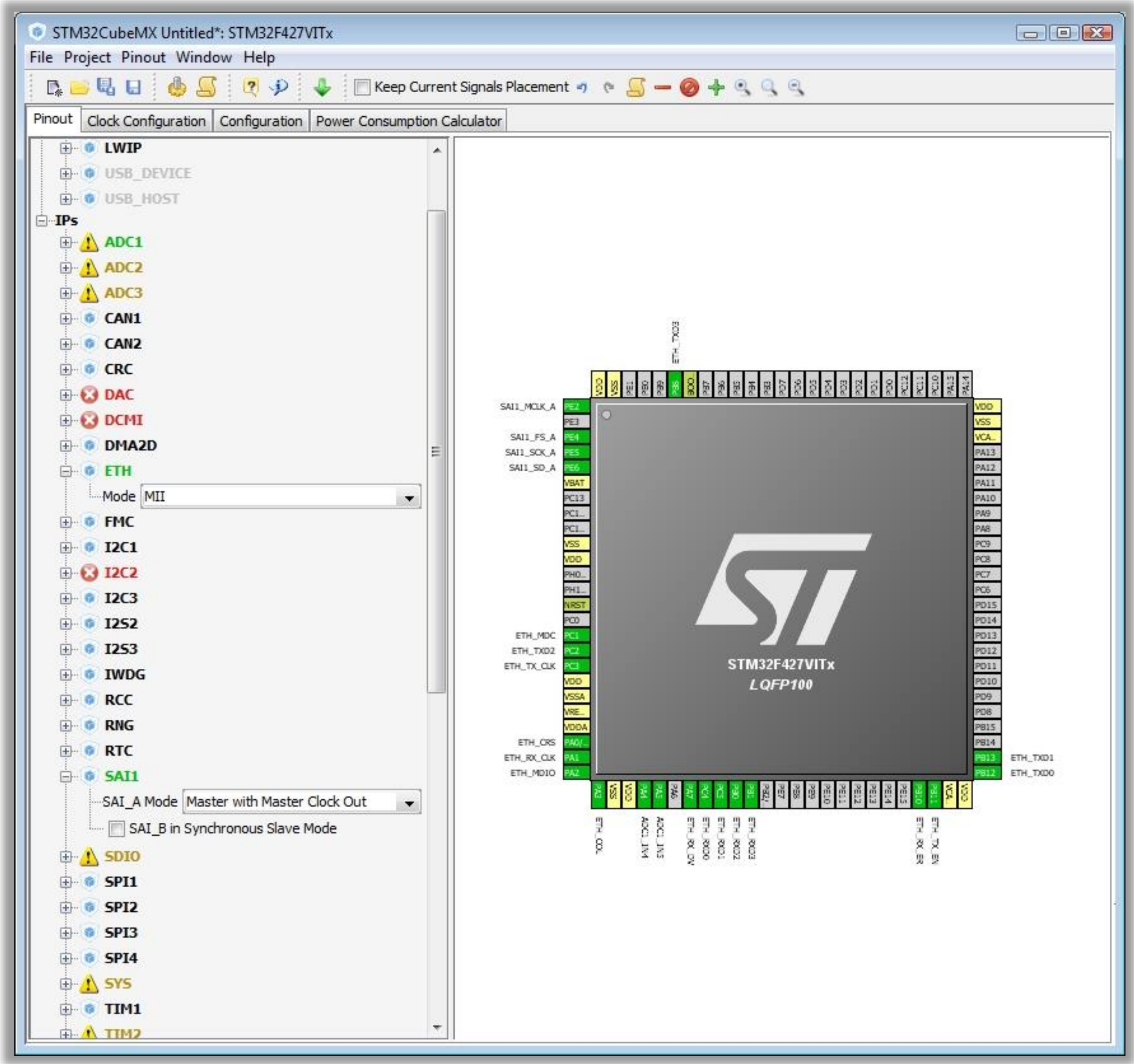


STM32CubeMX: Pinout configuration

- Pinout from:
  - Peripheral tree
  - Manually

- Automatic signal remapping

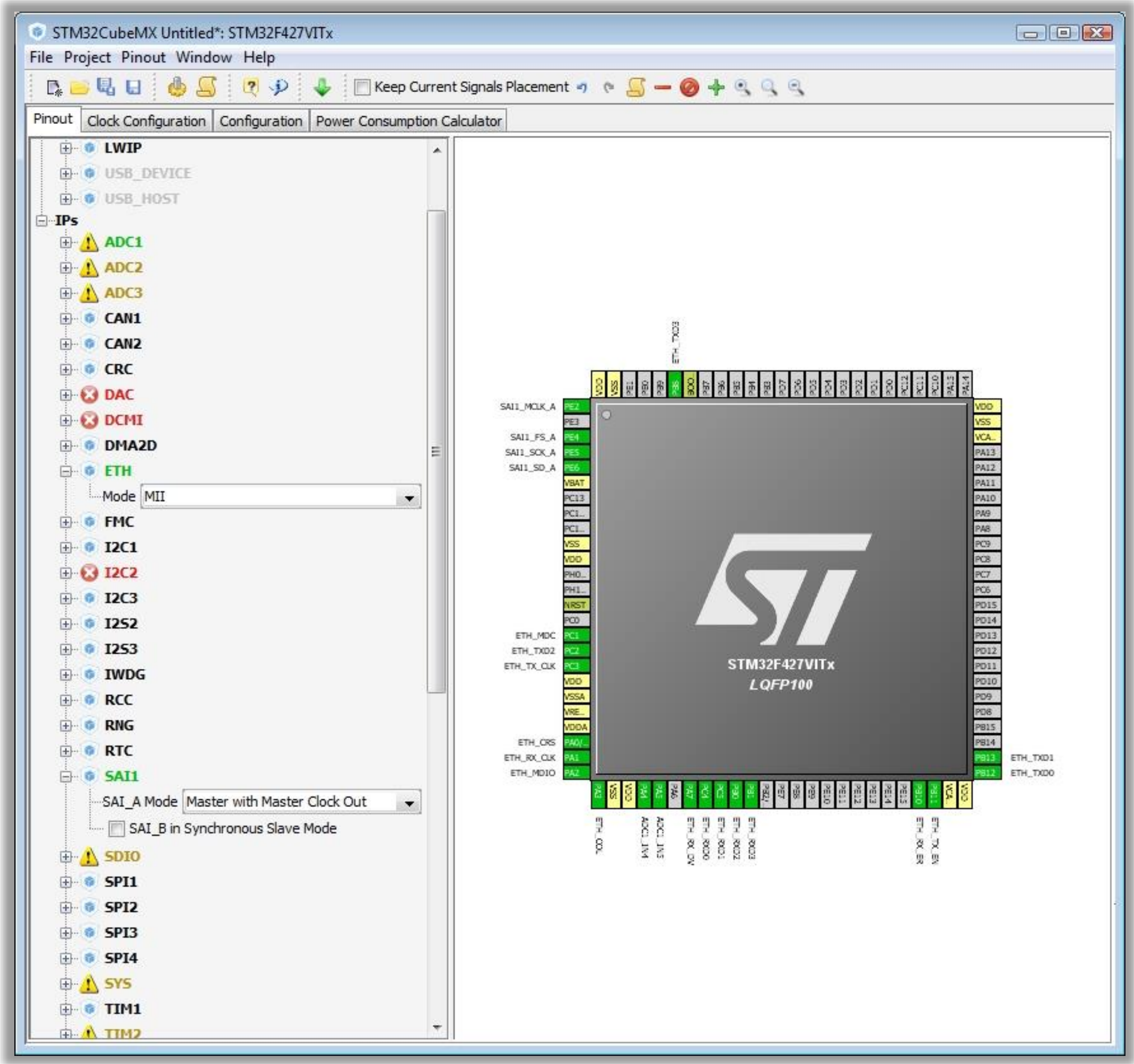
- Management of dependencies between peripherals and/or middleware (FatFS, LWIP, …)



life.augmented

STM32CubeMX: Pinout configuration

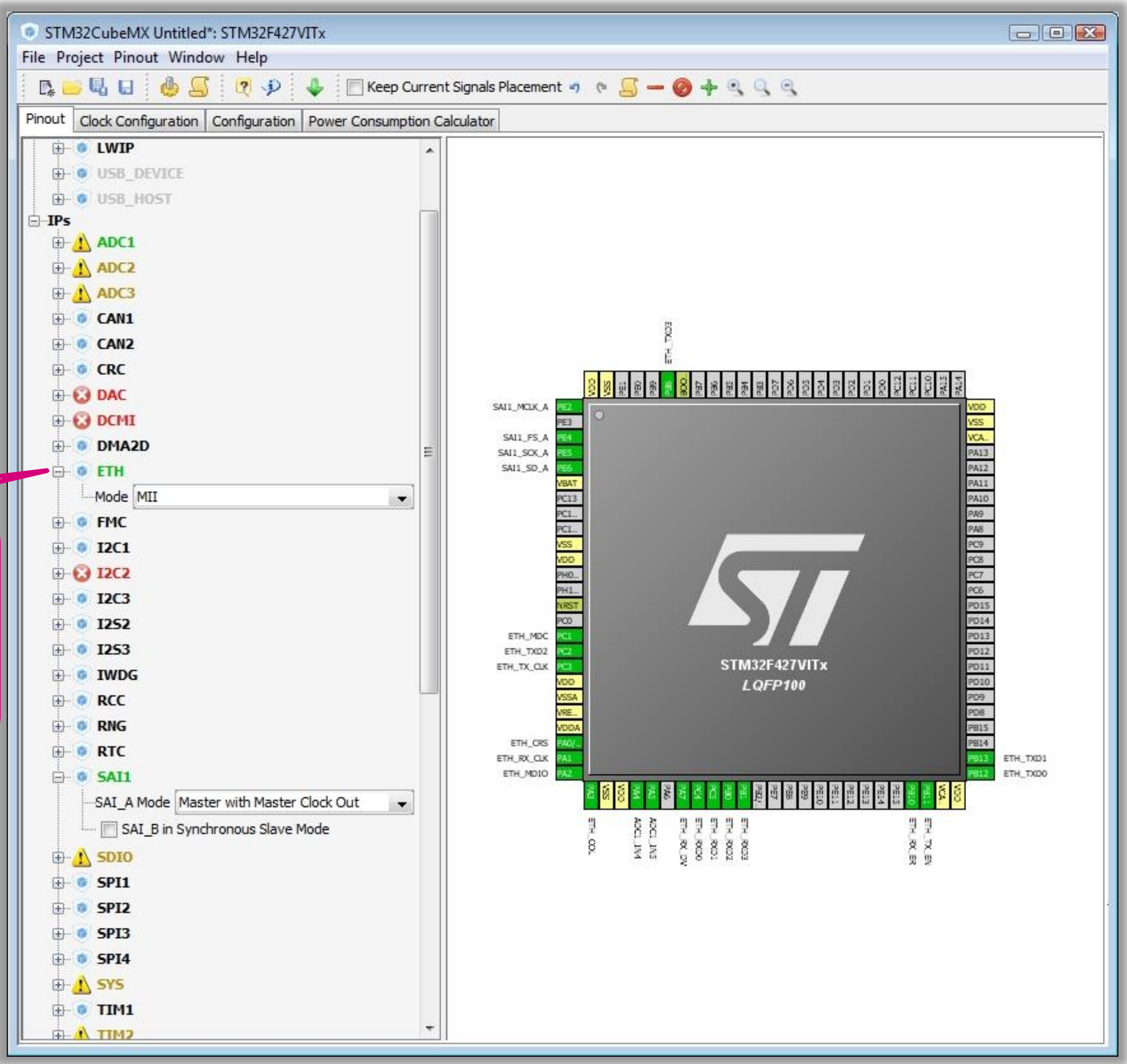
- Pinout from:
  - Peripheral tree
  - Manually
- Automatic signal remapping
- Management of dependencies between peripherals and/or middleware (FatFS, LWIP, …)
- **Motor Control Library is not part of STM32CubeMX Middleware, but is added (to .ioc file)externally by ST Motor Control Workbench**



