

# STM32CubeMX

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

- CubeMX tool

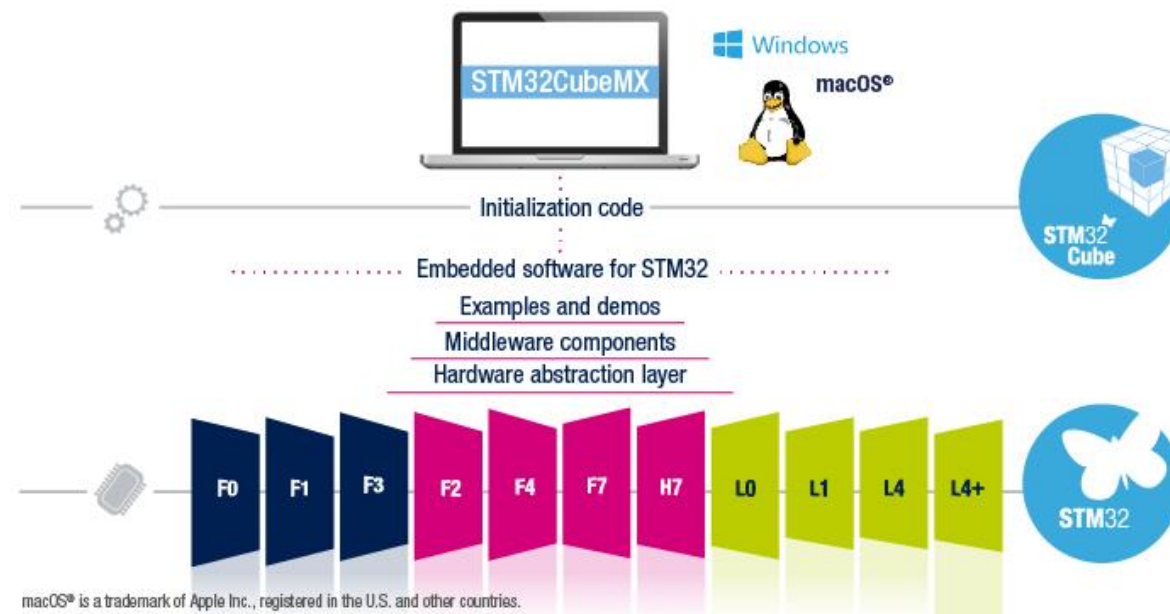
- [http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?s\\_searchtype=partnumber](http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?s_searchtype=partnumber)

- 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](http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1533/PF259242?s_searchtype=partnumber)

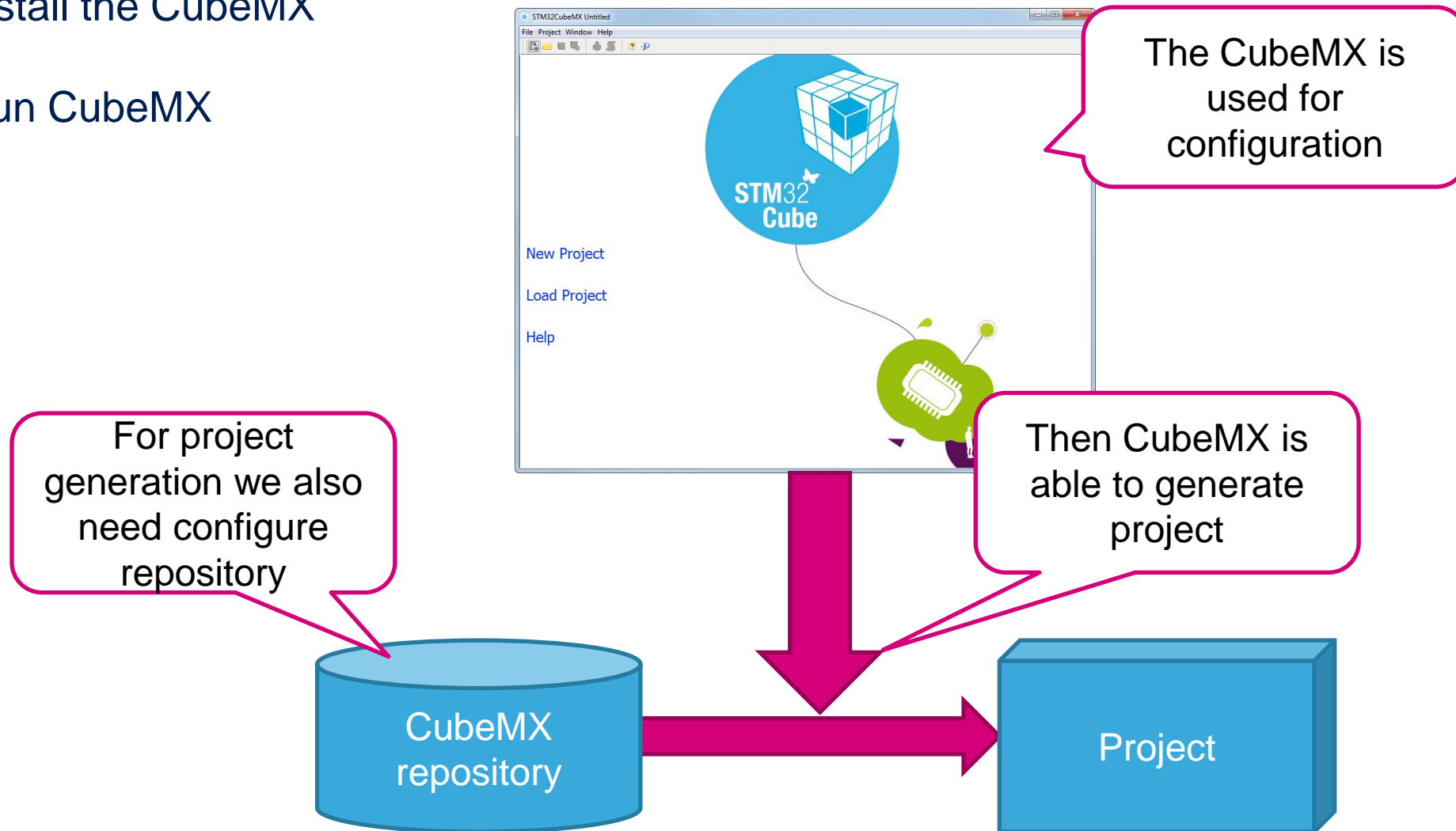


- The CubeMX can generate the code for some IDE
  - Atollic TrueSTUDIO
  - IAR
  - Keil
  - System Workbench
- For the debugging is necessary to have the ST-Link drivers
  - STSW-LINK009 driver for Win XP/Vista/7/8/10  
<http://www.st.com/web/en/catalog/tools/PF260219>
- For driver installation you will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



# CubeMX repository configuration

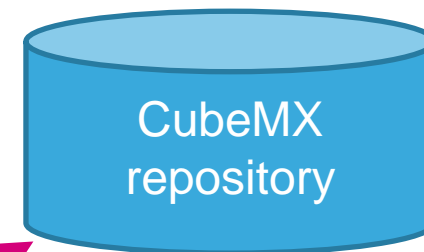
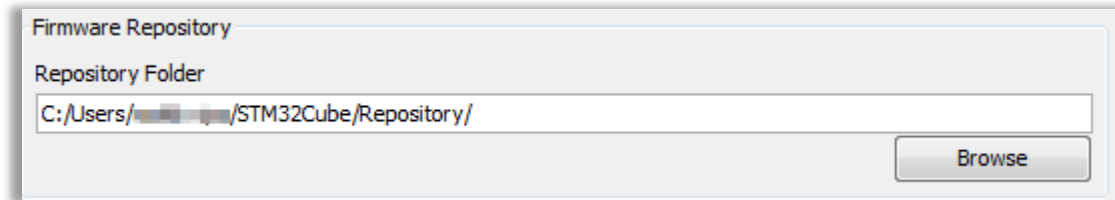
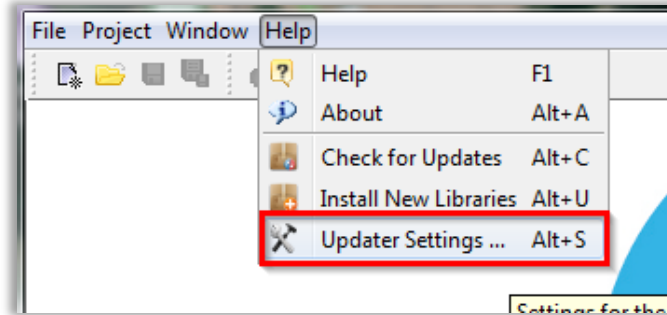
- Install the CubeMX
- Run CubeMX



# CubeMX repository configuration

5

- 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

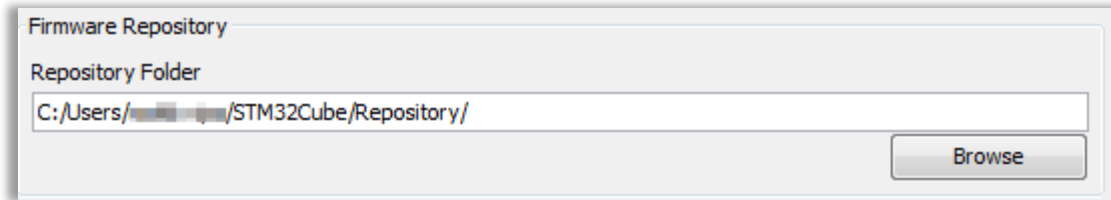


Specifying path to repository

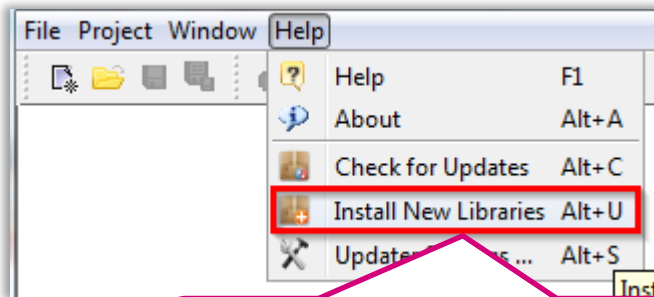
The "/" in the end of Repository is necessary

# CubeMX repository configuration

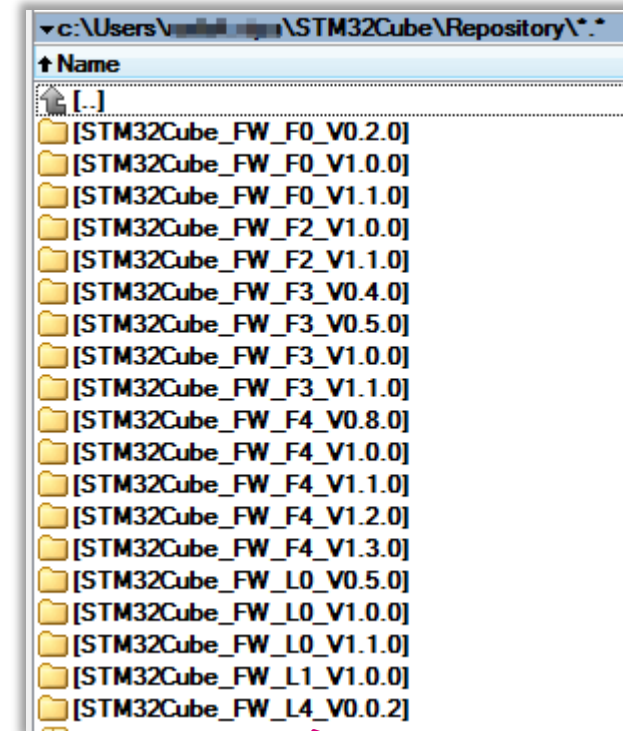
- The comparison of the CubeMX repository settings and structure in this folder