STM32CubeMX: Pinout configuration

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

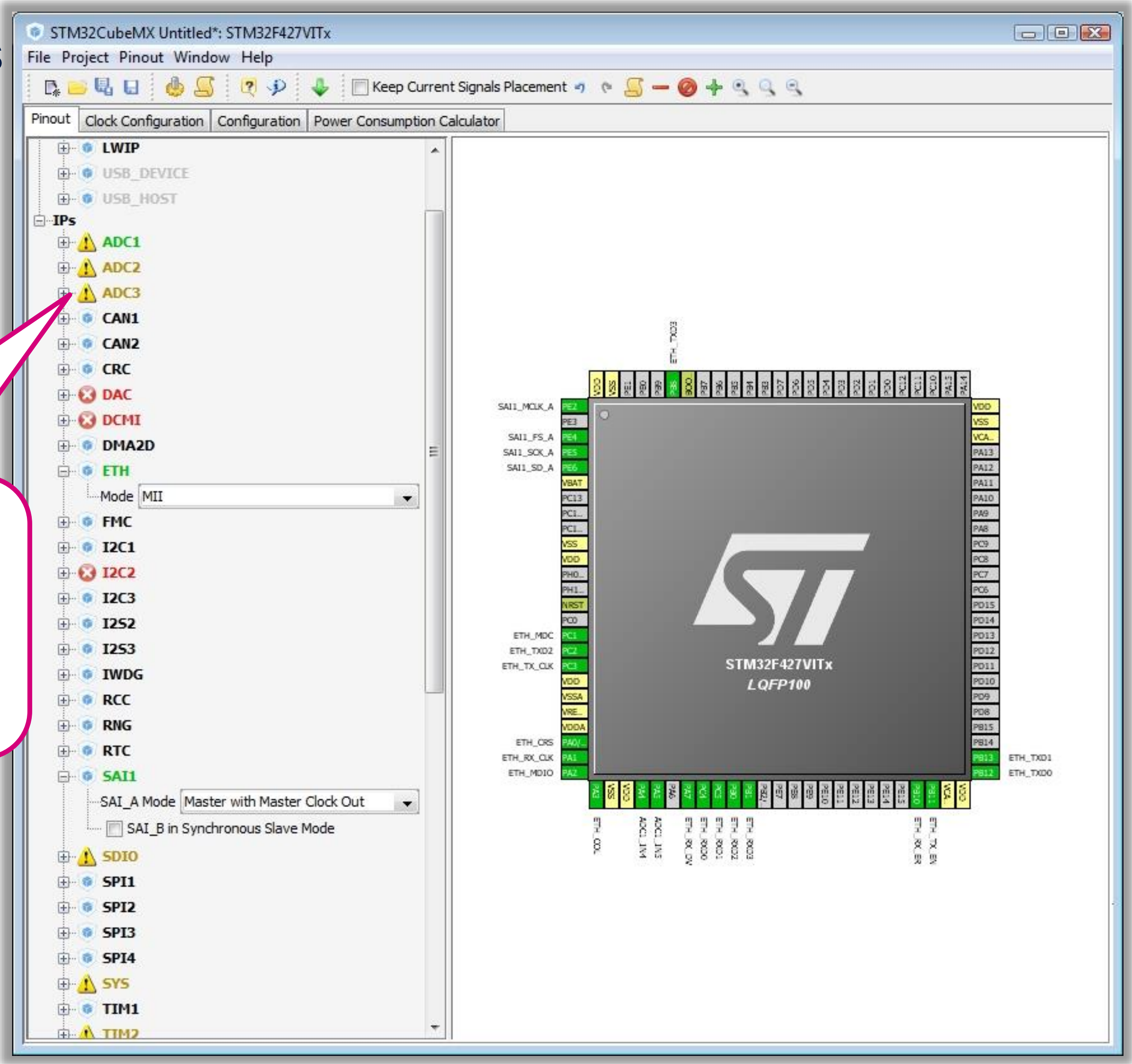
Green:
Periphery will be functional



STM32CubeMX: Pinout configuration

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

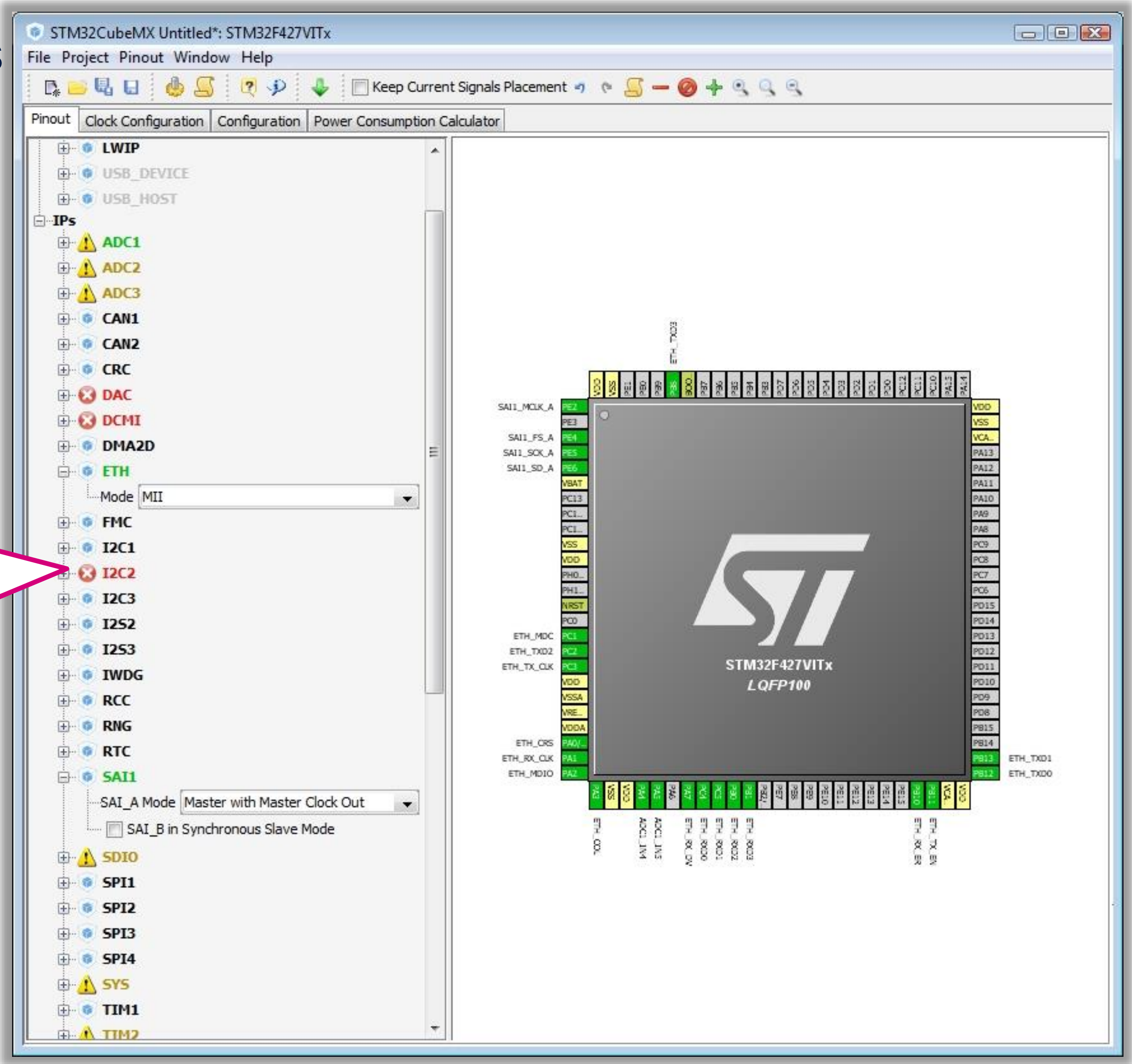
Yellow:
On ADC only some
channels can be used



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)

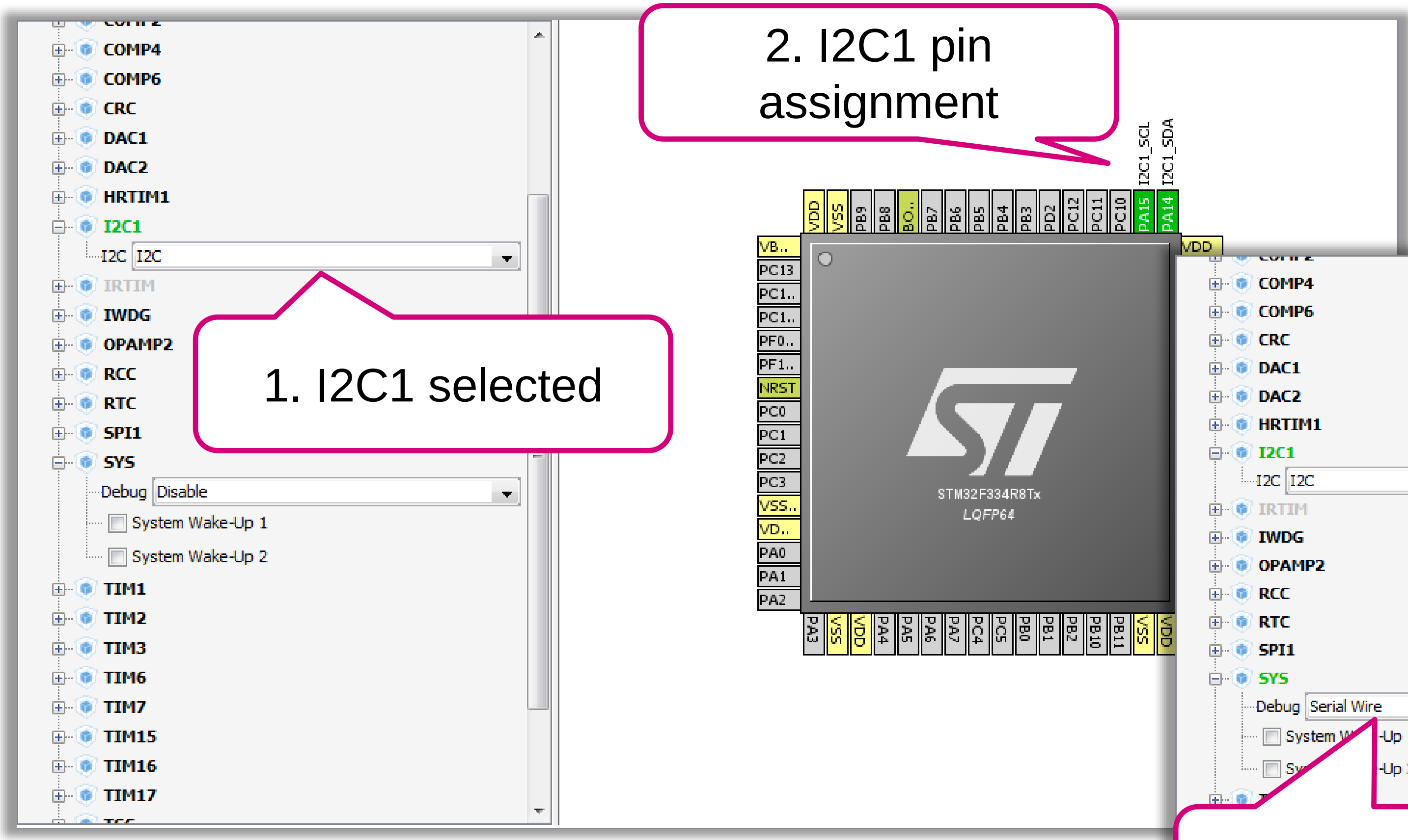
Red:
Periphery cannot be used in this pinout setup

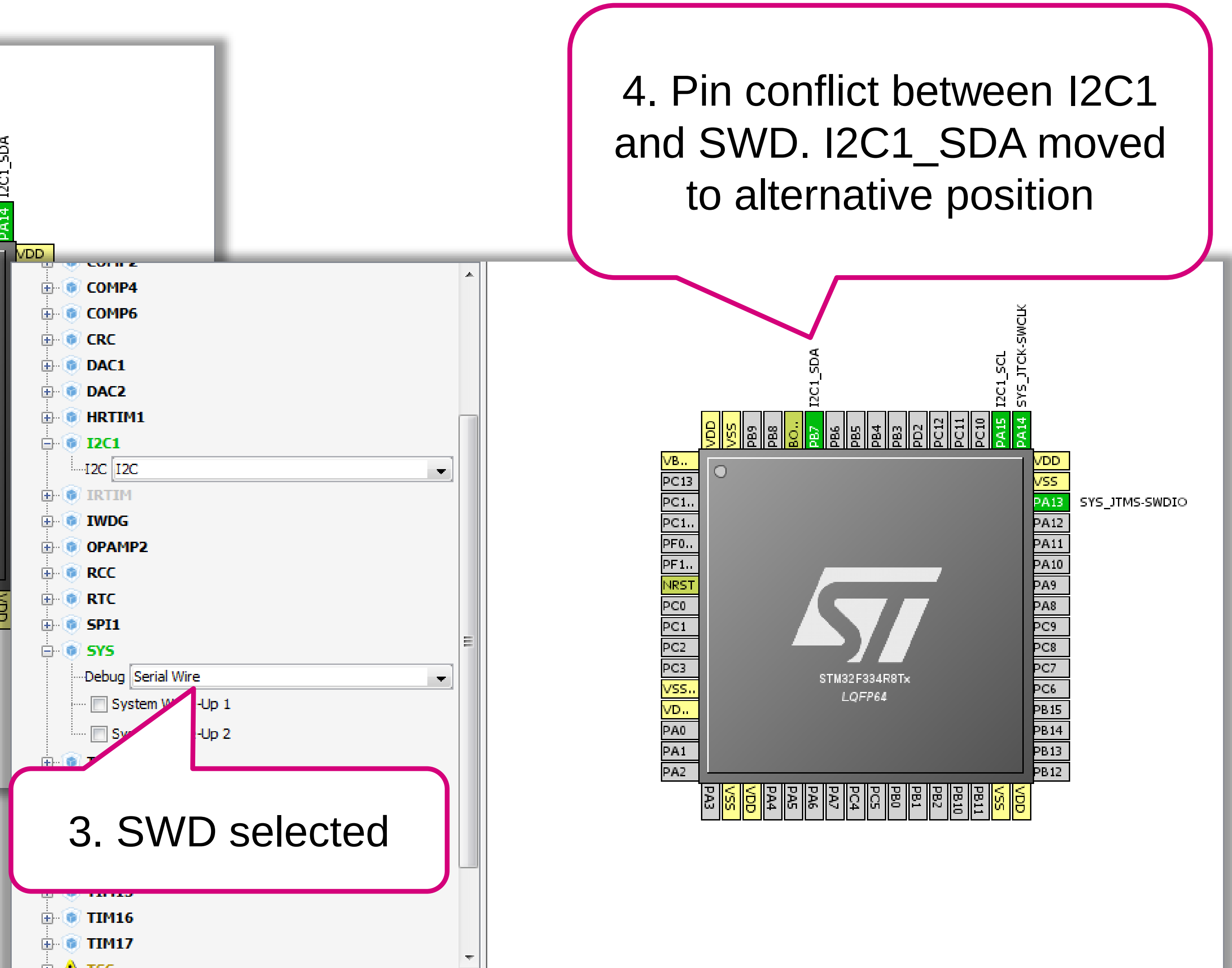


life.augmented

STM32CubeMX: Pinout configuration

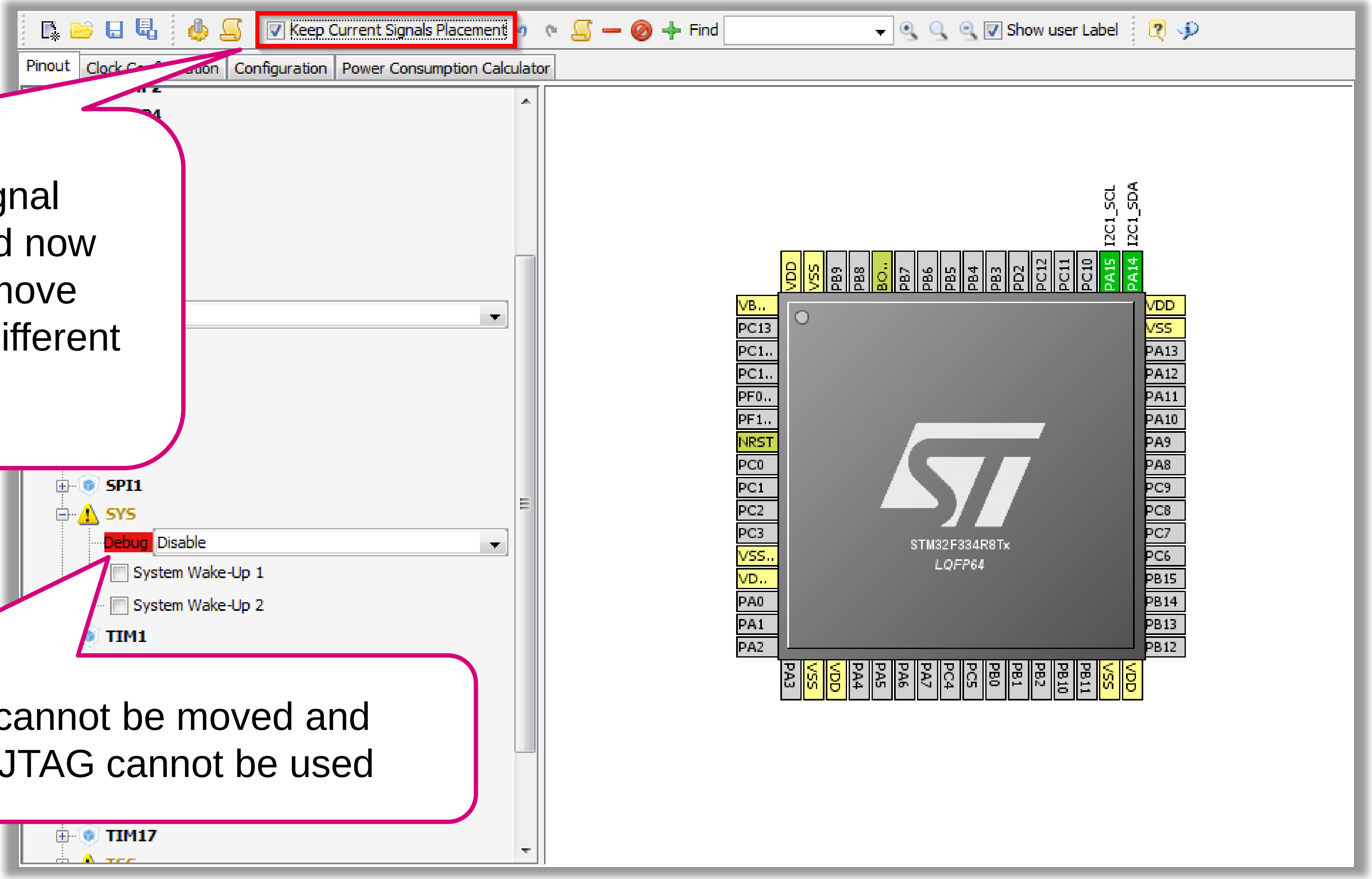
- Keep User Placement renamed to Keep Current Signal Placement and is unchecked by default





STM32CubeMX: Pinout configuration

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



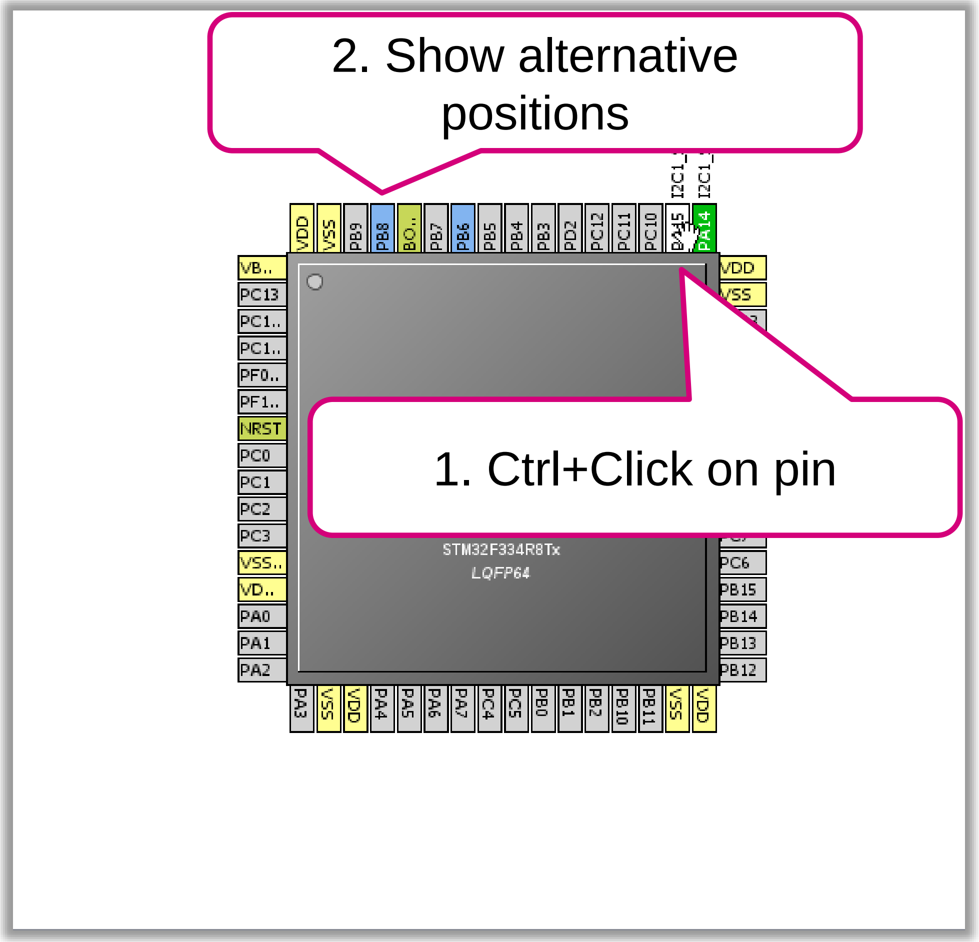
Keep Current Signal Placement checked now CubeMX cannot move selected signals to different alternate pin

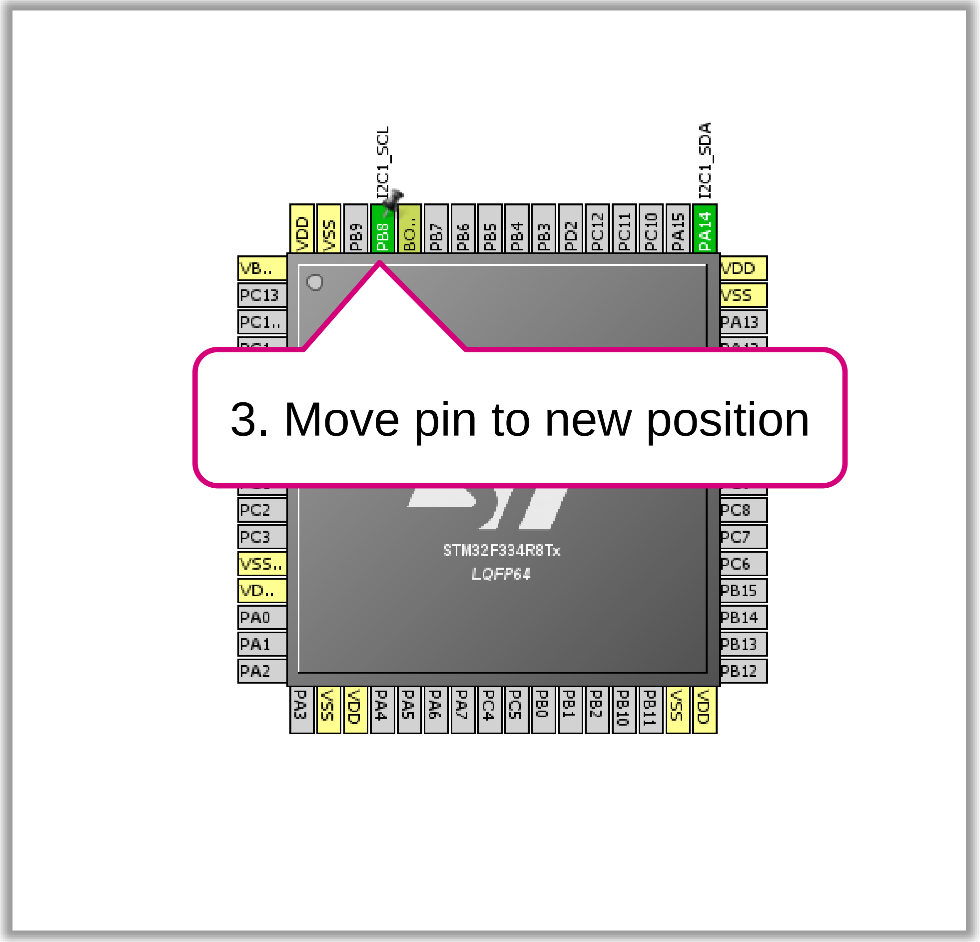
I2C1 cannot be moved and SWD/JTAG cannot be used

life.augmented

STM32CubeMX: Pinout configuration

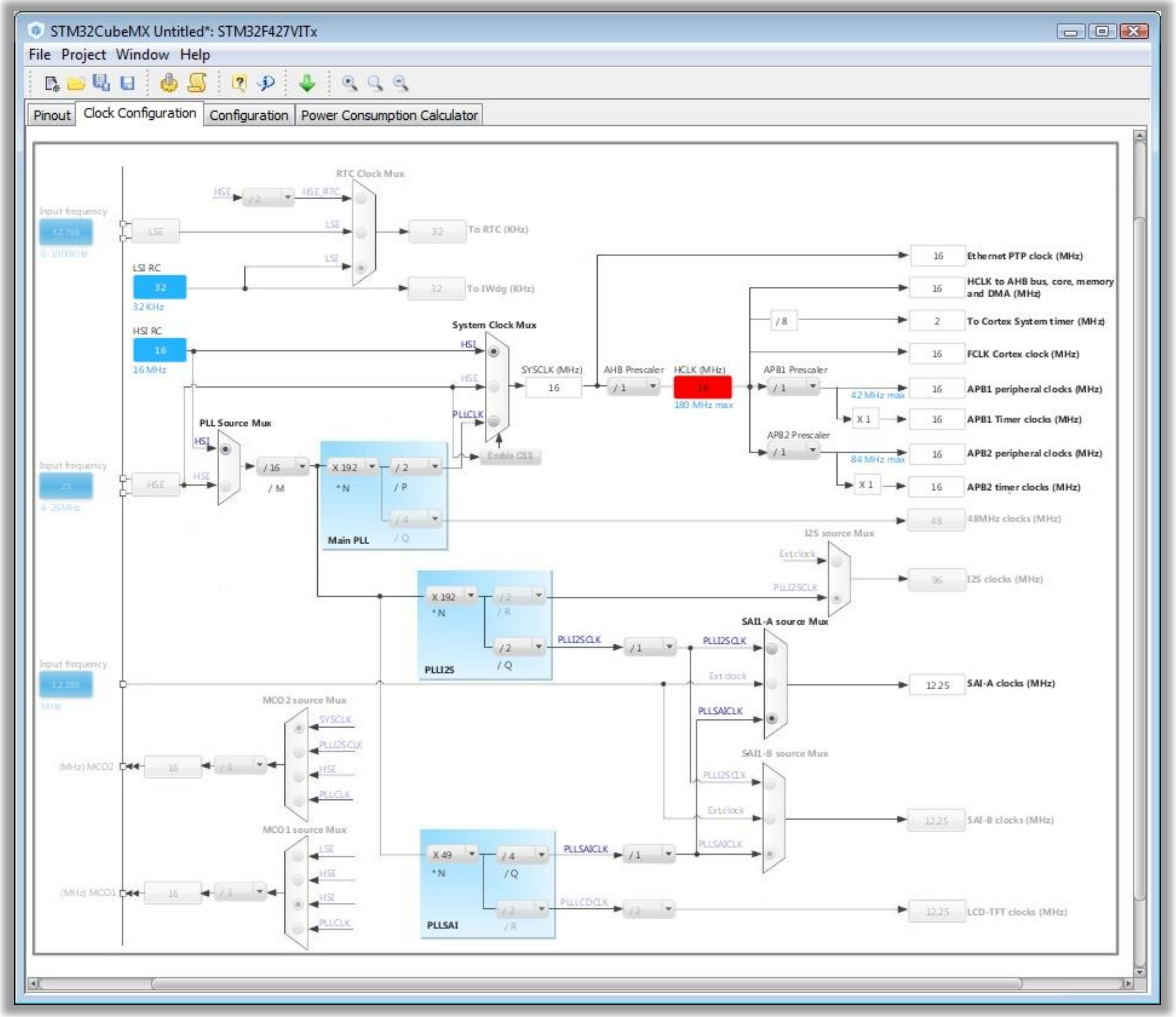
- Signals can be set/moved directly from the pinout view
  - To see alternate pins for a signal Ctrl+Click on the signal, you can then drag and drop the signal to the new pin (keep pressing the Ctrl key)





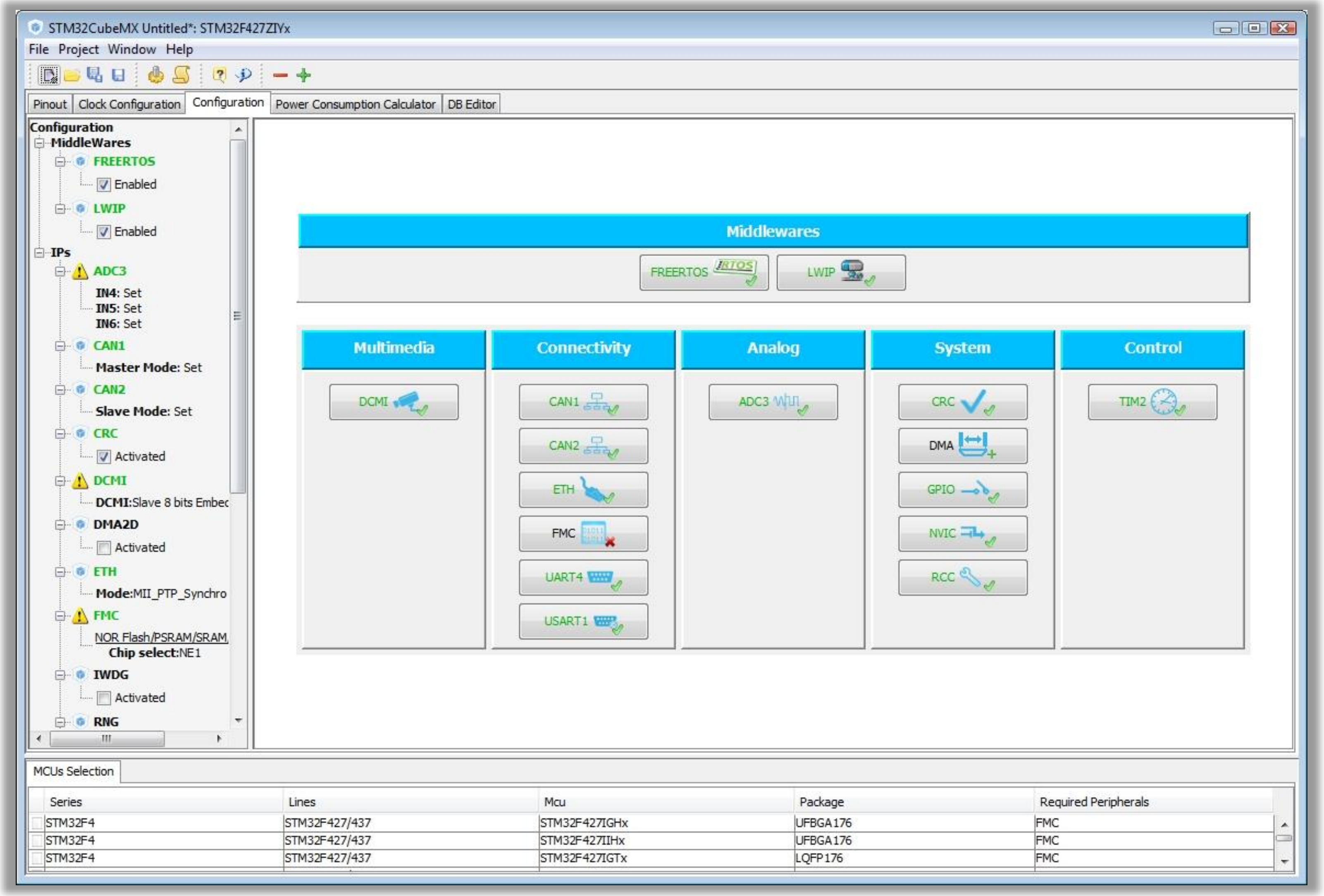
STM32CubeMX: Clock tree

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



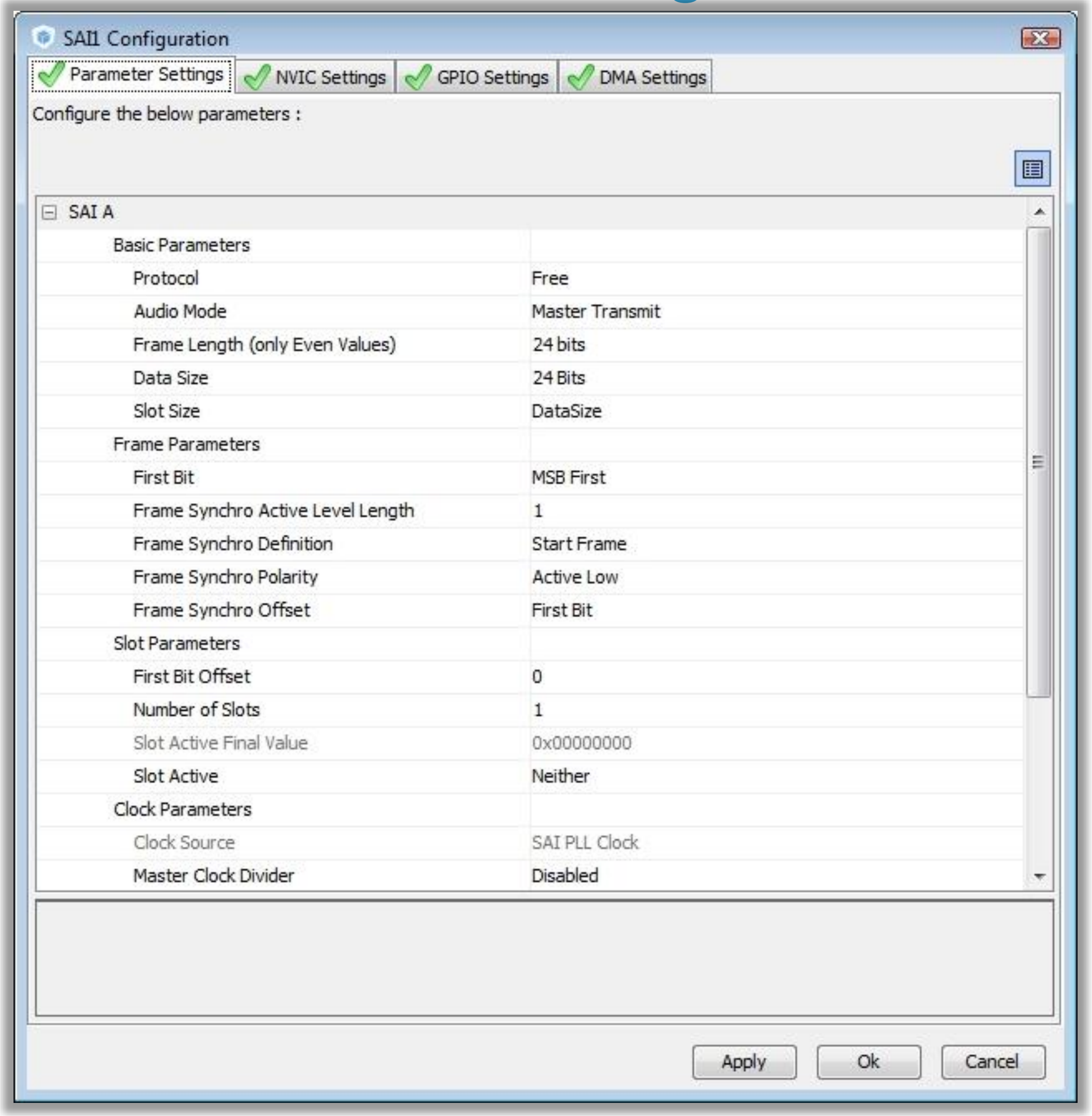
STM32CubeMX: Peripheral and middleware configuration