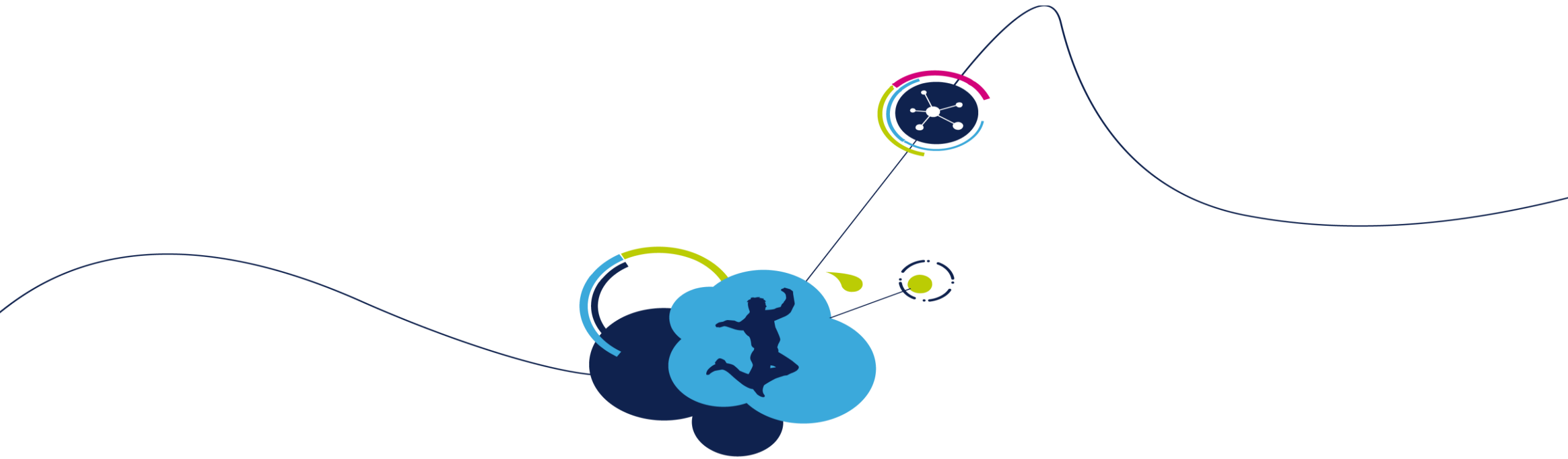
- In case you want to download this files automatically use in CubeMX
  - MENU>Help>Install New Libraries
  - Select libraries which you want
  - Force download with button Install Now



CubeMX can download for you the repository packages automatically



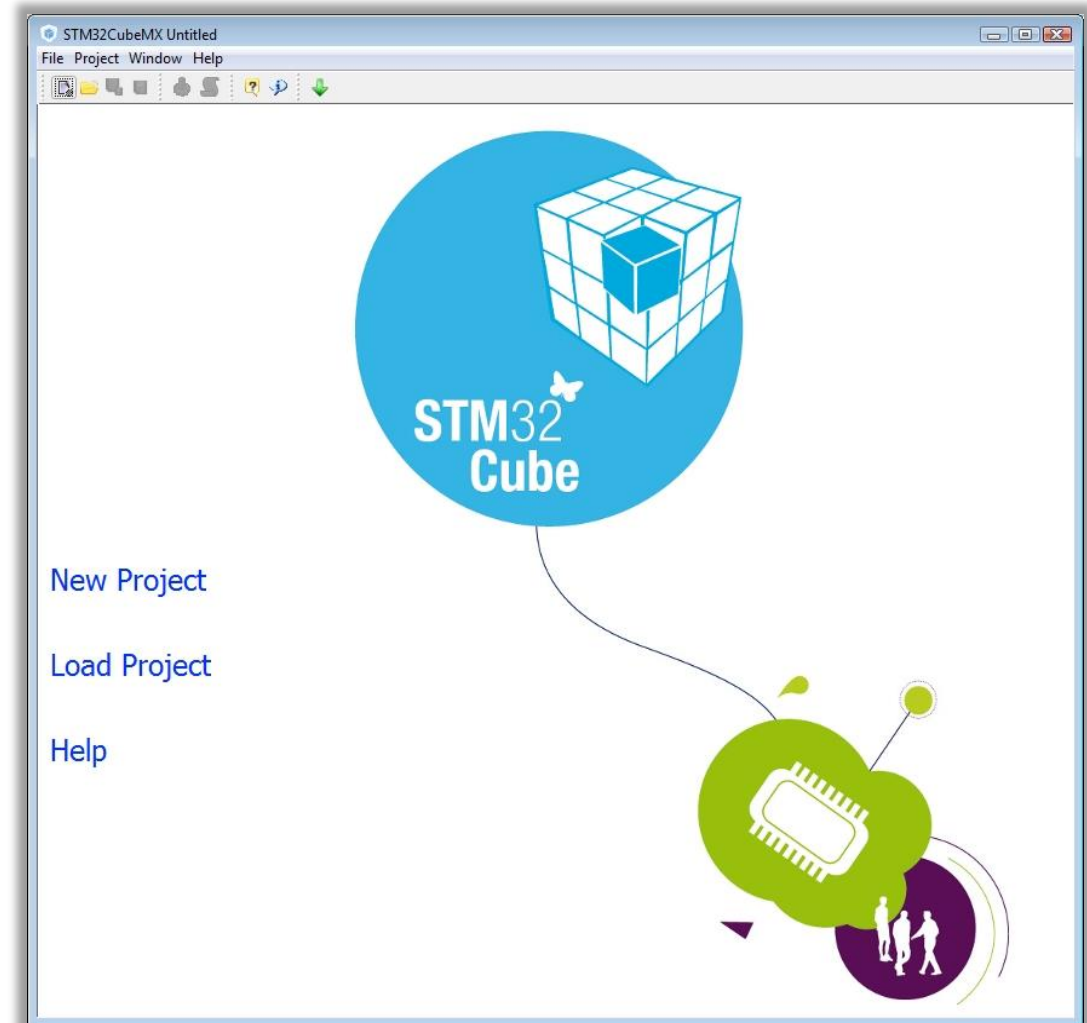
Example how the repository structure looks like



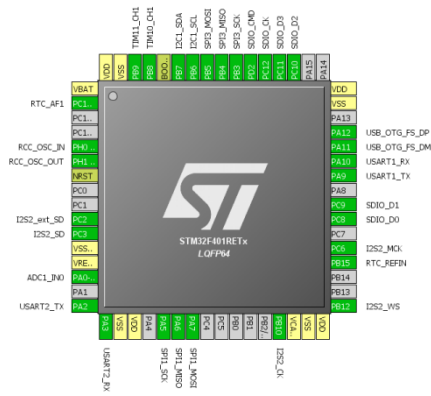
# STM32CubeMX presentation

Step by step:

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







Pinout Wizard

# STM32CubeMX



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

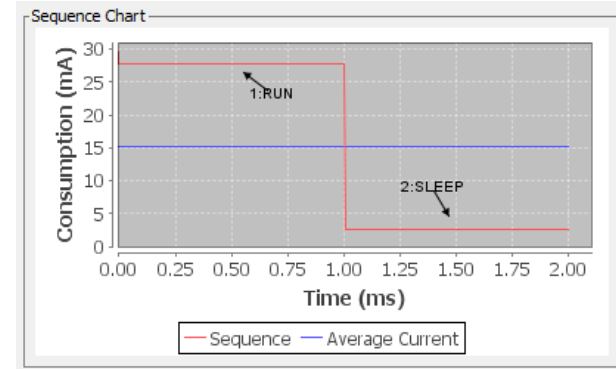
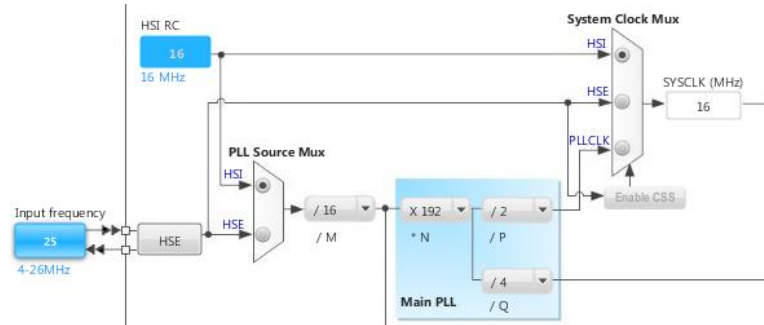
**Baud Rate**  
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Peripherals & Middleware Wizard



Power Consumption Wizard

Clock Tree wizard



# STM32CubeMX: MCU Selector

Easy Optional filtering:

- Series
- Line
- Package
- Peripherals

MCU Filters

Part Number Search

Core

Series

Line

Package

Advanced Choice

Graphic Choice

Peripheral Choice

Peripherals	Nb	Max
ADC 12-bit	0	40
ADC 16-bit	0	21
AES		
CAN	0	1
COMP	0	7
CRYP		
DAC 12-bit	0	3

STM32F302C8

Mainstream Mixed signals MCUs ARM Cortex-M4 core with DSP and FPU, 64 Kbytes Flash, 72 MHz CPU, 12-bit ADC 5 MSPS, Comparator, Op-Amp

Unit Price

Fea... Block Dia... Data... Docs & Reso... Start Pr...

MCUs List: 91 items

Display similar items

* Part No	Refe...	Mark...	U...	...	Package	Fla...	RAM	IO	Fr...	G...
STM32F301K6	STM32...	Active	1.272		UFQFPN32	32 kB...	16 k...	24	72 ...	0.0
STM32F301K8	STM32...	Active	1.342		LQFP32	64 kB...	16 k...	25	72 ...	0.0
STM32F301K8	STM32...	Active	1.342		UFQFPN32	64 kB...	16 k...	24	72 ...	0.0
STM32F301R6	STM32...	Active	1.758		LQFP64	32 kB...	16 k...	51	72 ...	0.0
STM32F301R8	STM32...	Active	1.828		LQFP64	64 kB...	16 k...	51	72 ...	0.0
STM32F302C6	STM32...	Active	1.712		LQFP48	32 kB...	16 k...	37	72 ...	0.0
STM32F302C8	STM32...	Active	1.782		LQFP48	64 kB...	16 k...	37	72 ...	0.0
STM32F302C8	STM32...	Active	1.782		WL CSP49	64 kB...	16 k...	37	72 ...	0.0
STM32F302CB	STM32...	Active	1.99		LQFP48	128 k...	32 k...	37	72 ...	0.0
STM32F302...	STM32...	Active	2.288		LQFP48	256 k...	40 k...	37	72 ...	0.0
STM32F302K6	STM32...	Active	1.596		UFQFPN32	32 kB...	16 k...	24	72 ...	0.0
STM32F302K8	STM32...	Active	1.666		UFQFPN32	64 kB...	16 k...	24	72 ...	0.0
STM32F302R6	STM32...	Active	1.874		LQFP64	32 kB...	16 k...	51	72 ...	0.0