- 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



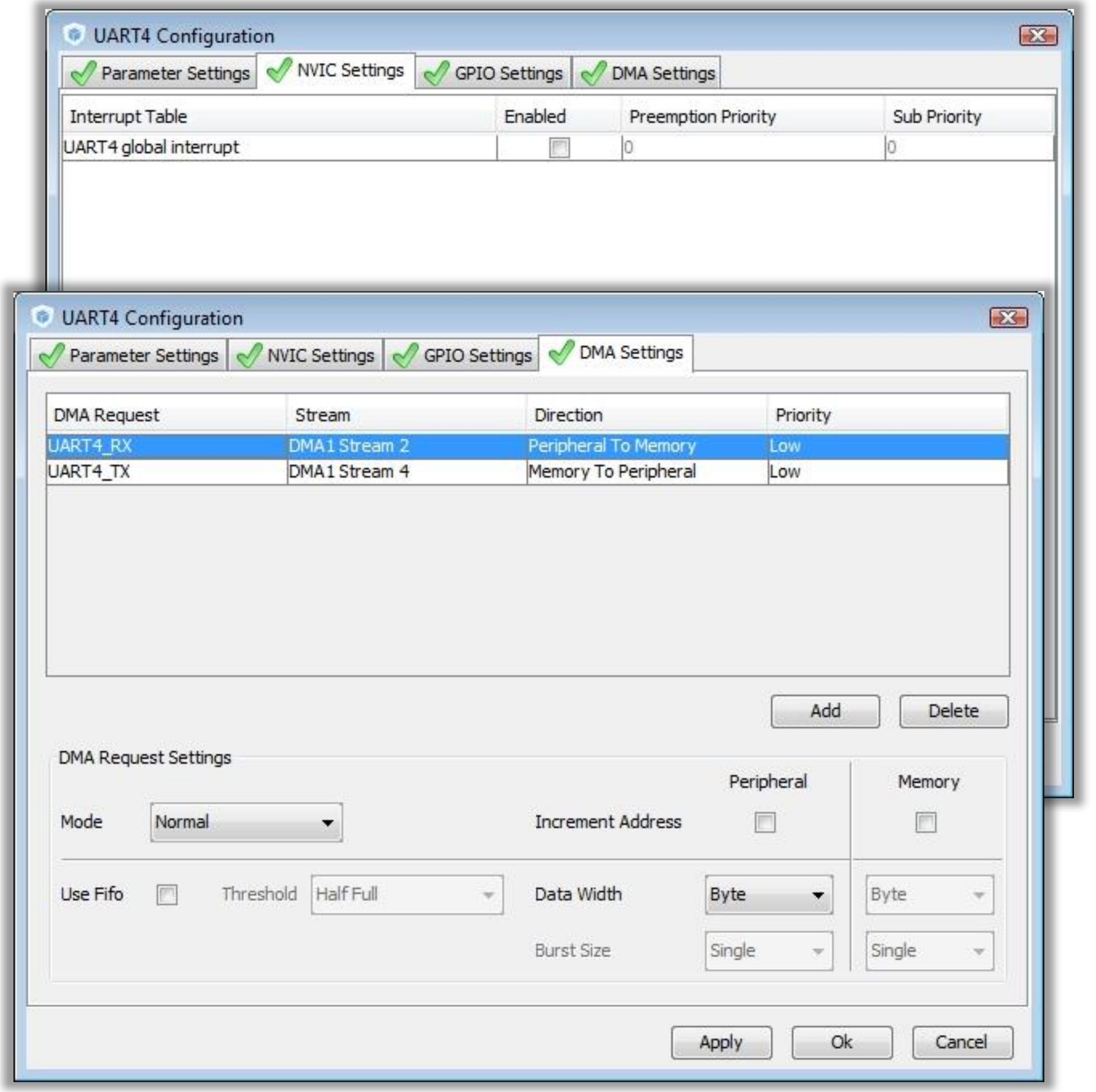
STM32CubeMX: Peripheral and middleware configuration

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



STM32CubeMX: Peripheral and middleware configuration

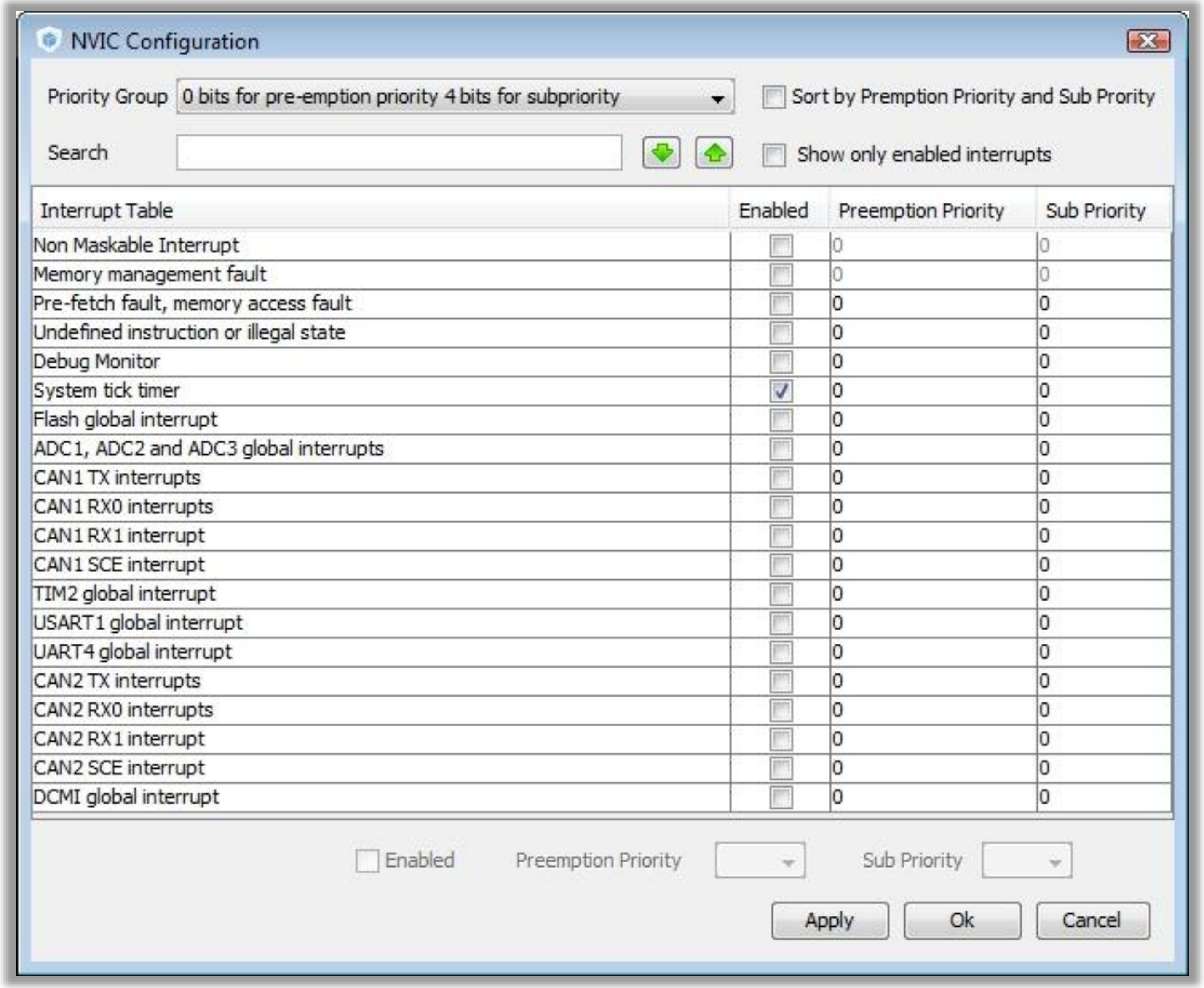
- Manage Interruptions
  - priorities can only be set in the NVIC global view
- Manage GPIO parameters
- Manage DMA
  - Configure all the parameters of the DMA request
  - Runtime parameters (start address, …) are not managed



life.augmented

NVIC Panel

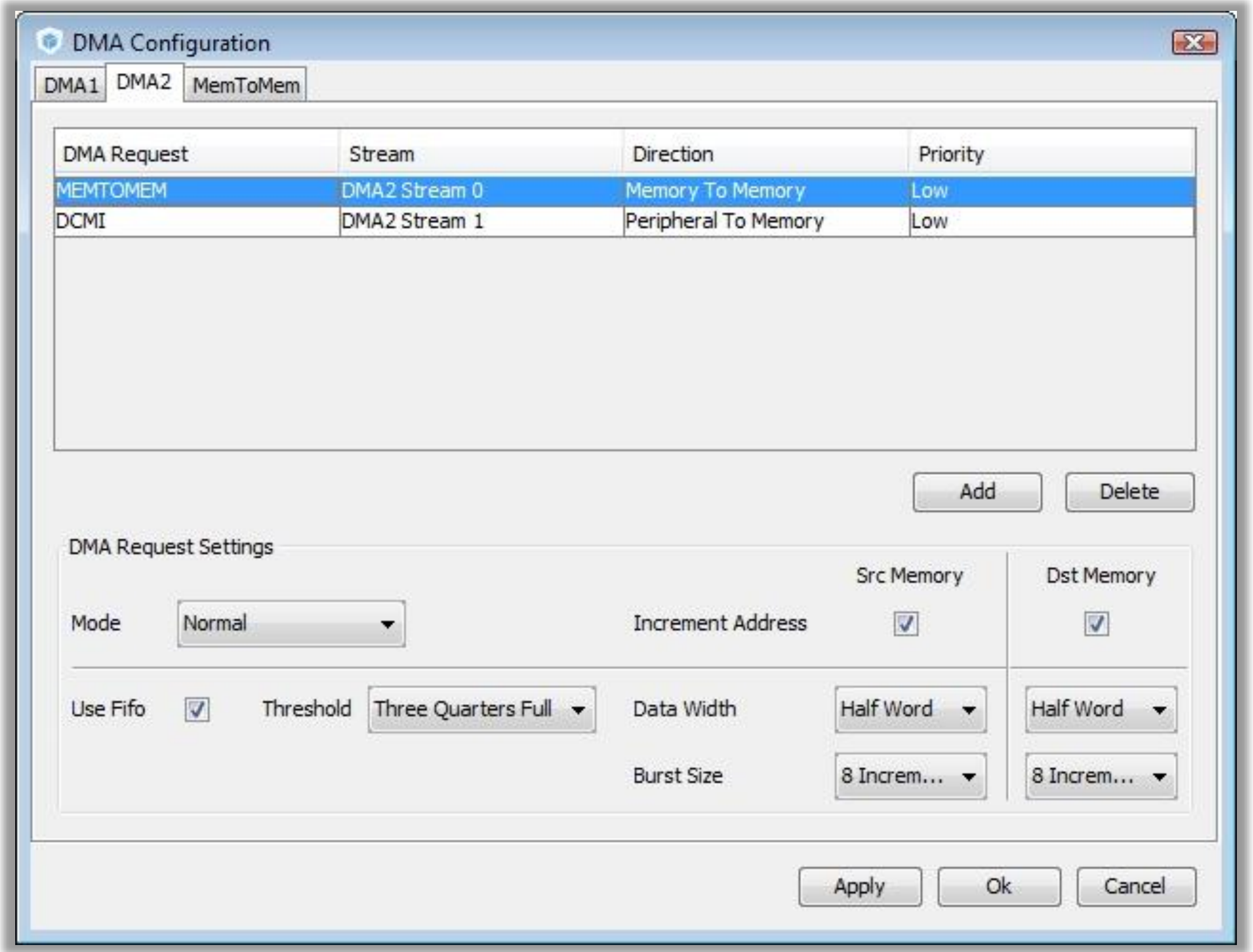
- Manage all interruptions
- Manage priorities and sort by priorities
- Search for a specific interrupt in the list



life.augmented

DMA Panel

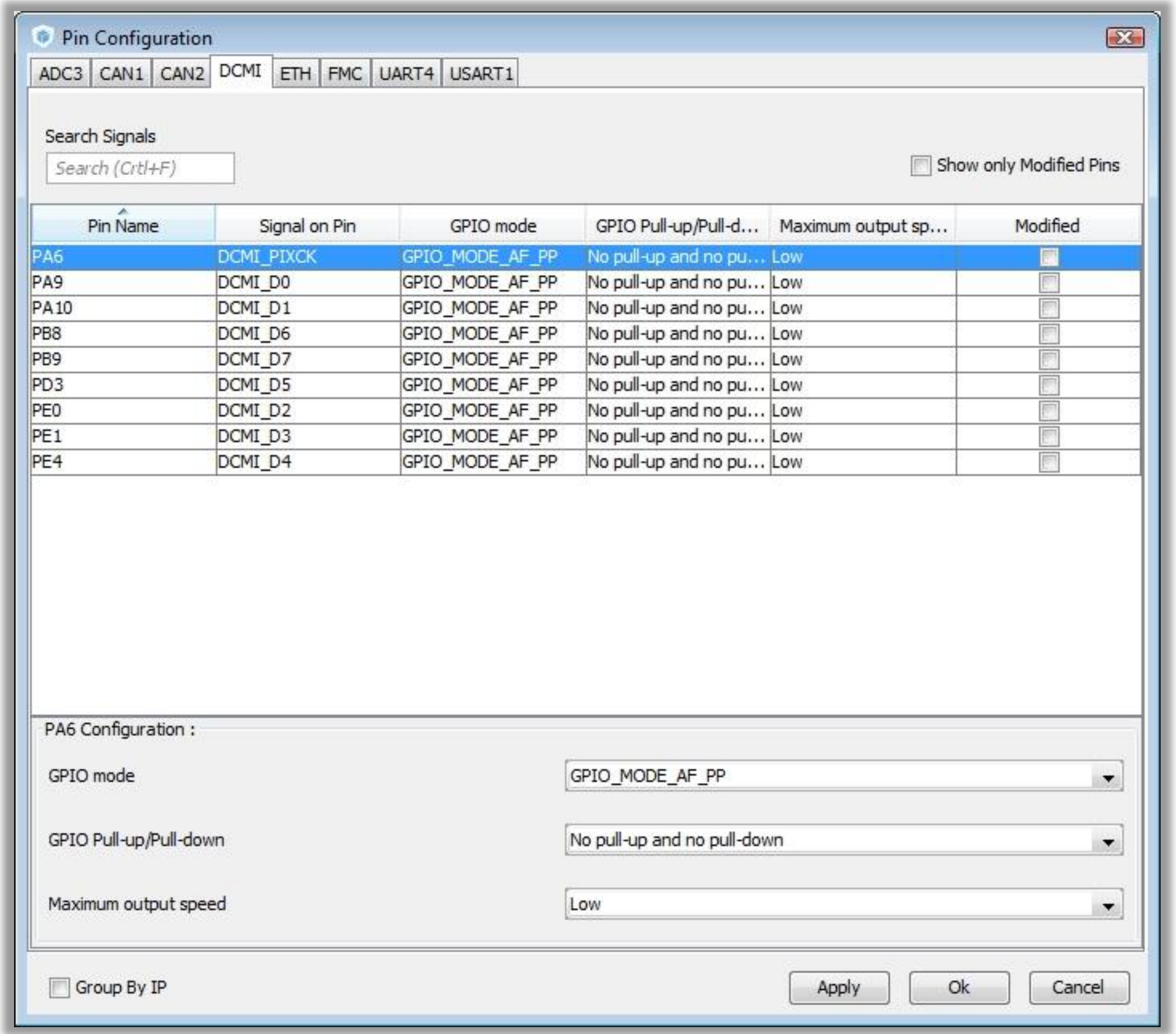
- Manage All DMA requests including Memory to Memory
- Set Direction and priority
- Set specific parameters



life.augmented

GPIO Panel

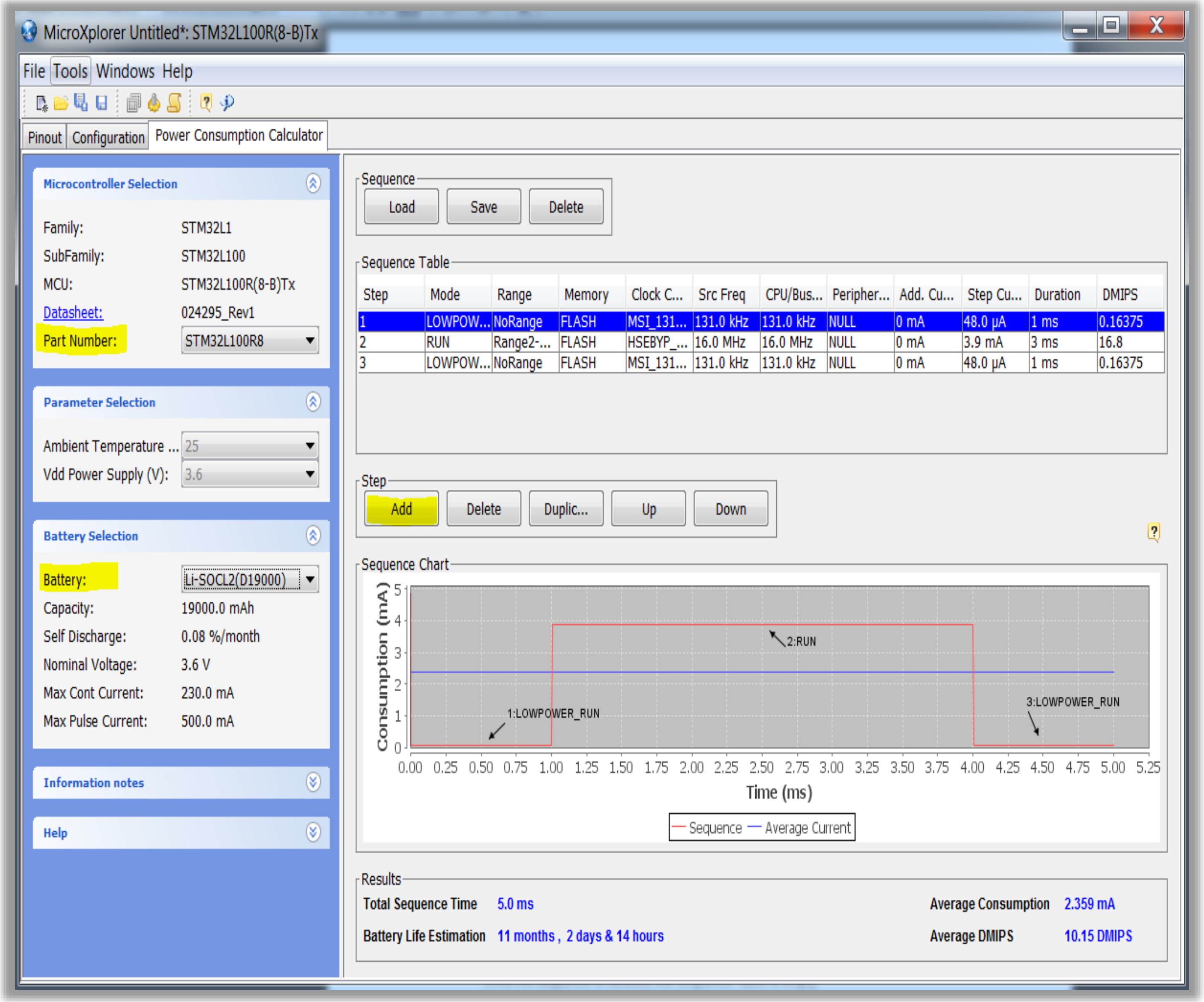
- Most of the GPIO parameters are set by default to the correct value
- You may want to change the maximum output speed
- You can select multiple pin at a time to set the same parameter



life.augmented

STM32CubeMX: Power consumption calculator

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



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

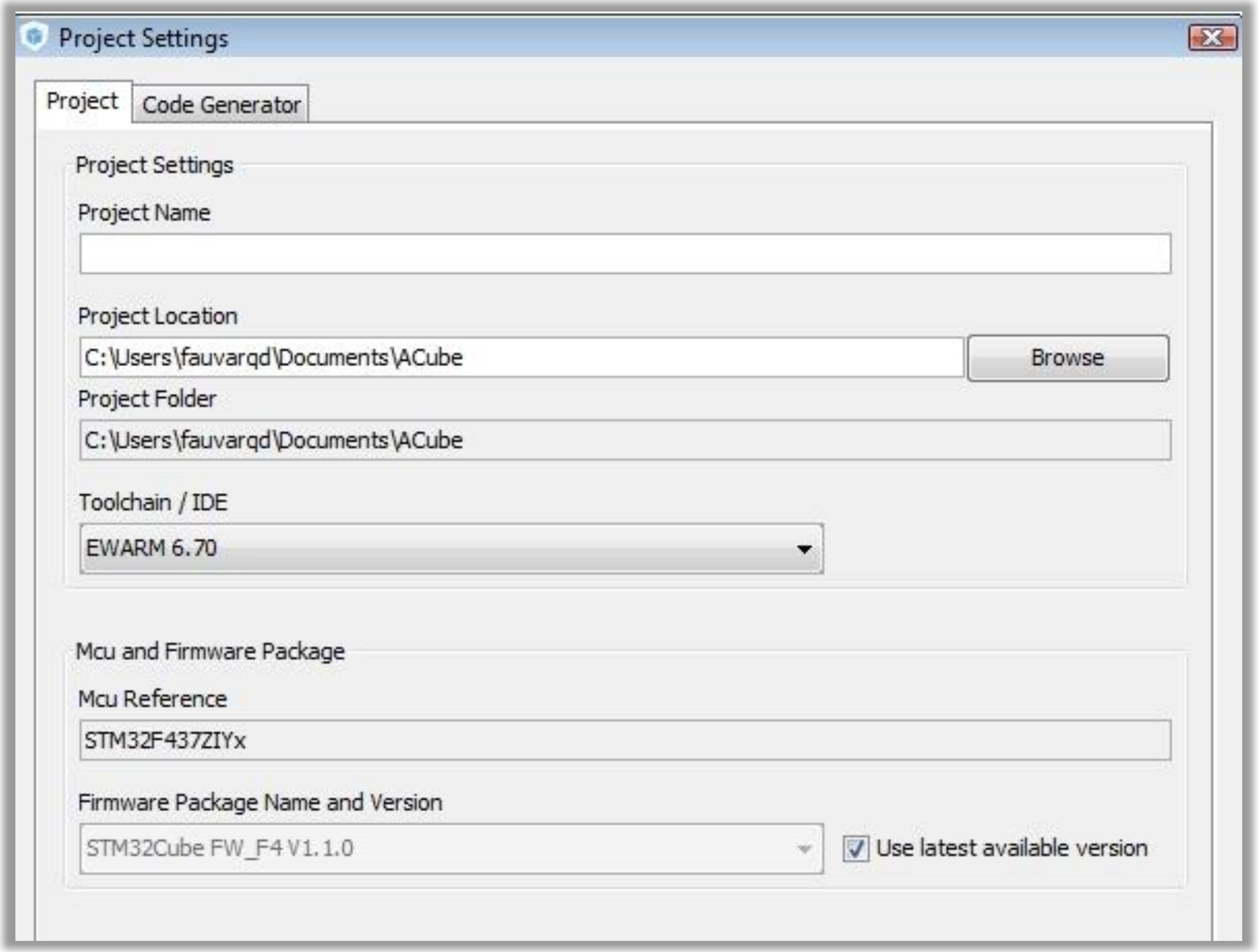
- Help->Updater settings
  - Choose location of STM32CubeFx firmware libraries repository
  - Choose manual or automatic check
  - Set Connection proxy
    - Inside ST use appgw.sgp.st.com port 8080 with your windows login name and password

- 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

Can be happen, that STM32CubeMX window in ST MC Workbench get you errors during generation(but generated code is proper). You change in Updater Settings in Check and Update Settings to **Manual Check** and Data Auto-Refresh to **No Auto-Refresh** at application start

STM32CubeMX: Project settings

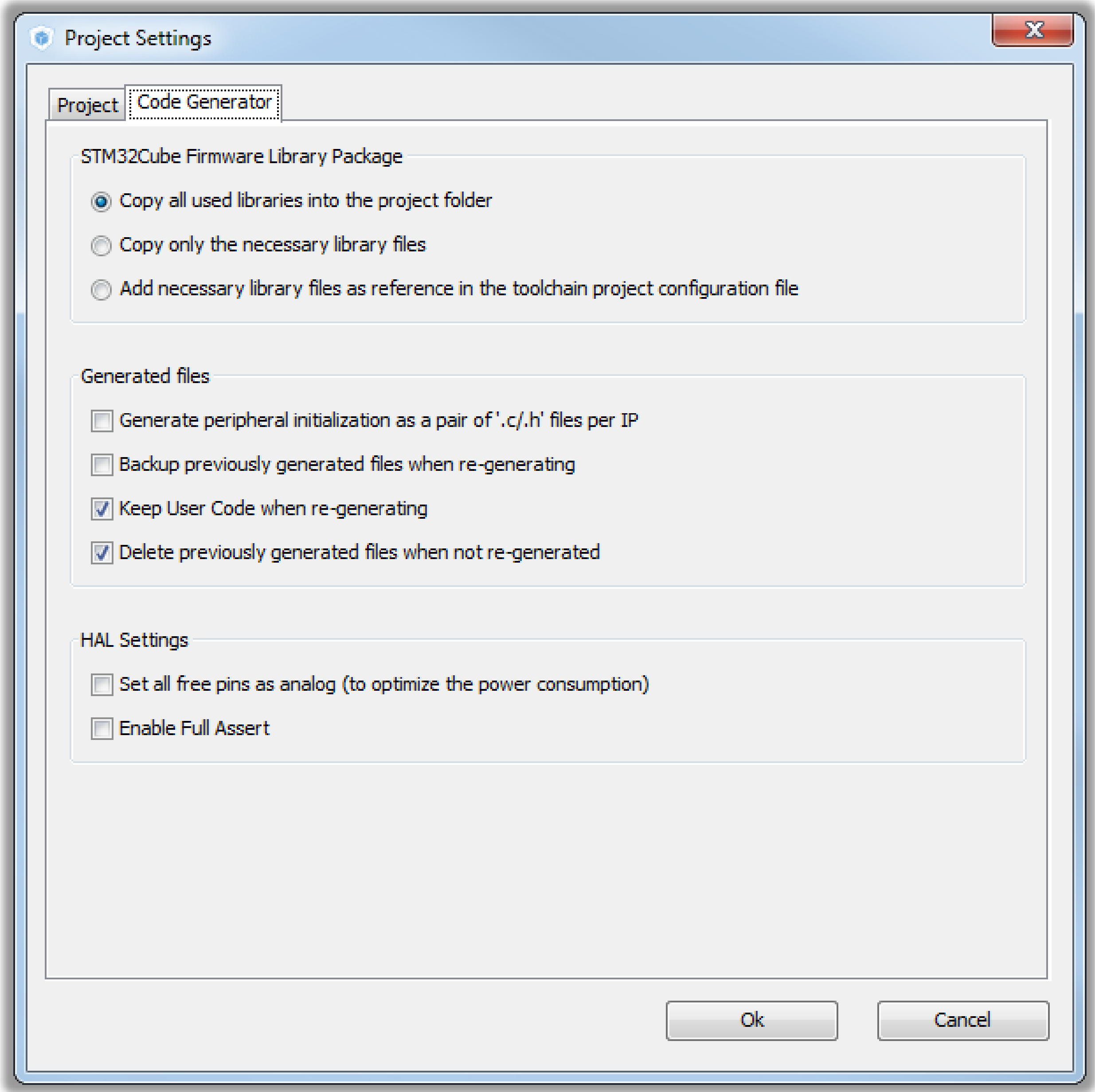
- Project -> Settings
  - Set project name and location
  - A full folder will be created named with project name.
  - Inside this folder you’ll find the saved configuration and all the generated code
  - Select toolchain
    (Keil, IAR, Atollic, SW4STM32)
  - You can choose to use the latest version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



life.augmented

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 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



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

Generated files

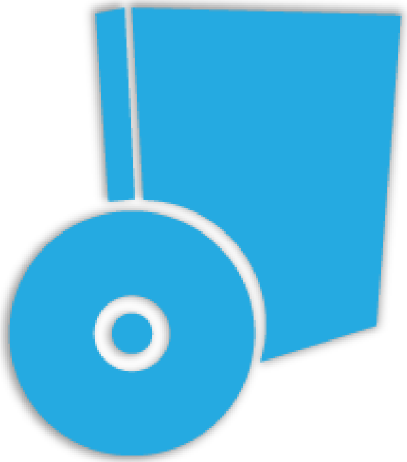
- [ ] Generate peripheral initialization as a pair of '.c/.h' files per IP
- [ ] Backup previously generated files when re-generating
- [x] Keep User Code when re-generating
- [x] Delete previously generated files when not re-generated