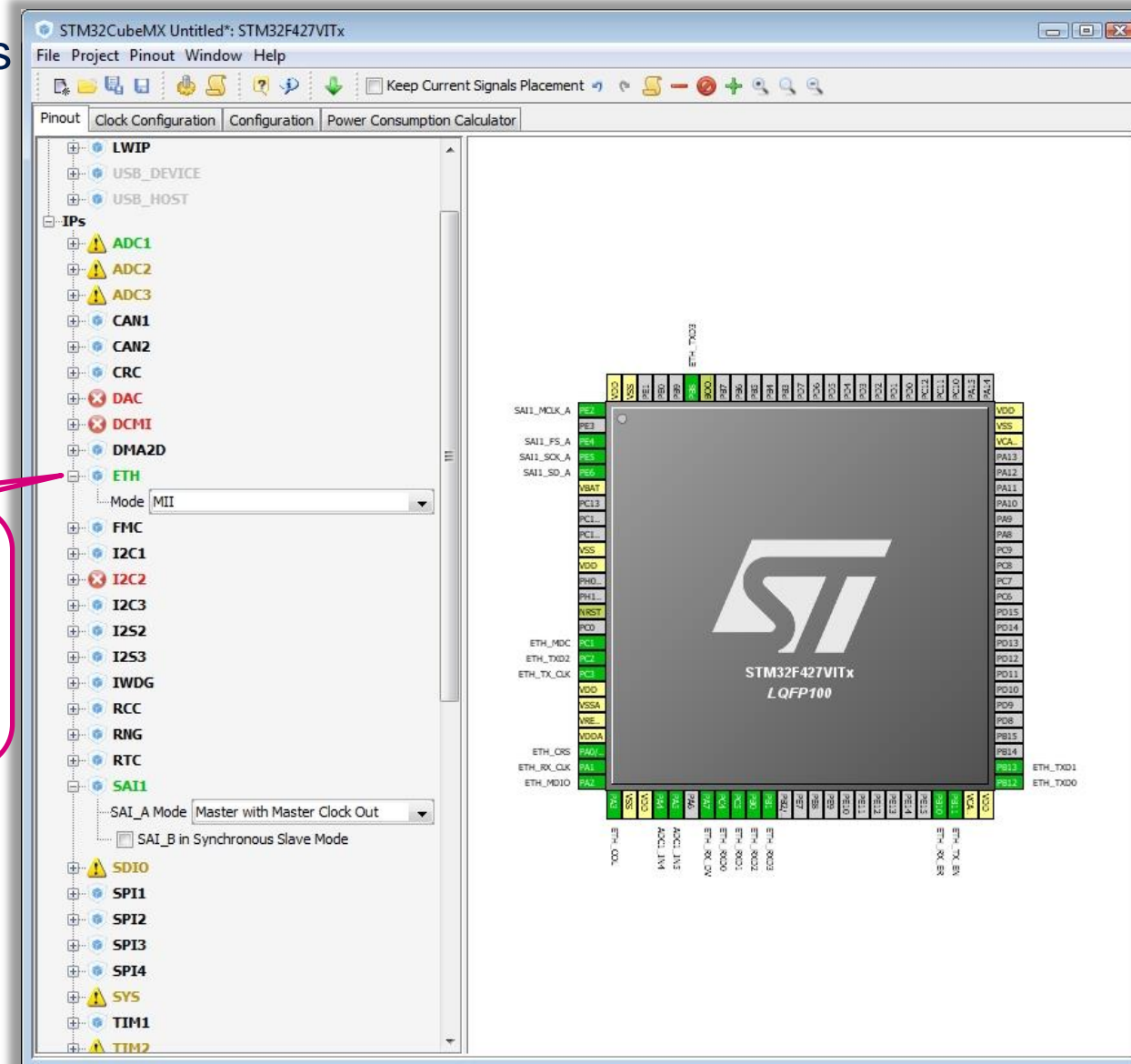




# STM32CubeMX: Pinout configuration

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

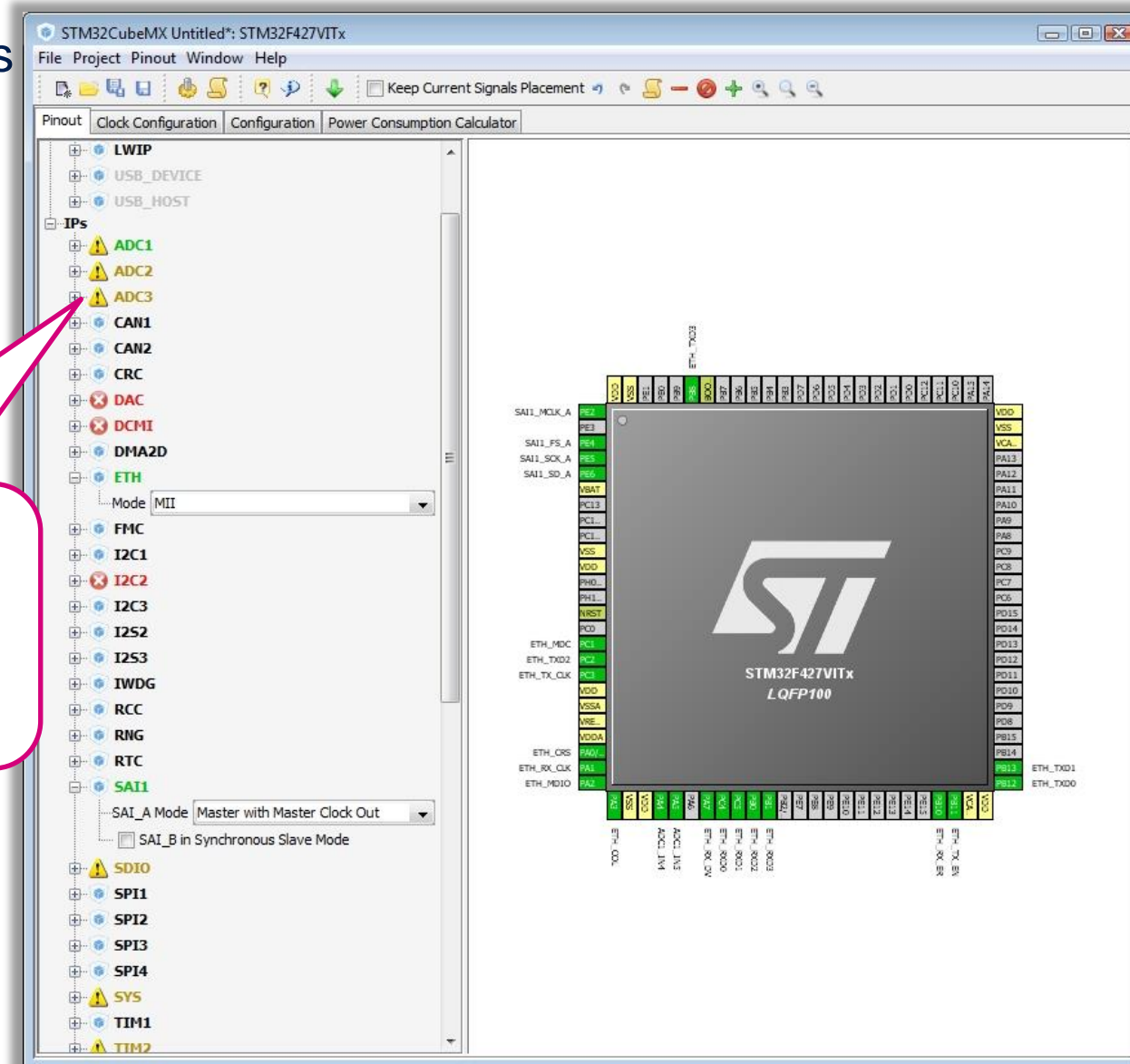
**Green:**  
Peripheral will be functional



# STM32CubeMX: Pinout configuration

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

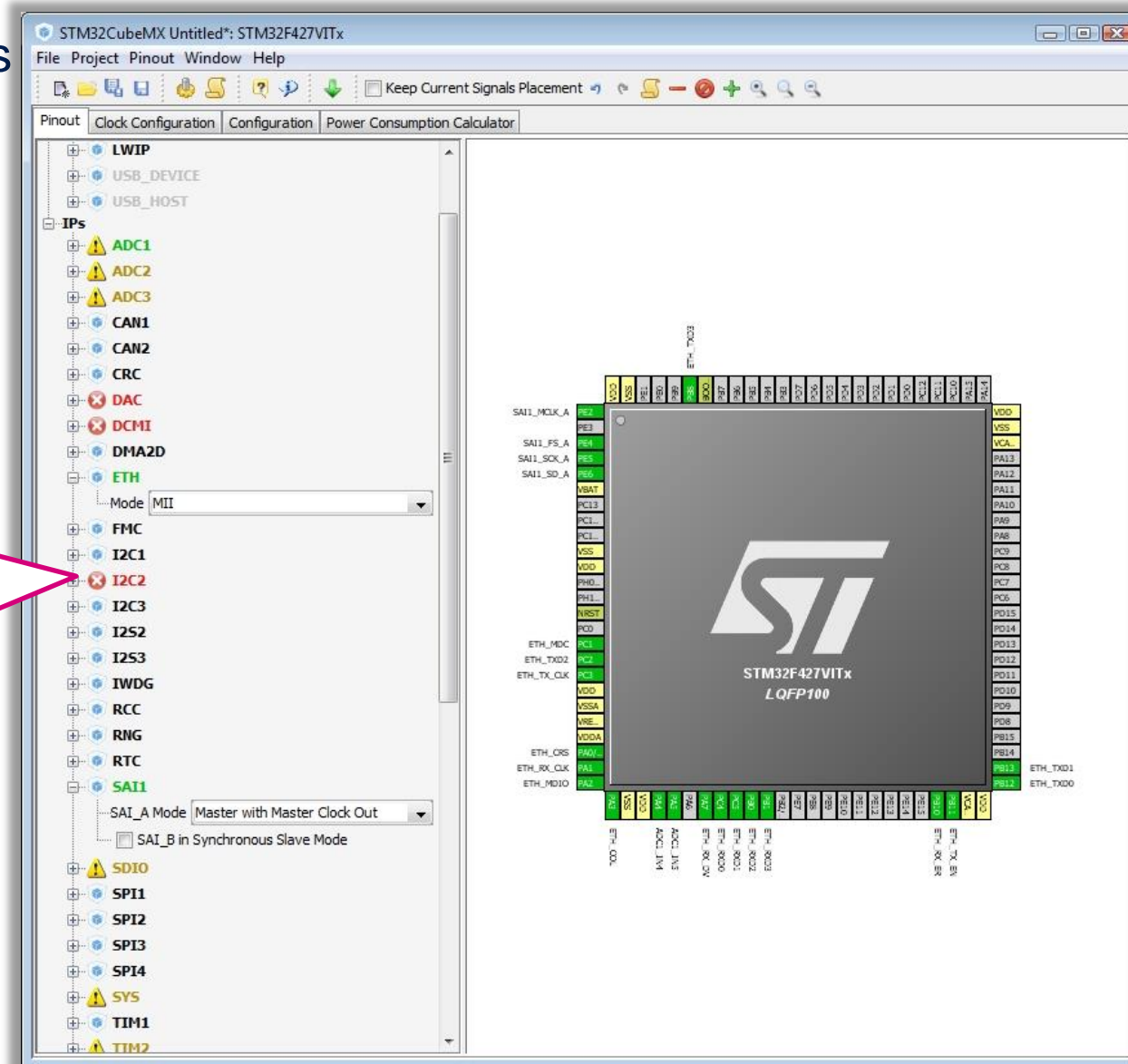
**Yellow:**  
On ADC only some channels can be used



# STM32CubeMX: Pinout configuration

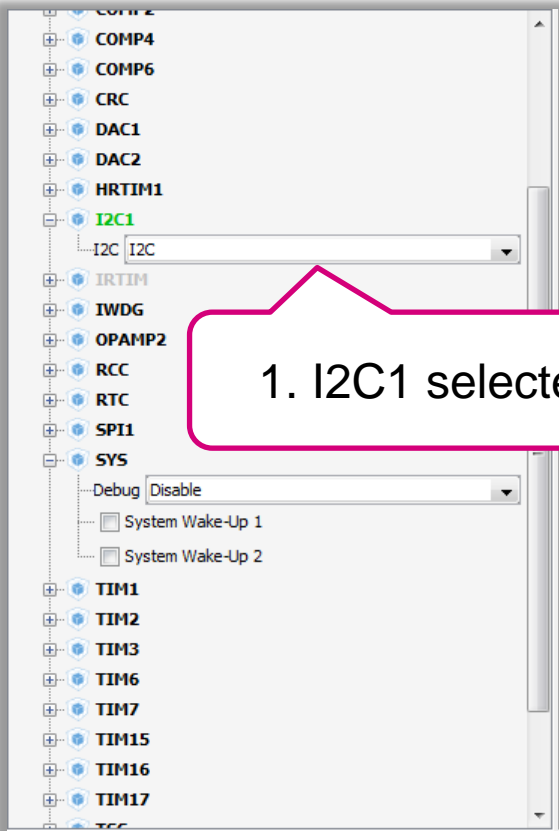
- Different possible states for a peripheral modes
  - Red:  
Signals required for this mode  
can't be mapped on the pinout  
(see tooltip to see conflicts)