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
  - Stm32f4xx_it.c
  - Stm32f4xx_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 */
```

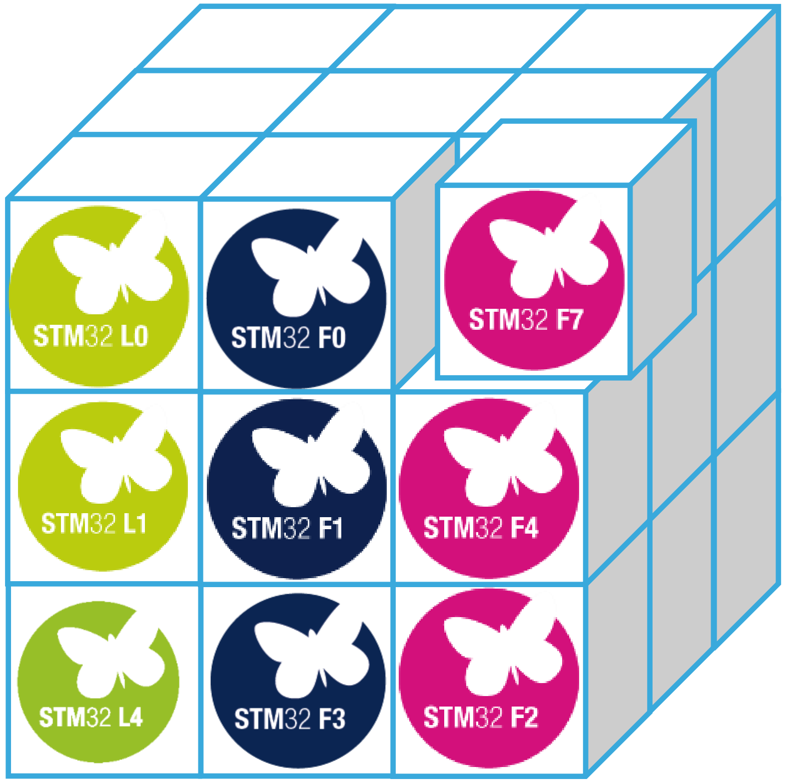
STM32Cube HAL package



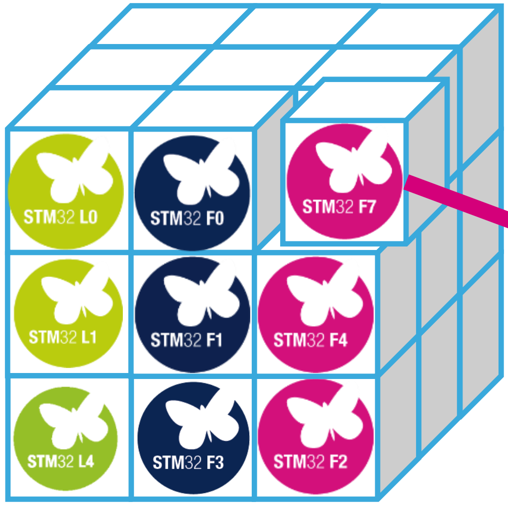
STM32Cube™ V1 Introduction

- STM32Cube™ Version 1:
  - A configuration tool, STM32CubeMX generating initialization code from user choices
  - A full embedded software offer, delivered per series (like STM32CubeF3) with:
    - An STM32 Abstraction Layer embedded software: STM32Cube HAL
    - A consistent set of Middlewares: Motor Control,RTOS, USB, TCP/IP, Graphics, …
  - Available at st.com as free solution

STM32CubeMX



STM32Cube FW Package Organization



Can be found in CubeMX repository

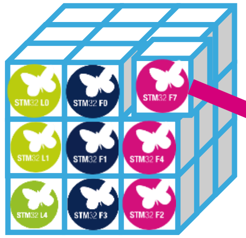
Cube XX HAL package

- Documents
- Drivers
- Middleware
- Projects
- Utilities

Getting started with CubeF3 document

Supporting files like fonts, or pictures for graphic examples, …

STM32Cube FW Package Drivers



Cube F4 HAL package

- Documents
- Drivers
- Middleware
- Projects
- Utilities

DRIVE

Drivers

CMSIS

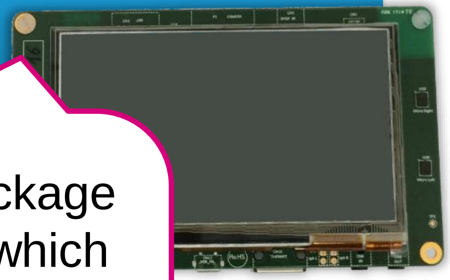
CMSIS COMPLIANT
ARM® Cortex™ Microcontroller Software Interface Standard

Register definitions for Core, startup files, ARM cortex libraries

STM32Fxxx_HAL_Driver

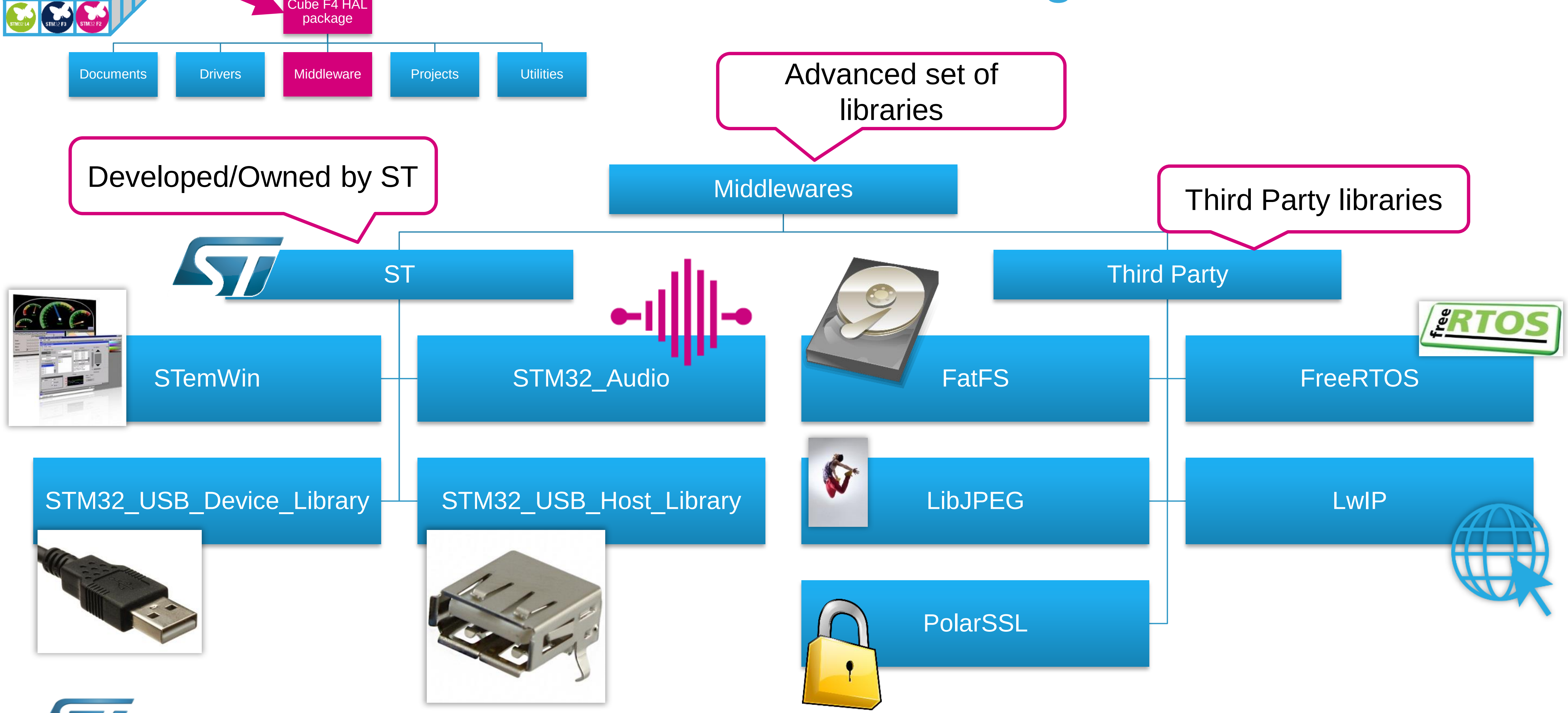
HAL Drivers for each periphery in STM32

BSP



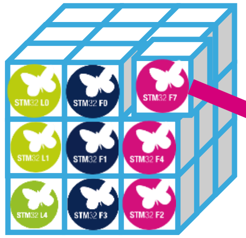
Board Support Package contains function which use HAL drivers to communicate with other components present n EVAL/Discovery boards

STM32Cube FW Package Middlewares



life.augmented

STM32Cube FW Package Projects



- Cube F4 HAL package
  - Documents
  - Drivers
  - Middleware
  - Projects
  - Utilities

Complete projects for STM32 boards

- Projects
  - STM32F-Discovery
    - Templates
    - Examples
    - Applications
    - Demonstrations
  - ….

STM32Nucleo, Discovery kits, Eval Boards

Empty project only with startup file prepared for modification

Simple examples for STM32 Peripheries (GPIO, USART, …)

Advanced examples which use Middlewares (USB virtual com port)

Demonstration project combine multiple Middlewares together

STM32Cube FW Package Organization

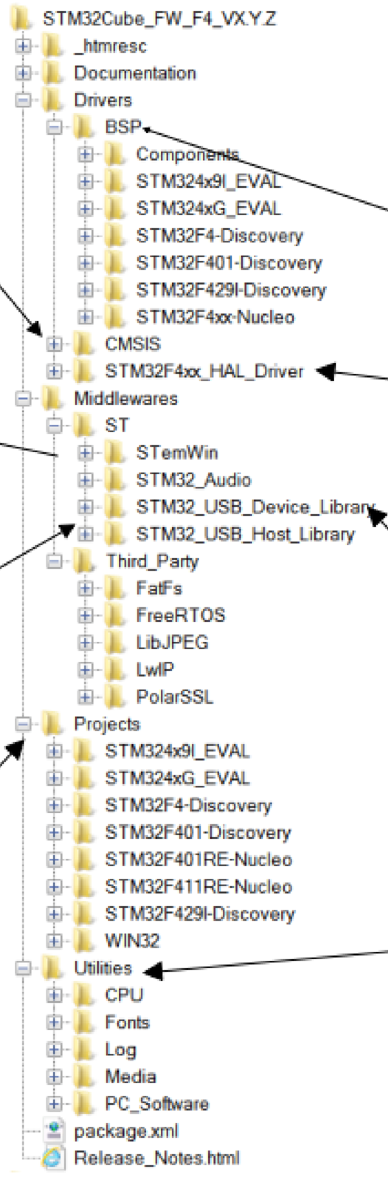
Contains STM32F4xx CMSIS files that defines Peripheral's registers declarations, bits definition and the address mapping

STemWin professional stack coming from SEGGER and available in binary form

USB Host Library supporting both OTG FS and HS cores and offering the following classes: HID, MSC, CDC, Audio, and MTP

Set of examples organized by board and provided with preconfigured projects.

WIN32 directory contains Windows-based simulation project for STemWin.



BSP drivers for the supported boards

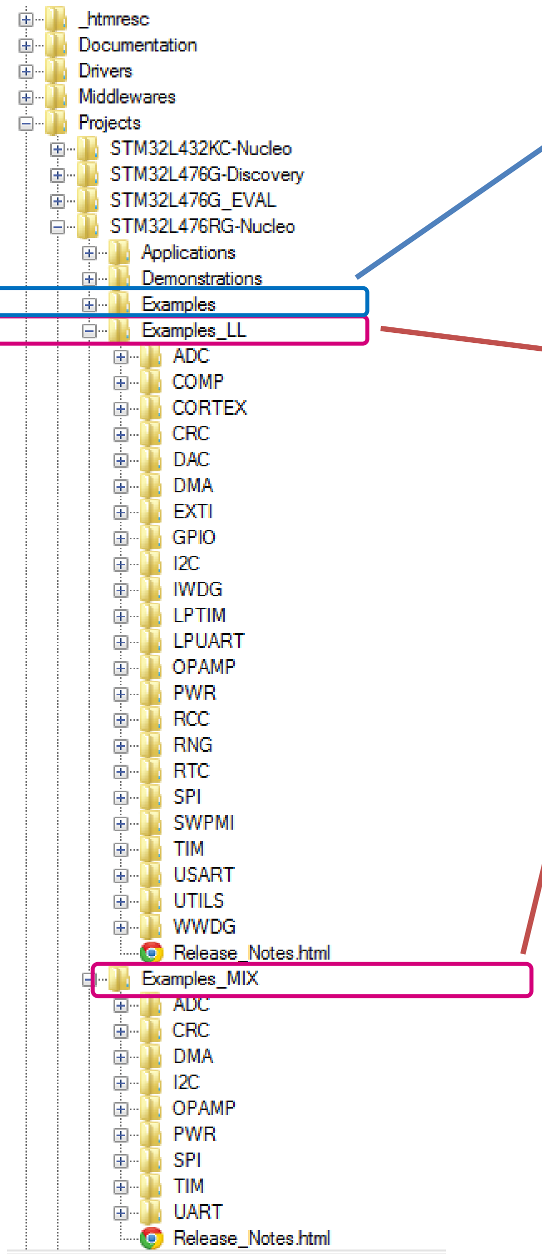
STM32F4xx HAL drivers for all modules

USB Device Library supporting both OTG FS and HS cores and offering the following classes: HID, MSC, CDC, Audio, and DFU

Miscellaneous utilities

life.augmented

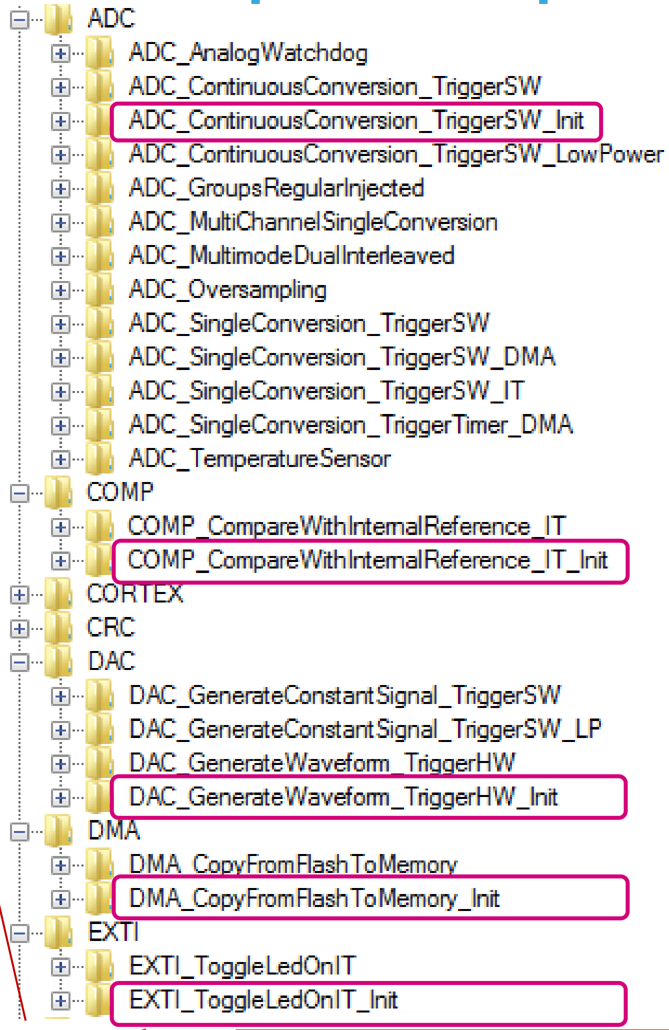
Examples repository



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

NEW Examples that are based ONLY on LL drivers

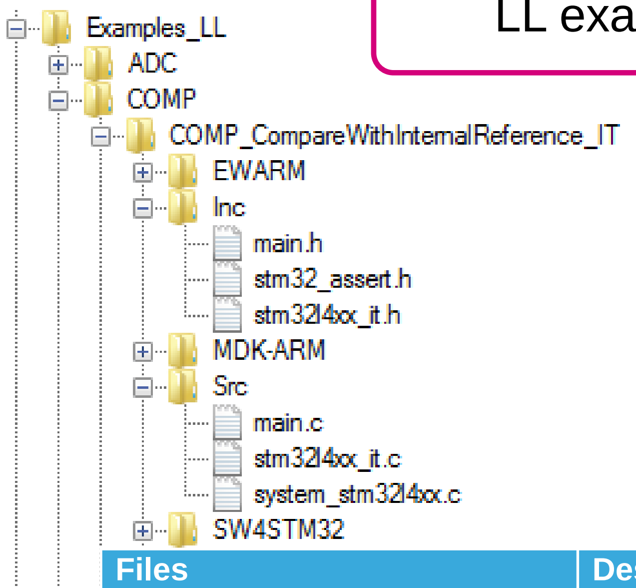
NEW Examples that are based on HAL and LL drivers (Mixed)



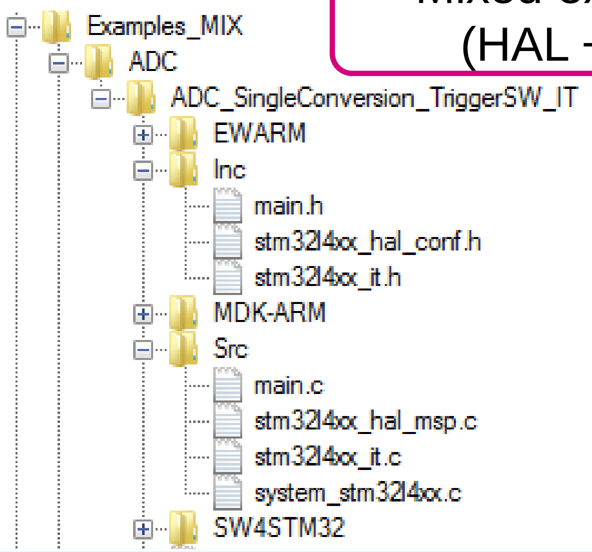
Each IP which offer Initialization functions, provide **ONE** example to demonstrate his usage (USE_FULL_LL_DRIVER activated)

Examples files

LL example



Mixed example
(HAL + LL)



Files	Description
main.c	This file should include the example program It should include only the main.h file
main.h	Header of the main program file: it should include the associated low layer drivers header files and example defines and data structures when needed.
system_stm32yyxx.c	CMSIS Cortex-Mx Device Peripheral Access Layer System Source File.
stm32yyxx_it.c	This file contains main Interrupt Service Routines.
stm32_assert.h	This file contains the assert_param macro implementation, at least needed by stm32yyxx_utils.c which provides utility services.
stm32yyxx_it.h	This file contains the headers of the interrupt handlers.
Readme.txt	This file contains Example usage explanations for end user.

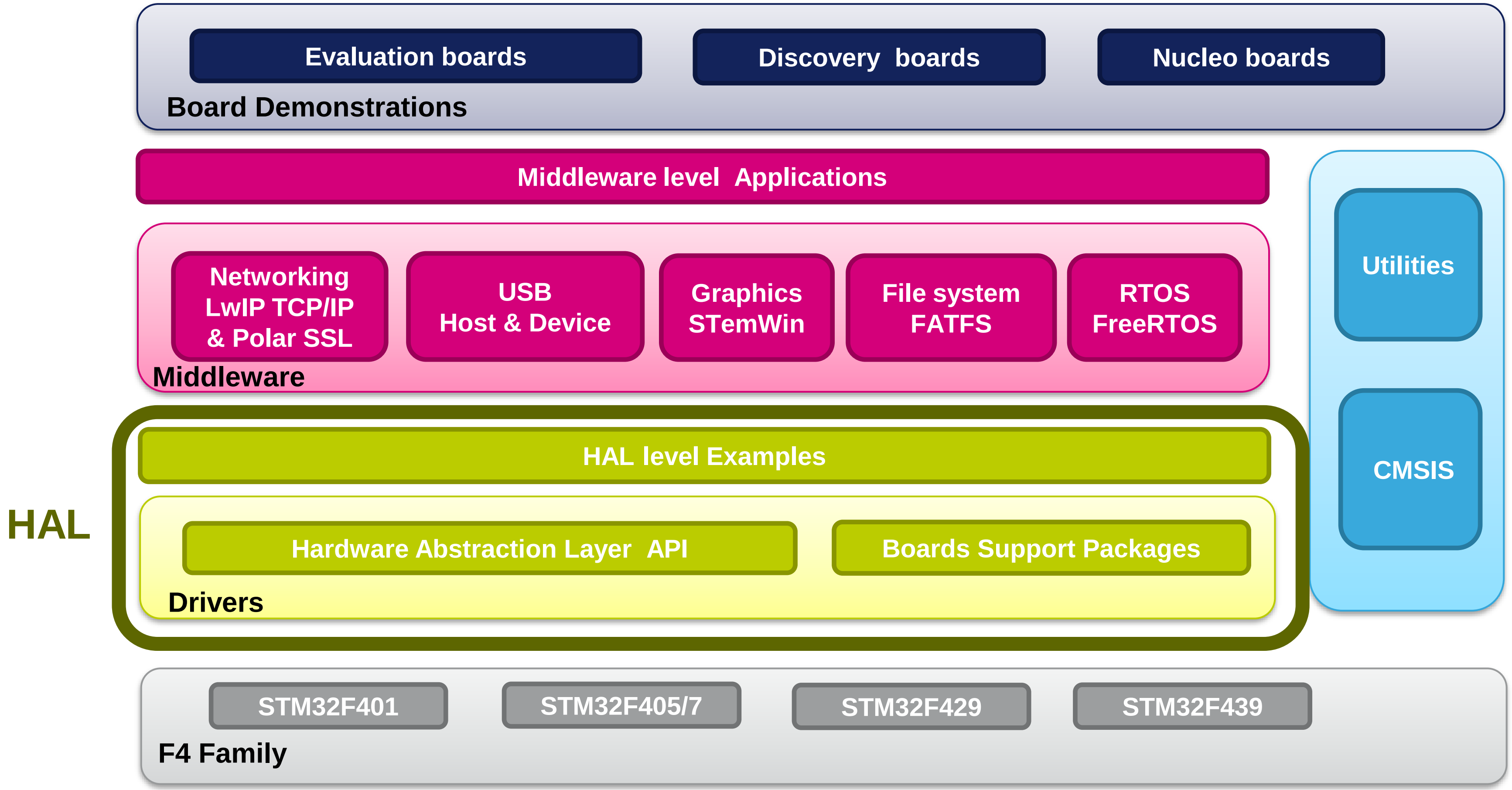
STM32Cube vs. StdLib

- Existing Standard Peripheral Libraries (StdLib) will be supported, but not recommended for new designs
- StdLib will be updated to support new F0/F3/F4 derivatives
- No StdLib for new Series, such as L0/L4/F7
- Migrating an application from StdLib to STM32Cube:
  - The 2 solutions are completely different (different architecture, APIs set…) and thus there is no direct migration path, i.e. no “find and replace” possibility.
  - Customers needs to rewrite their applications developed on StdLib to work with STM32Cube.
  - Combined with the fully portable HAL, user will do the migration to STM32Cube only once then can easily migrate to any other MCU with more performance, memory, and peripherals without rewriting the application. As a result, developers can leverage the same application and toolchain across an entire product line and a variety of MCUs.

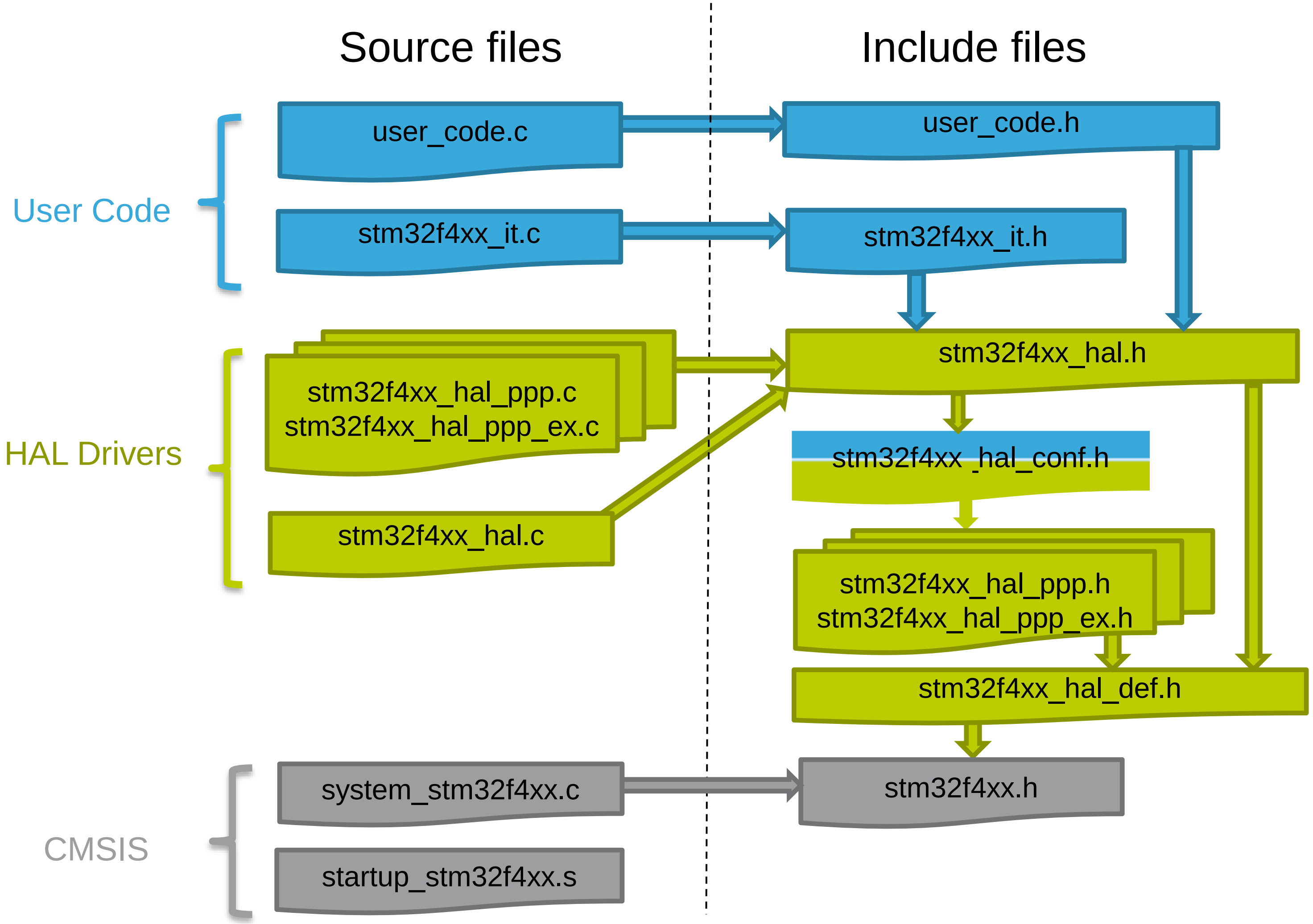
STM32Cube Hardware Abstraction Layer (HAL)

HAL general concepts

HAL general concepts
The HAL in STM32Cube FW package



HAL general concepts
HAL based project organization



life.augmented

HAL general concepts
HAL configuration file

- The HAL config file stm32f4xx_hal_conf.h allows to select the modules to include:

```
/* ########################### Module Selection ############################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED
//#define HAL_ADC_MODULE_ENABLED
//#define HAL_CAN_MODULE_ENABLED
//#define HAL_CRC_MODULE_ENABLED
//#define HAL_CRYP_MODULE_ENABLED
//#define HAL_DAC_MODULE_ENABLED
//#define HAL_DCMI_MODULE_ENABLED
//#define HAL_DMA2D_MODULE_ENABLED
//#define HAL_ETH_MODULE_ENABLED
//#define HAL_NAND_MODULE_ENABLED
//#define HAL_NOR_MODULE_ENABLED
```

Commented modules will be **not included** into project this may cause errors

- It defines also some system and HAL parameters including
  - HSE clock/HSI clock values
  - Instruction/data cache and prefetch queue setting
  - VDD voltage value

HAL general concepts Callbacks

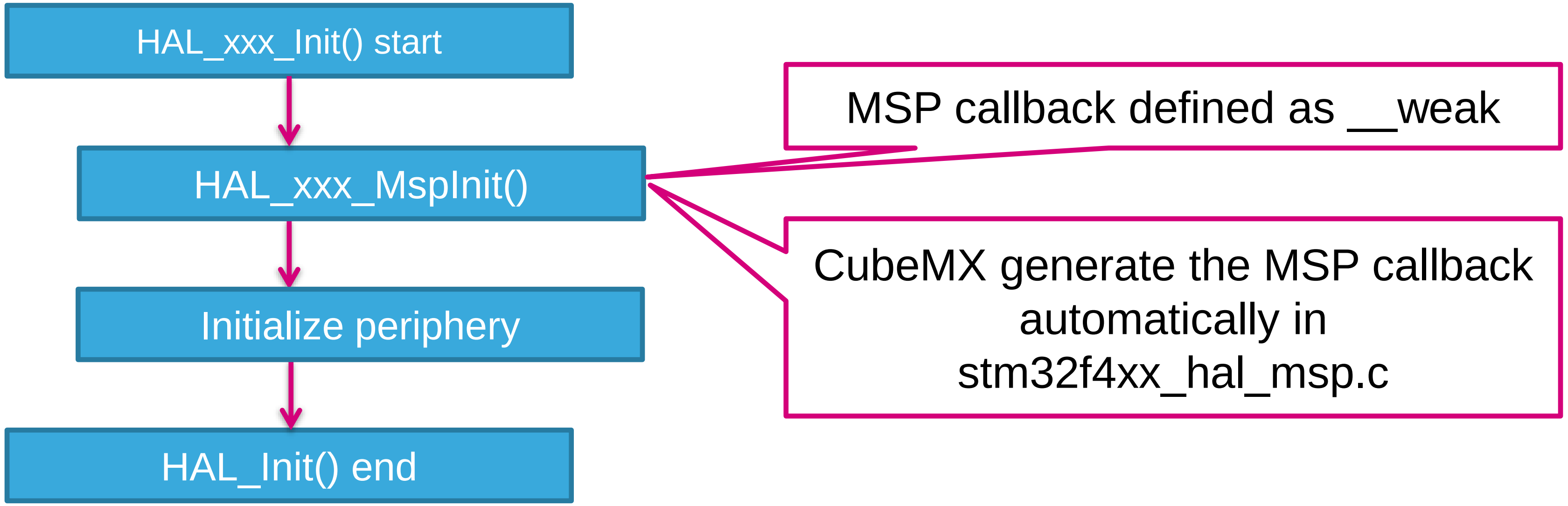
- Cube HAL library use the callbacks
  - To inform application about interrupts
  - About periphery initialization/deinitialization
- The callback functions are defined as __weak
- You can find them in stm32f4xx_hal_xxx.c

SPI callback are in stm32fxxx_hal_spi.c or in stm32fxxx_hal_spi_ex.c