**Red:**  
Peripheral cannot be  
used in this pinout  
setup

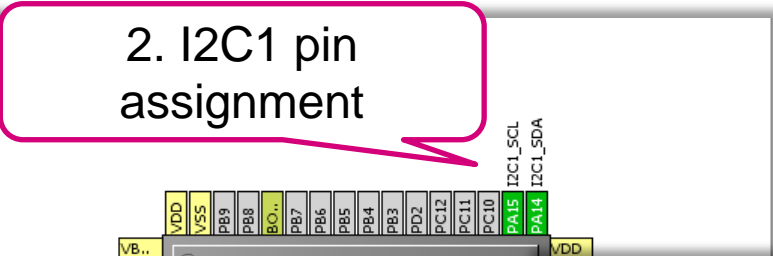


# STM32CubeMX: Pinout configuration

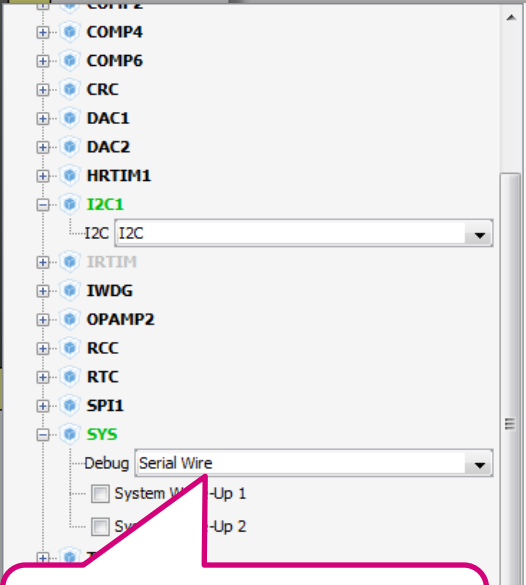
- Keep User Placement renamed to Keep Current Signal Placement and is unchecked by default



1. I2C1 selected

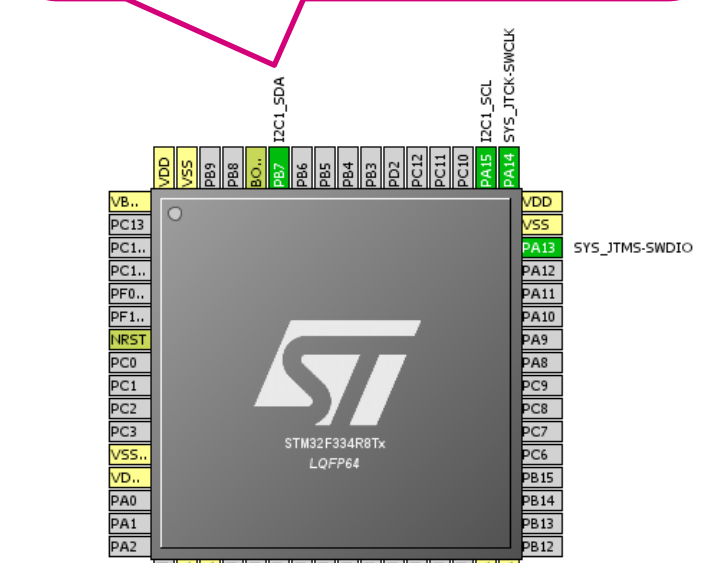


2. I2C1 pin assignment



3. SWD selected

4. Pin conflict between I2C1 and SWD. I2C1\_SDA moved to alternative position



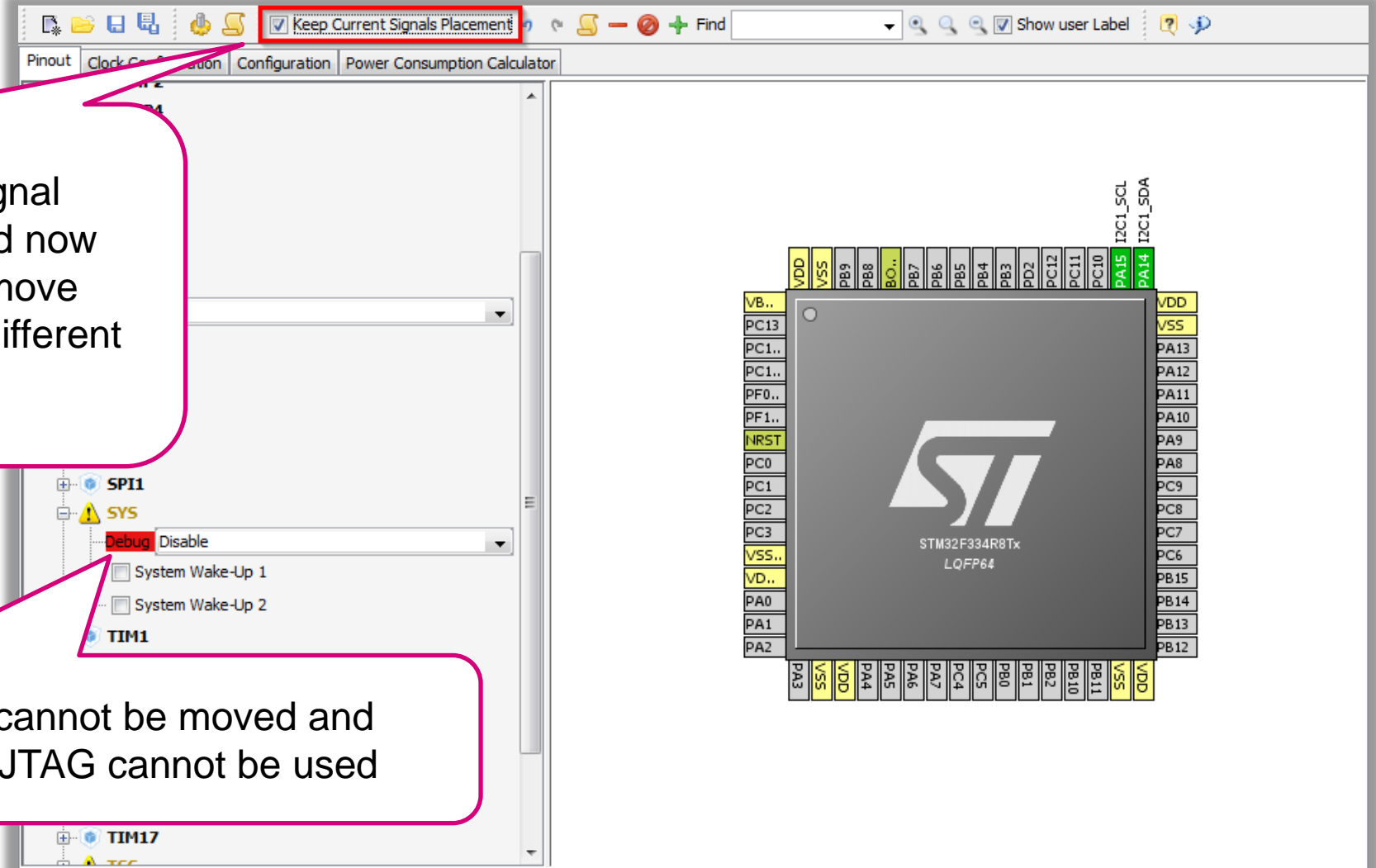


# STM32CubeMX: Pinout configuration

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

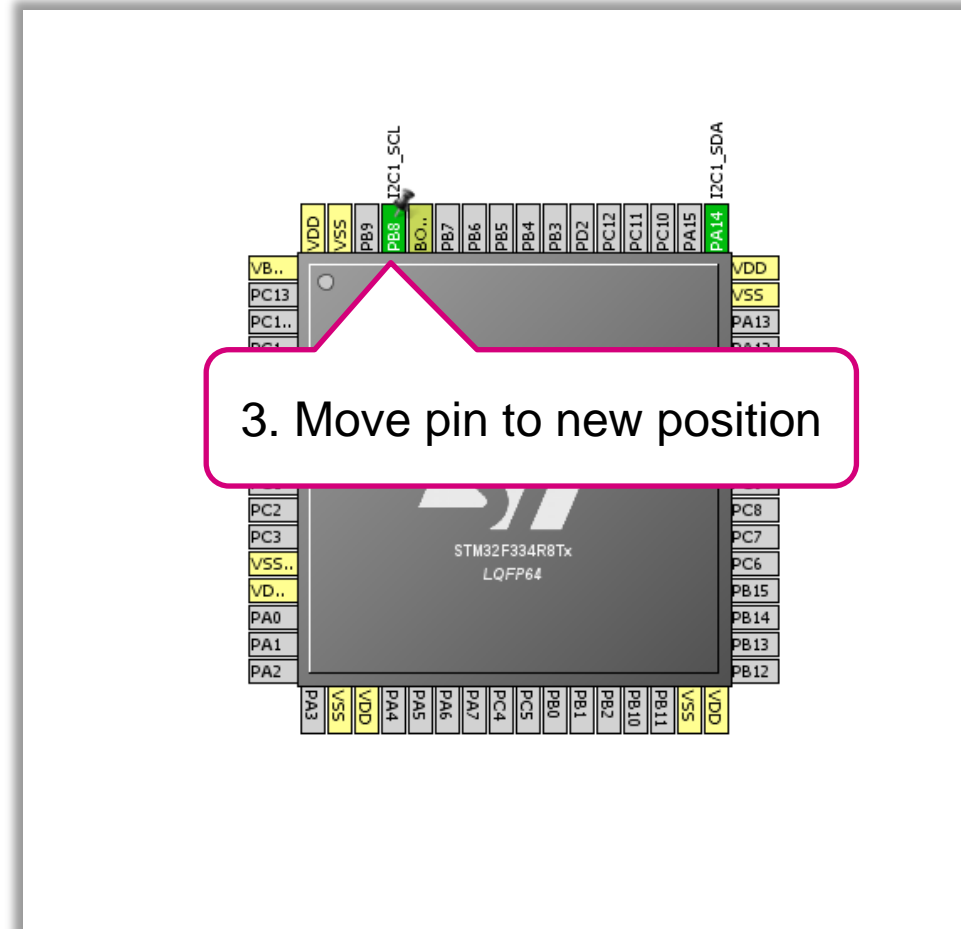
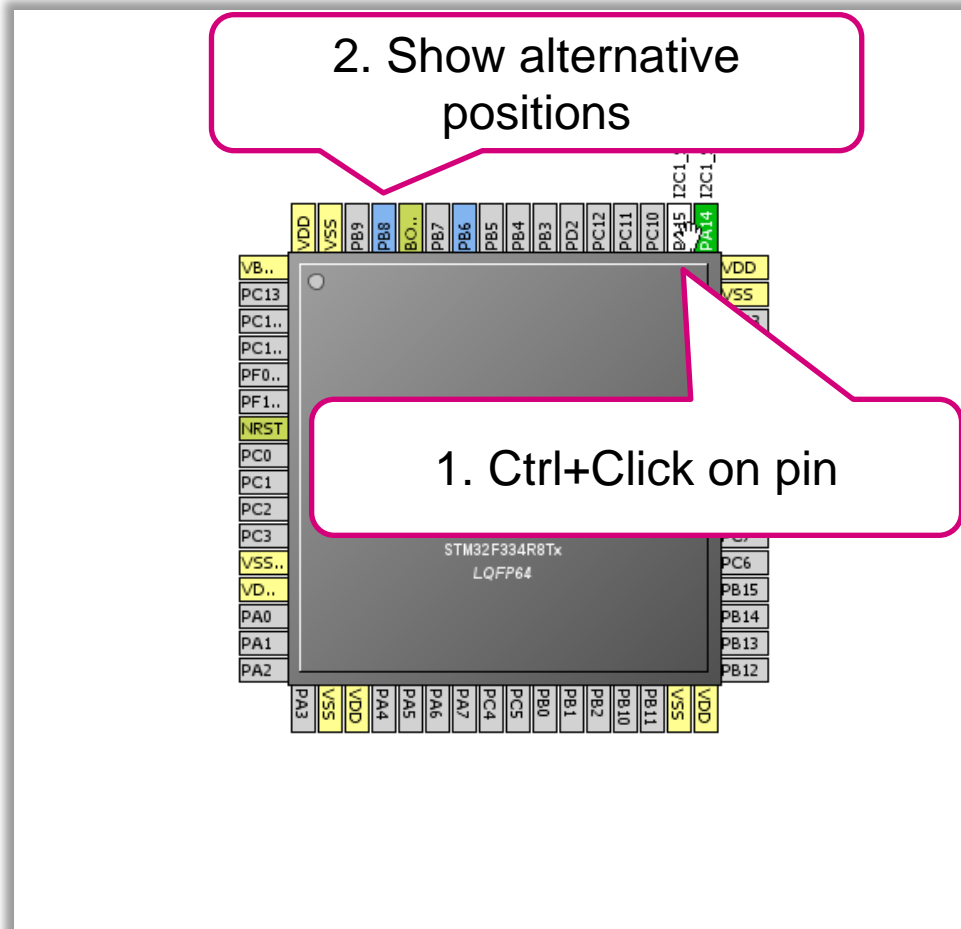
Keep Current Signal Placement checked now CubeMX cannot move selected signals to different alternate pin