```
/**
  * @brief Tx Transfer completed callbacks
  * @param  hspi: pointer to a SPI_HandleTypeDef structure that contains
  *                the configuration information for SPI module.
  * @retval None
  */
__weak void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
{
  /* NOTE : This function Should not be modified, when the callback is needed,
            the HAL_SPI_TxCpltCallback could be implemented in the user file
   */
}
```

HAL general concepts
Init functions

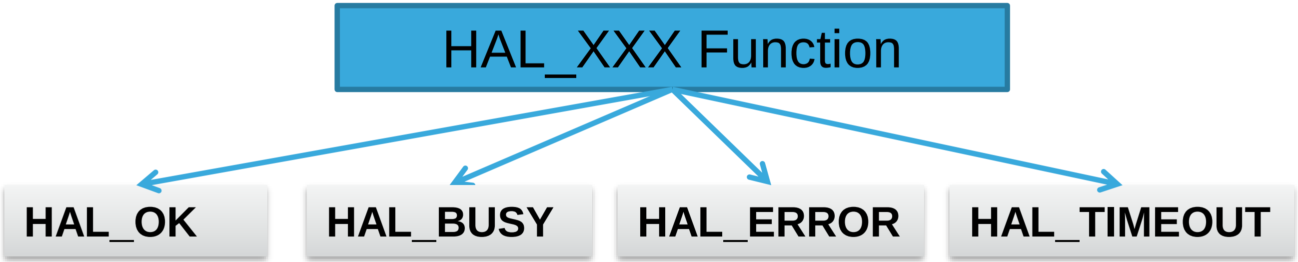
- HAL_XXX_Init() functions
- Inside init function are written data from input parameters/structure into registers
- Before the register write is processed HAL_XXX_MspInit callback is called



life.augmented

HAL general concepts
HAL API returns parameters

- HAL API can return a value of enumerated type HAL_StatusTypeDef:
  - HAK_OK : API executed with success
  - HAL_ERROR : API call parameters error or operation execution error
  - HAL_BUSY : API was not executed because peripheral is busy with other operation
  - HAL_TIMEOUT : API timeout error



Low Layer drives

Context

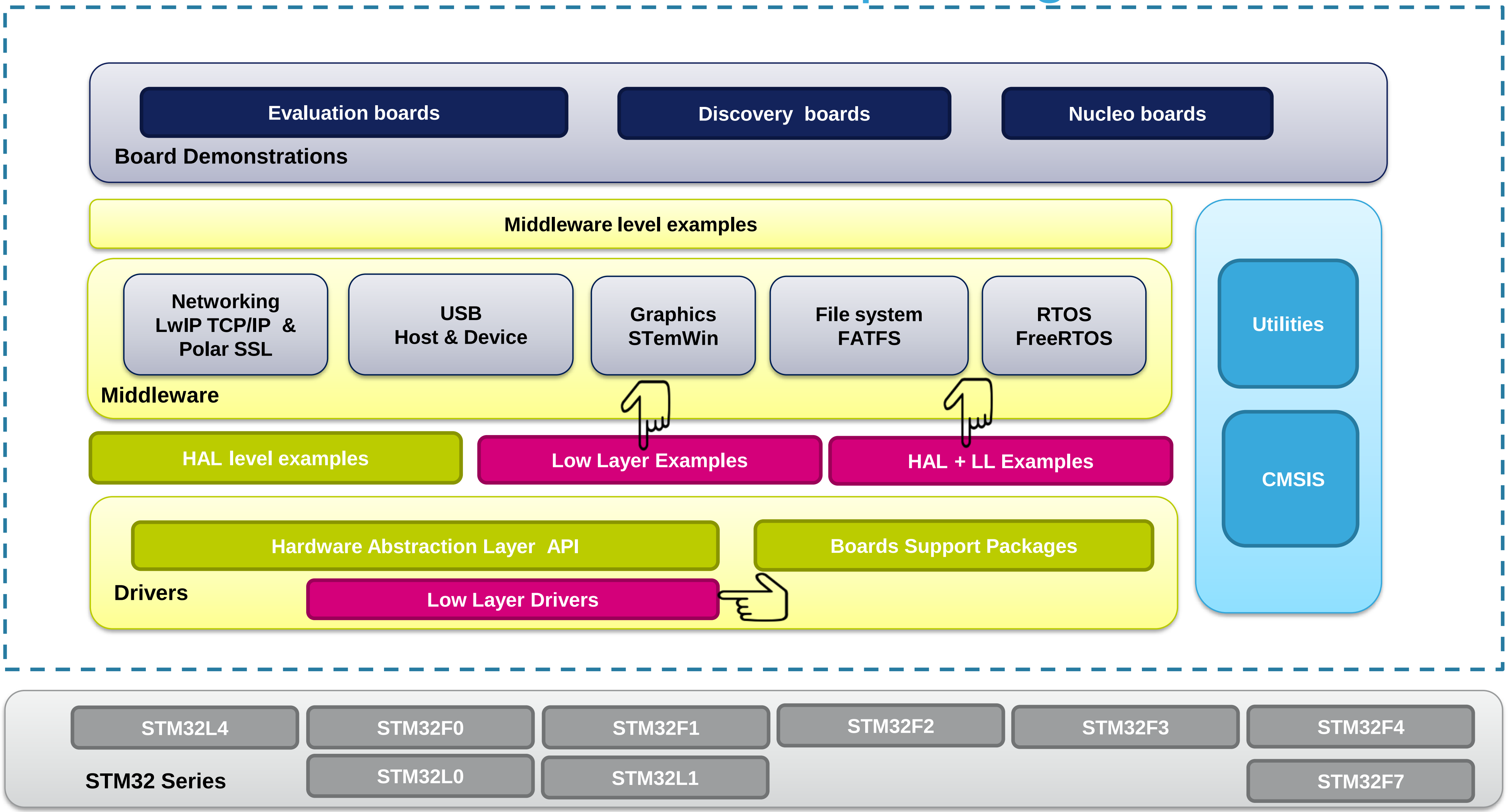
- **February 2014: 1st Release of STM32CubeV1 solution, based on F4 Series**

- After this release, we received some **concerns about the HAL**, mainly:
  - Big code called at peripheral initialization
  - Need to go-through full peripheral initialization even if slight modification is needed at run-time, making it inefficient
  - User have limited access to the peripheral resources and control
  - Missing level equivalent to SPL

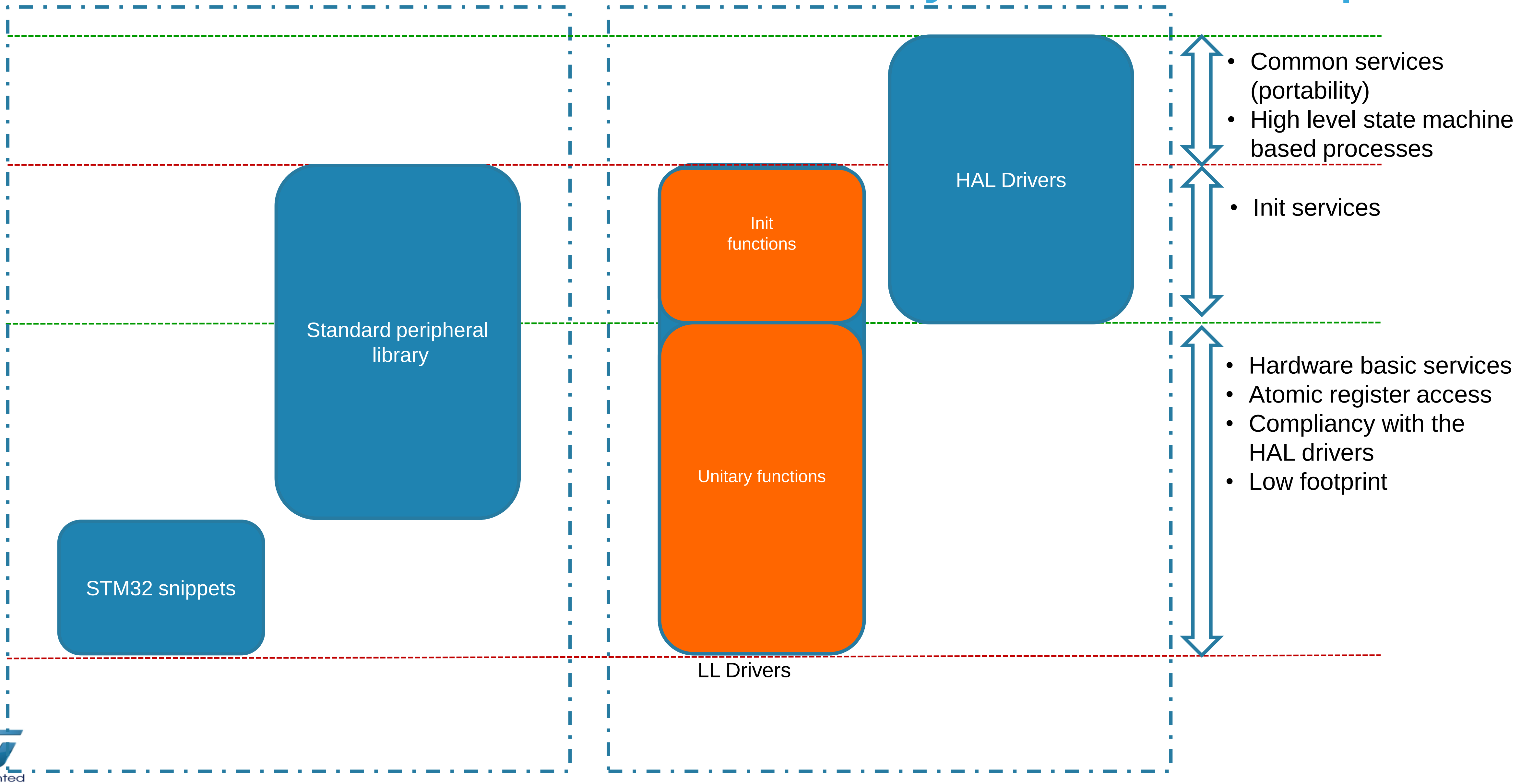
  ➔ **New *Low Layer (LL) APIs* will be the answer**

- **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**

STM32Cube FW package block view

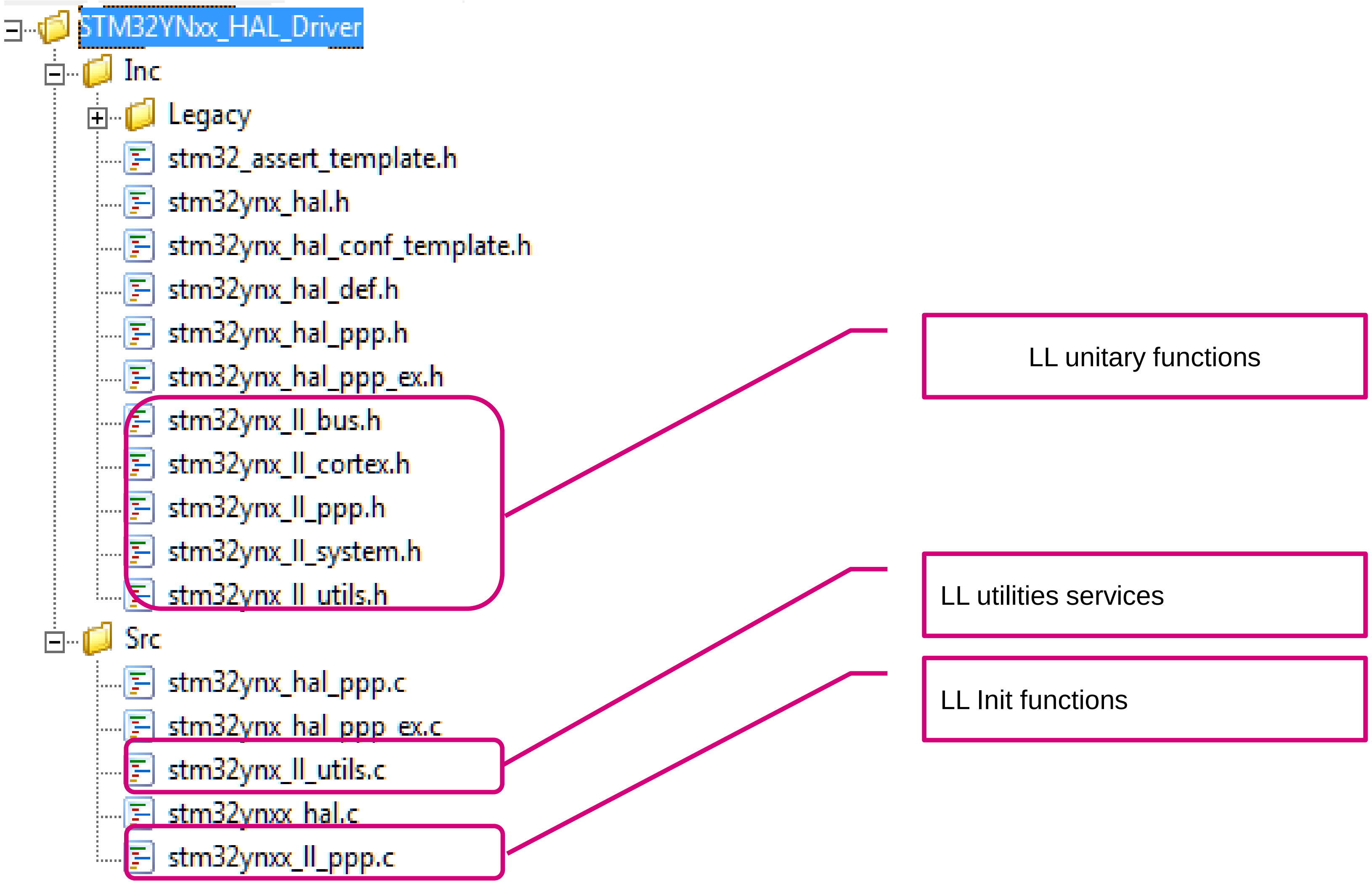


Low Layer Drivers scope



LL Files

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

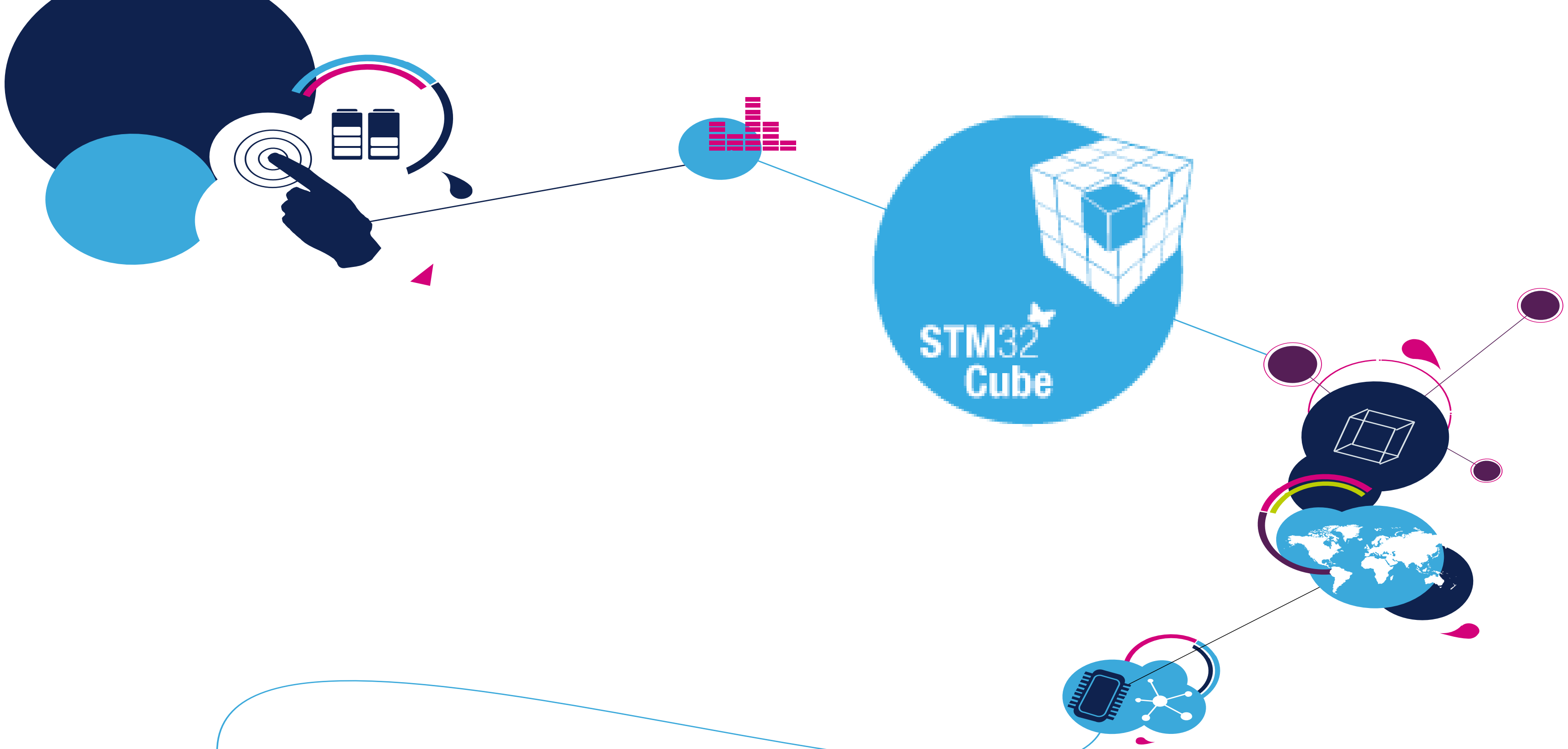


LL Typical Usage

- The low layer services have to be called *following the programming model of the reference manual document* by calling the *elementary* low layer drivers services
    - ***Ex : Use GPIO To toggle continuously a LED***

```c
/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* Configure the system clock to 80 MHz */
  SystemClock_Config();

  /* -2- Configure IO in out
  Configure_GPIO();

  /* Toggle IO in an infinite loop */
  while (1)
  {
    LL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);

    /* Insert delay 250 ms */
    LL_mDelay(250);
  }
}
```

```c
__STATIC_INLINE void Configure_GPIO(void)
{
  /* Enable the LED2 Clock */
  LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOA);

  /* Configure IO in output push-pull mode to drive external LED2 */
  LL_GPIO_SetPinMode(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_MODE_OUTPUT);
  LL_GPIO_SetPinOutputType(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_OUTPUT_PUSHPULL);
  LL_GPIO_SetPinSpeed(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_SPEED_LOW);
  LL_GPIO_SetPinPull(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_PULL_NO);
}
```



Thank you!

life.augmented