I2C1 cannot be moved and SWD/JTAG cannot be used



# STM32CubeMX: Pinout configuration

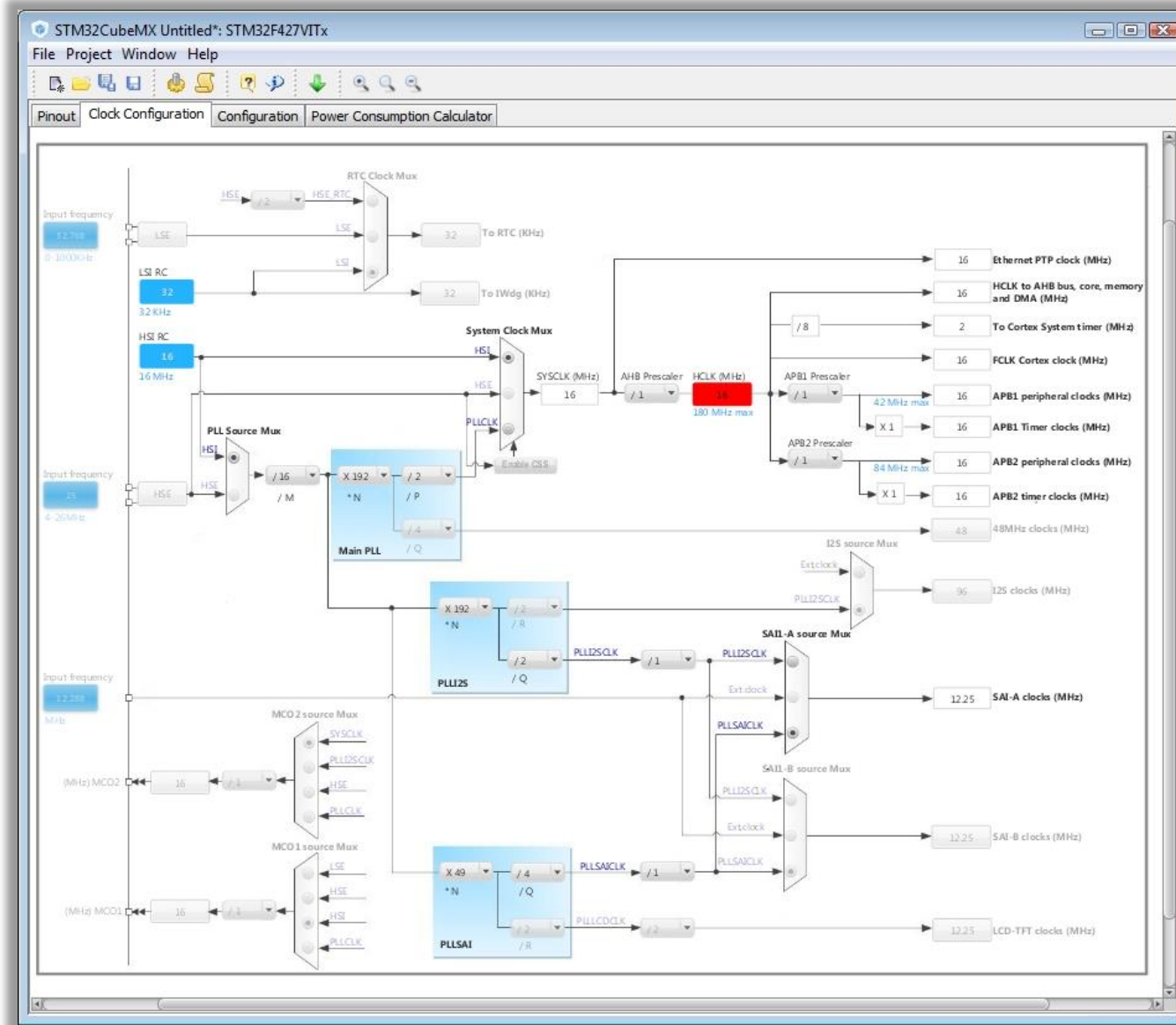
- Signals can be set/moved directly from the pinout view
  - To see alternate pins for a signal Ctrl+Click on the signal, you can then drag and drop the signal to the new pin (keep pressing the Ctrl key)



# STM32CubeMX: Clock tree

19

- 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
  - ✓ OK
  - ✗ Error
- Read only tree view on the left with access to IPs / Middleware having no impact on the pinout

The screenshot displays the STM32CubeMX configuration tool. On the left, a tree view shows the configuration status for various components:

- MiddleWares:** FREERTOS (Enabled), LWIP (Enabled).
- IPs:** ADC3 (Warning), CAN1 (OK), CAN2 (OK), CRC (OK), DCMI (Warning), DMA2D (OK), ETH (OK), FMC (Warning), IWDG (OK), RNG (OK).

The main area shows a grid of peripheral configuration cards:

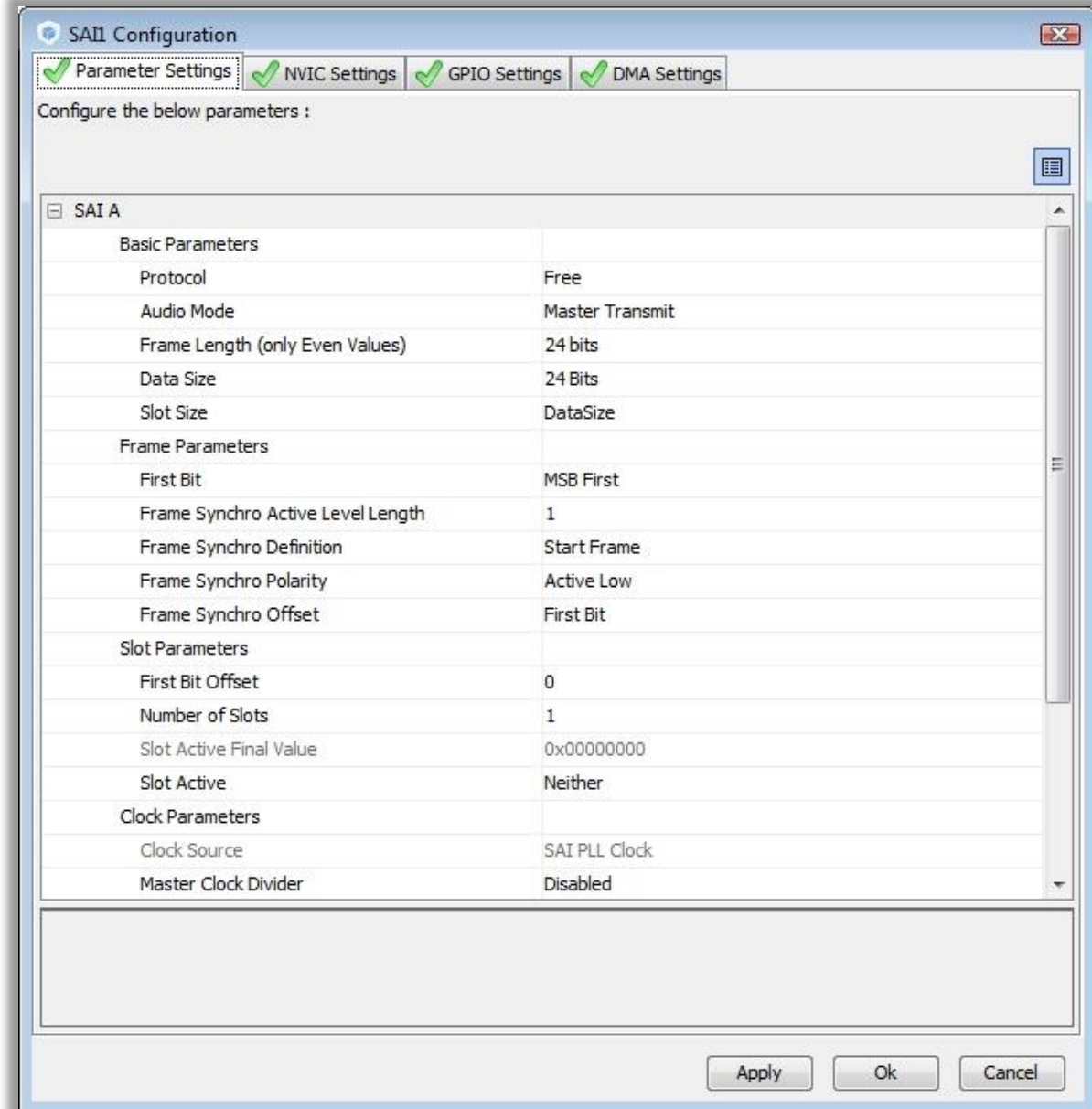
- Middlewares:** FREERTOS, LWIP.
- Multimedia:** DCMI.
- Connectivity:** CAN1, CAN2, ETH, FMC, UART4, USART1.
- Analog:** ADC3.
- System:** CRC, DMA, GPIO, NVIC, RCC.
- Control:** TIM2.

At the bottom, the MCUs Selection table is visible:

Series	Lines	McU	Package	Required Peripherals
<input type="checkbox"/> STM32F4	STM32F427/437	STM32F427IGHx	UFBGA 176	FMC
<input type="checkbox"/> STM32F4	STM32F427/437	STM32F427IIHx	UFBGA 176	FMC
<input type="checkbox"/> STM32F4	STM32F427/437	STM32F427IGTx	LQFP 176	FMC

# STM32CubeMX: Peripheral and middleware configuration

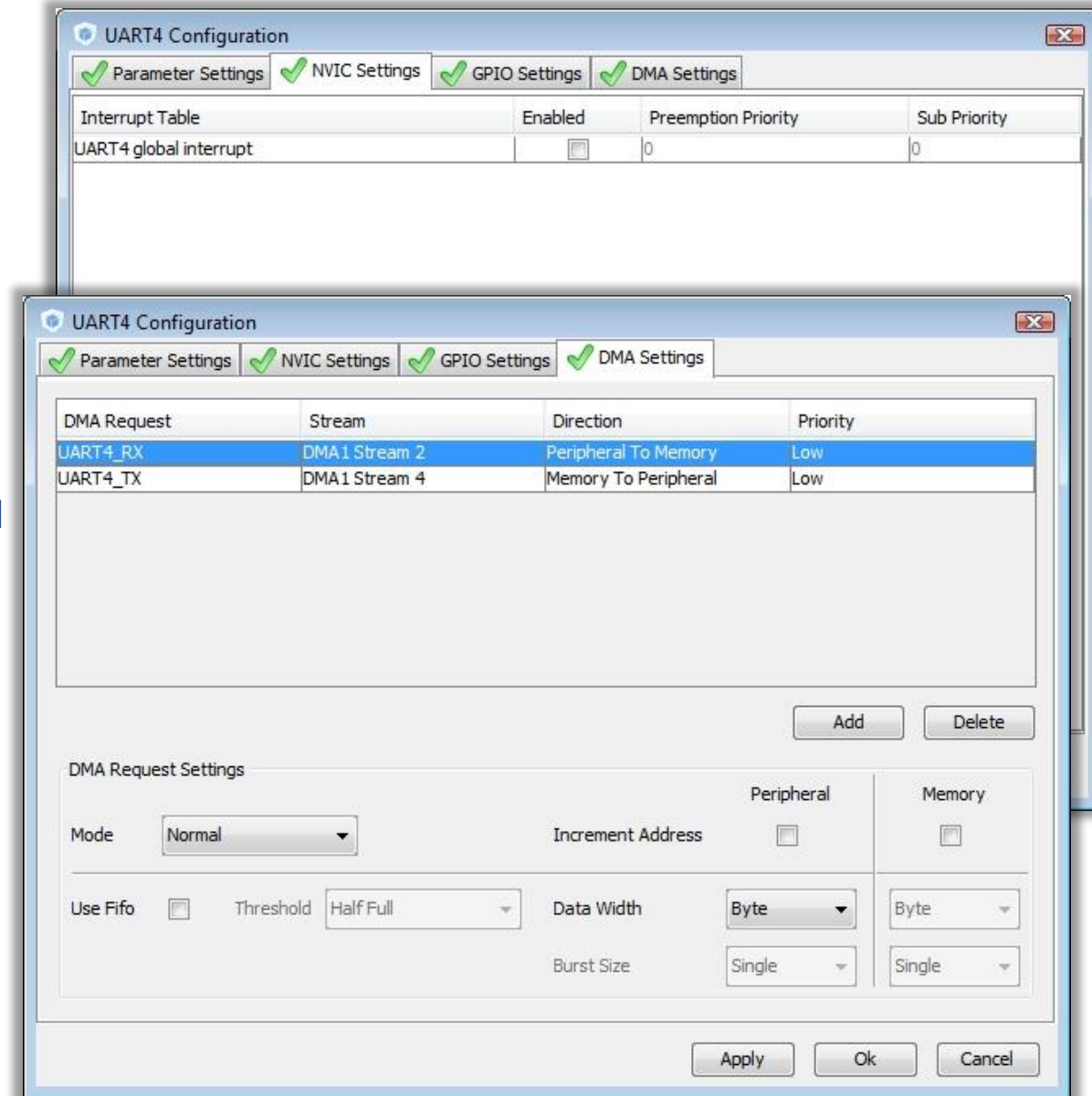
- Parameters with management of dependencies and constraints
- Interrupts
- GPIO
- DMA



# STM32CubeMX: Peripheral and middleware configuration

22

- 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

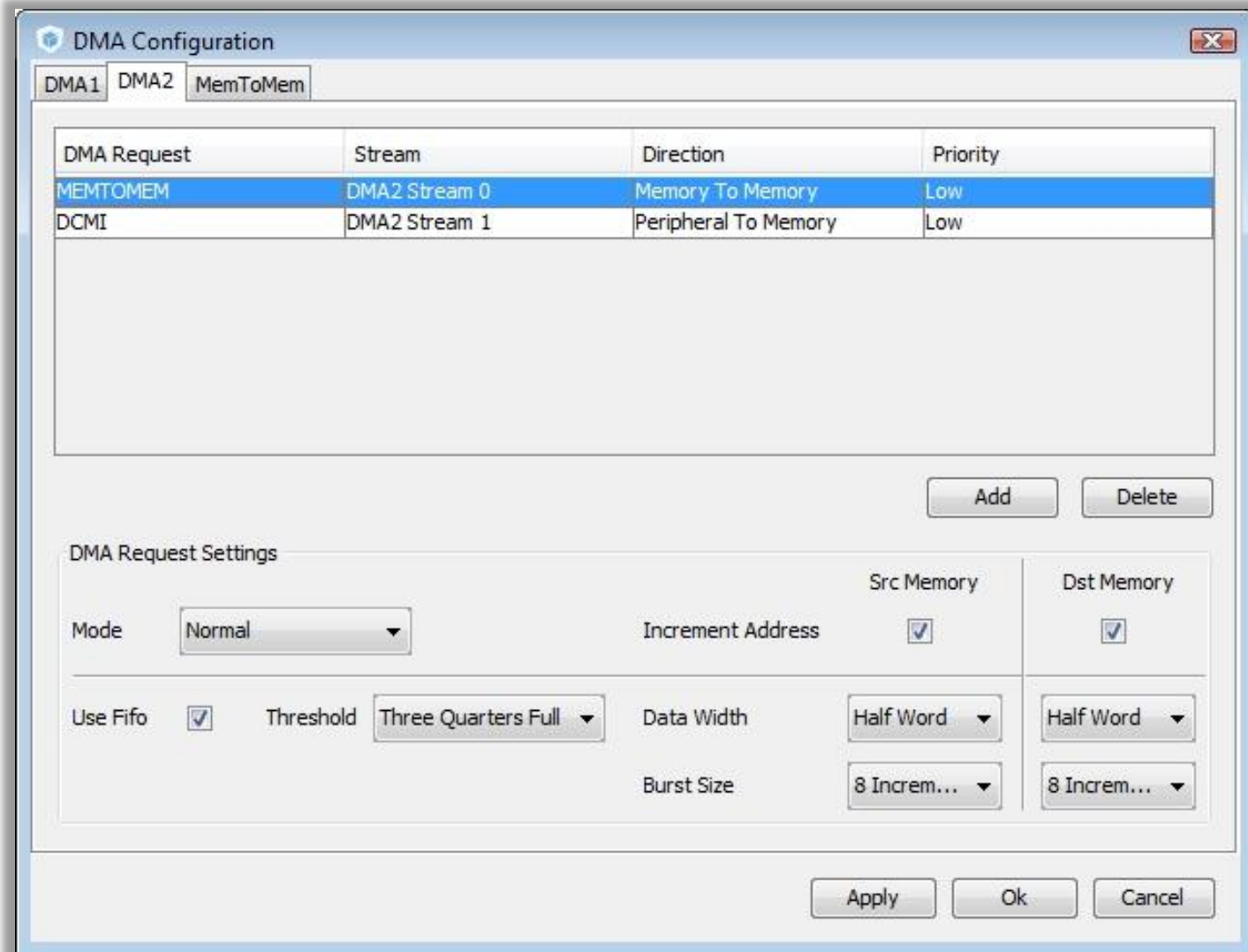


- Manage all interruptions
- Manage priorities and sort by priorities
- Search for a specific interrupt in the list

The screenshot shows the 'NVIC Configuration' window. At the top, there is a 'Priority Group' dropdown set to '0 bits for pre-emption priority 4 bits for subpriority'. To its right is a checkbox for 'Sort by Preemption Priority and Sub Priority'. Below this is a search bar with two arrow buttons (down and up) and another checkbox for 'Show only enabled interrupts'. The main area is a table with the following columns: 'Interrupt Table', 'Enabled', 'Preemption Priority', and 'Sub Priority'. The 'System tick timer' row is checked in the 'Enabled' column. At the bottom, there are controls for 'Enabled' (checkbox), 'Preemption Priority' (dropdown), and 'Sub Priority' (dropdown), along with 'Apply', 'Ok', and 'Cancel' buttons.

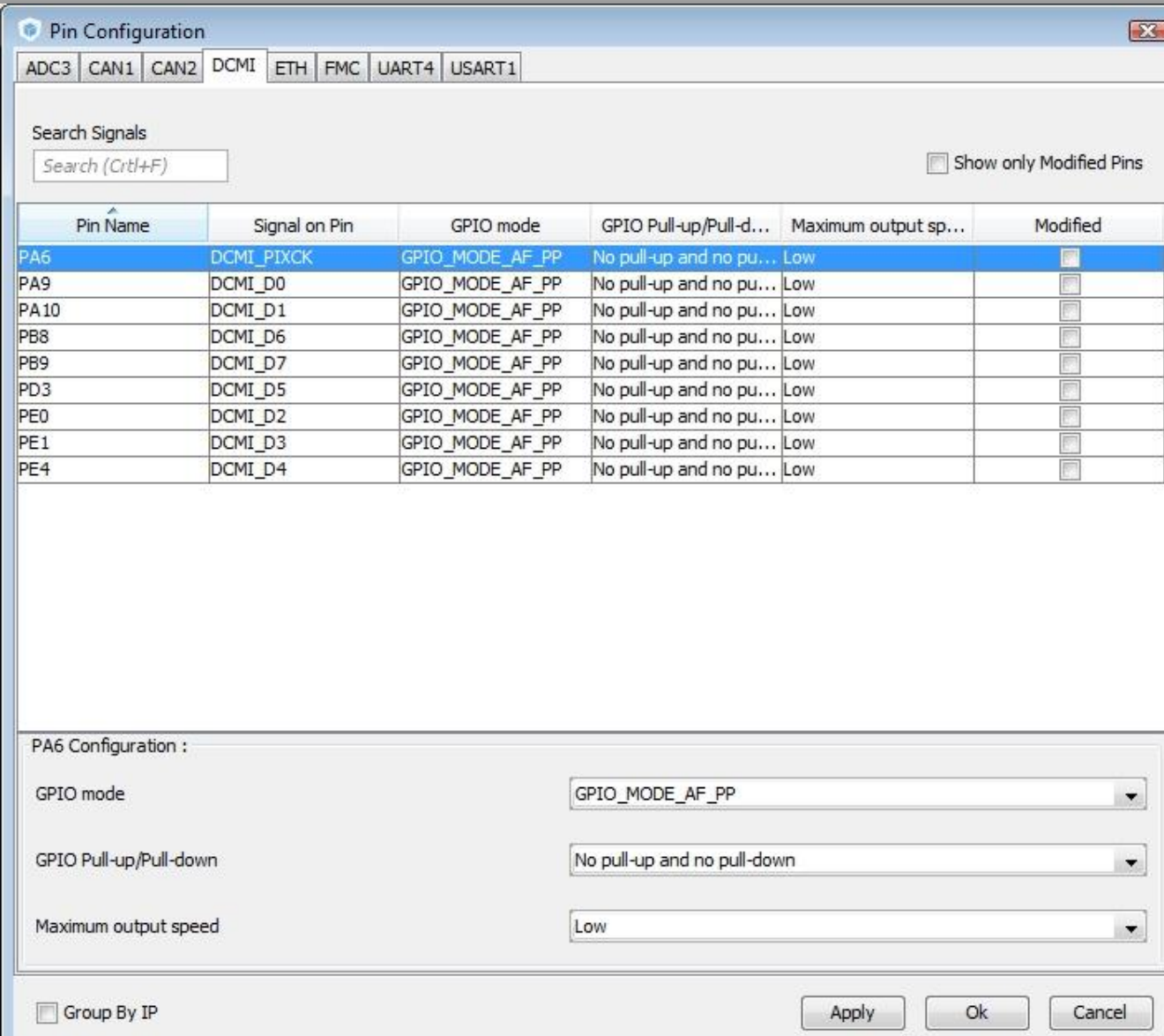
Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non Maskable Interrupt	<input type="checkbox"/>	0	0
Memory management fault	<input type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input type="checkbox"/>	0	0
Undefined instruction or illegal state	<input type="checkbox"/>	0	0
Debug Monitor	<input type="checkbox"/>	0	0
System tick timer	<input checked="" type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
ADC 1, ADC2 and ADC3 global interrupts	<input type="checkbox"/>	0	0
CAN1 TX interrupts	<input type="checkbox"/>	0	0
CAN1 RX0 interrupts	<input type="checkbox"/>	0	0
CAN1 RX1 interrupt	<input type="checkbox"/>	0	0
CAN1 SCE interrupt	<input type="checkbox"/>	0	0
TIM2 global interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input type="checkbox"/>	0	0
UART4 global interrupt	<input type="checkbox"/>	0	0
CAN2 TX interrupts	<input type="checkbox"/>	0	0
CAN2 RX0 interrupts	<input type="checkbox"/>	0	0
CAN2 RX1 interrupt	<input type="checkbox"/>	0	0
CAN2 SCE interrupt	<input type="checkbox"/>	0	0
DCMI global interrupt	<input type="checkbox"/>	0	0

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





- 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



Pin Configuration

ADC3 CAN1 CAN2 DCM1 ETH FMC UART4 USART1

Search Signals  
Search (Ctrl+F)  Show only Modified Pins

Pin Name	Signal on Pin	GPIO mode	GPIO Pull-up/Pull-d...	Maximum output sp...	Modified
PA6	DCMI_PIXCK	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PA9	DCMI_D0	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PA10	DCMI_D1	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PB8	DCMI_D6	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PB9	DCMI_D7	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PD3	DCMI_D5	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PE0	DCMI_D2	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PE1	DCMI_D3	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>
PE4	DCMI_D4	GPIO_MODE_AF_PP	No pull-up and no pu...	Low	<input type="checkbox"/>

PA6 Configuration :

GPIO mode: GPIO\_MODE\_AF\_PP

GPIO Pull-up/Pull-down: No pull-up and no pull-down

Maximum output speed: Low

Group By IP

Apply Ok Cancel

# STM32CubeMX: Power consumption calculator

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

MicroXplorer Untitled\*: STM32L100R(8-B)Tx

File Tools Windows Help

Pinout Configuration Power Consumption Calculator

**Microcontroller Selection**

Family: STM32L1  
SubFamily: STM32L100  
MCU: STM32L100R(8-B)Tx  
Datasheet: 024295\_Rev1  
Part Number: STM32L100R8

**Parameter Selection**

Ambient Temperature ... 25  
Vdd Power Supply (V): 3.6

**Battery Selection**

Battery: Li-SOCL2(D19000)  
Capacity: 19000.0 mAh  
Self Discharge: 0.08 %/month  
Nominal Voltage: 3.6 V  
Max Cont Current: 230.0 mA  
Max Pulse Current: 500.0 mA

**Information notes**

**Help**

Sequence

Load Save Delete

Sequence Table

Step	Mode	Range	Memory	Clock C...	Src Freq	CPU/Bus...	Peripher...	Add. Cu...	Step Cu...	Duration	DMIPS
1	LOWPOW...	NoRange	FLASH	MSI_131...	131.0 kHz	131.0 kHz	NULL	0 mA	48.0 µA	1 ms	0.16375
2	RUN	Range2-...	FLASH	HSEBYP_...	16.0 MHz	16.0 MHz	NULL	0 mA	3.9 mA	3 ms	16.8
3	LOWPOW...	NoRange	FLASH	MSI_131...	131.0 kHz	131.0 kHz	NULL	0 mA	48.0 µA	1 ms	0.16375

Step

Add Delete Duplic... Up Down

Sequence Chart

Consumption (mA)

Time (ms)

Sequence Average Current

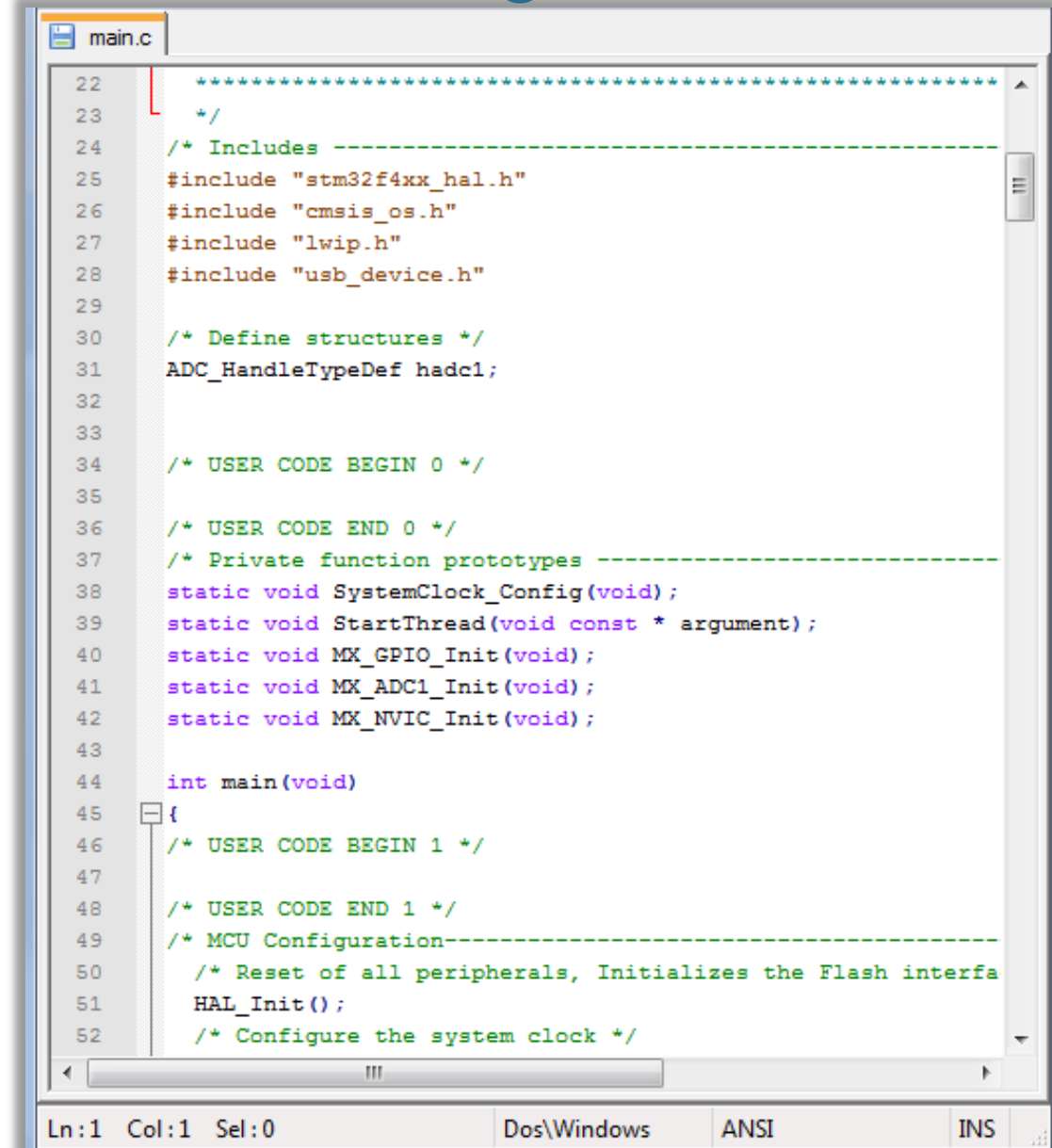
Results

Total Sequence Time 5.0 ms  
Average Consumption 2.359 mA  
Battery Life Estimation 11 months, 2 days & 14 hours  
Average DMIPS 10.15 DMIPS

# STM32CubeMX: Code generation

27

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

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

- 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

29

- 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)

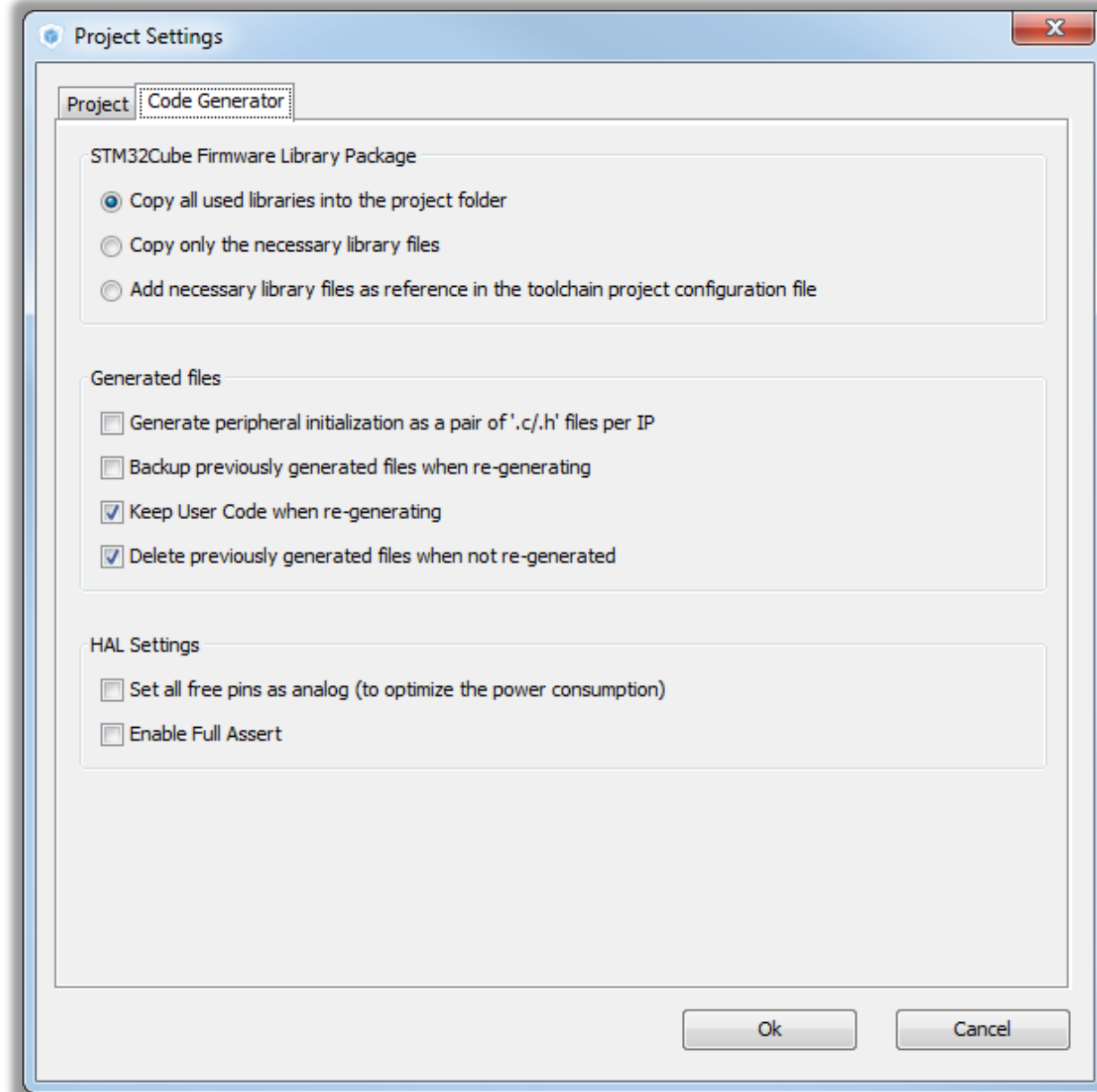
The screenshot shows the 'Project Settings' dialog box in STM32CubeMX. It has two tabs: 'Project' (selected) and 'Code Generator'. The 'Project' tab contains the following fields and options:

- Project Name:** An empty text input field.
- Project Location:** A text input field containing 'C:\Users\faubarqd\Documents\ACube' and a 'Browse' button to the right.
- Project Folder:** A text input field containing 'C:\Users\faubarqd\Documents\ACube'.
- Toolchain / IDE:** A dropdown menu with 'EWARM 6.70' selected.
- Mcu and Firmware Package:**
  - Mcu Reference:** A text input field containing 'STM32F437ZIYx'.
  - Firmware Package Name and Version:** A dropdown menu with 'STM32Cube FW\_F4 V1.1.0' selected. To its right is a checked checkbox labeled 'Use latest available version'.

# STM32CubeMX: Code Generator settings

30

- 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
- These 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
- Keep User Code when re-generating
- Delete previously generated files when not re-generated

# Keep User Code when re-generating

- Generated code contains USER CODE areas
- These 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

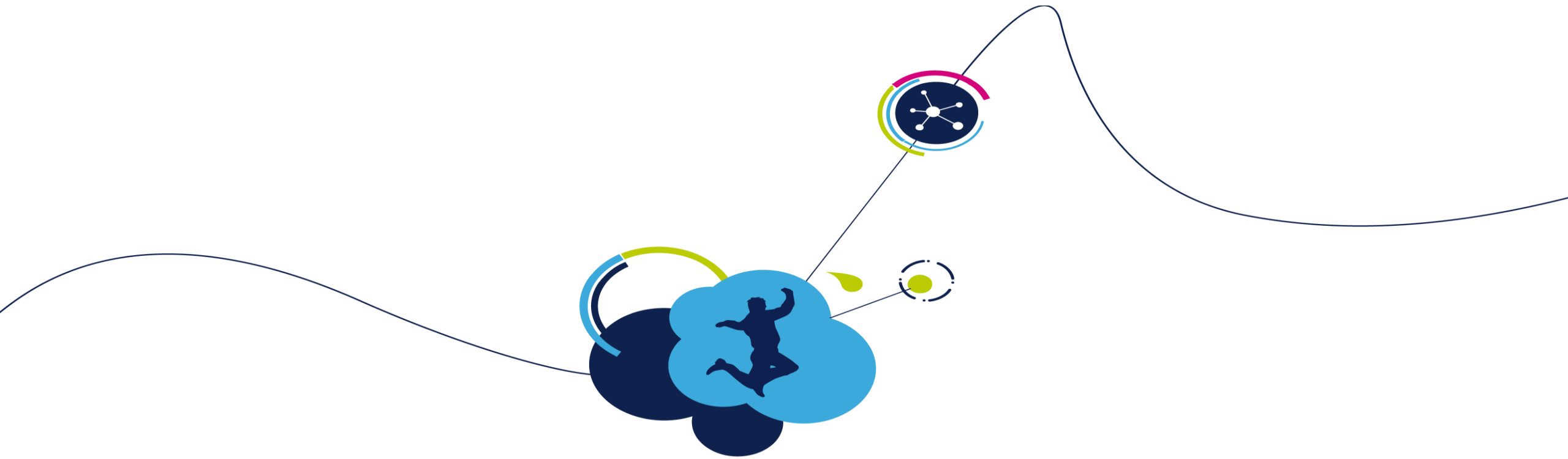
```
/* USER CODE BEGIN PFP */
```

```
/* USER CODE END PFP */
```

```
/* USER CODE BEGIN 0 */
```

```
/* USER CODE END 0 */
```





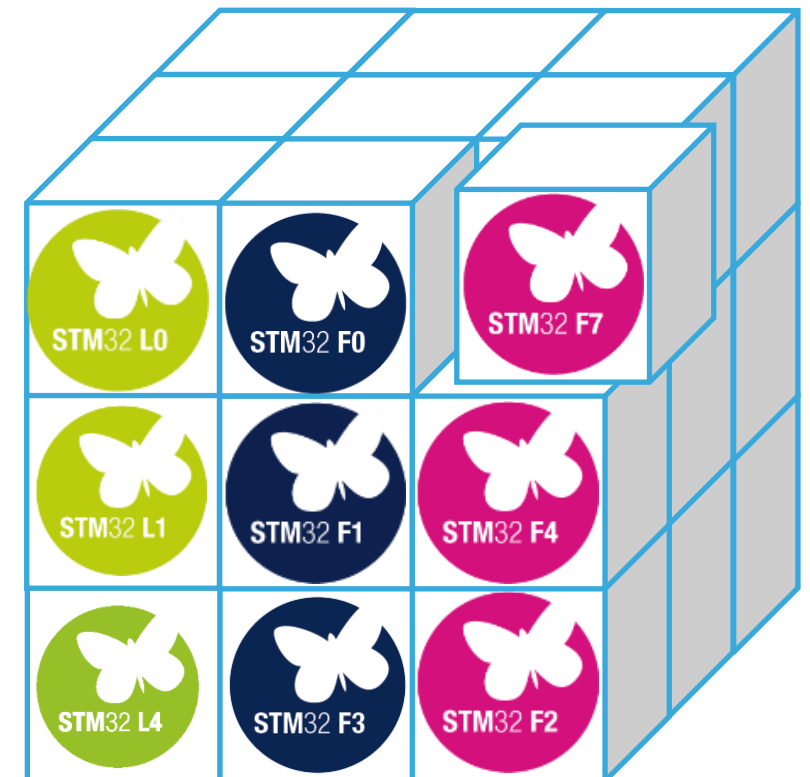
# STM32Cube HAL package



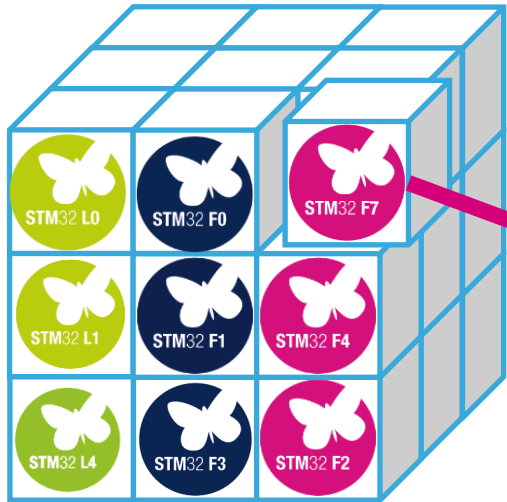
- STM32Cube™ Version 1:

- 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](http://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

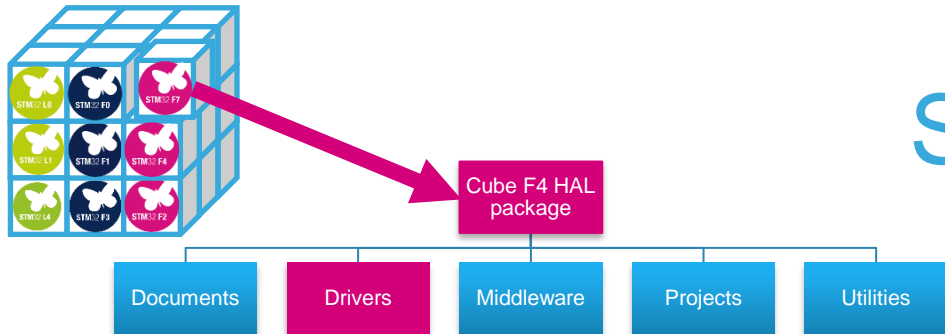


Getting started with  
CubeF3 document



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

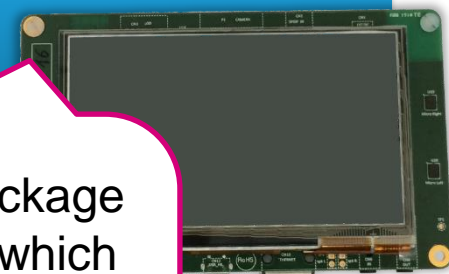
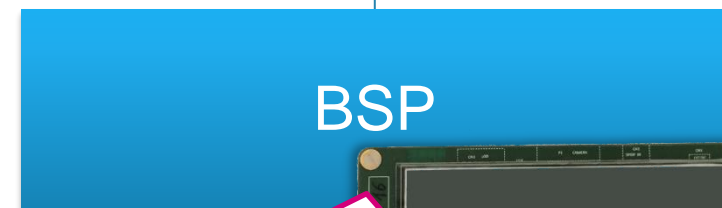
# STM32Cube FW Package Drivers



Register definitions for Core, startup files, ARM cortex libraries

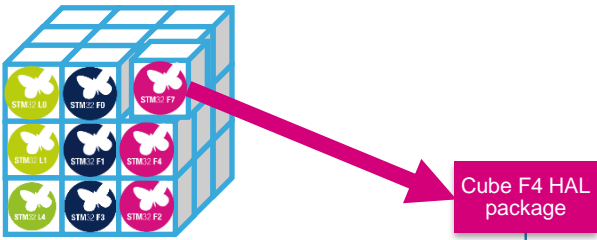


HAL Drivers for each periphery in STM32



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

# STM32Cube FW Package Middlewares



Advanced set of libraries

Developed/Owned by ST

Middlewares

Third Party libraries



ST

Third Party



STemWin



STM32\_Audio



FatFS



FreeRTOS

STM32\_USB\_Device\_Library



STM32\_USB\_Host\_Library



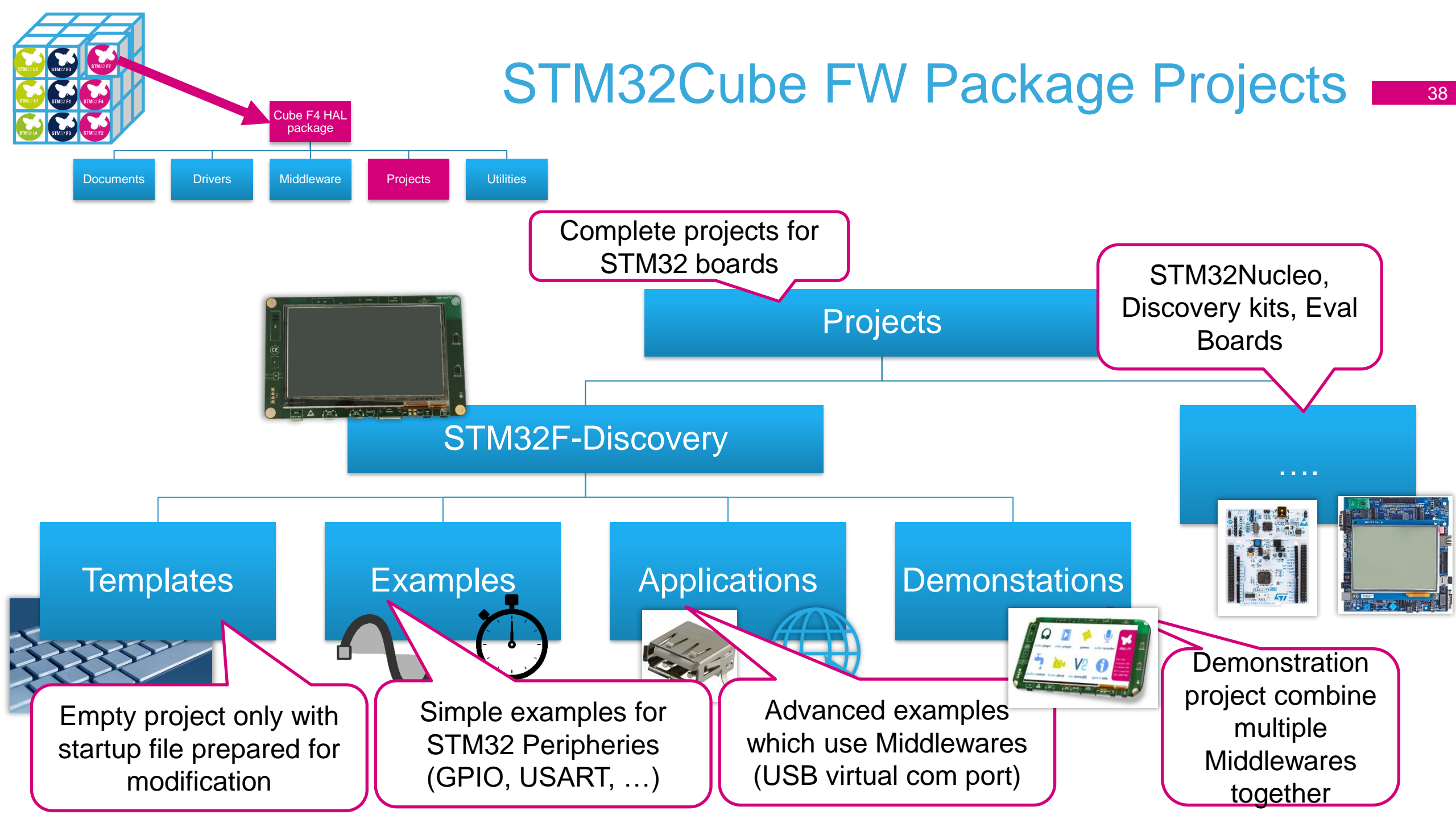
LibJPEG

LwIP



PolarSSL

# STM32Cube FW Package Projects



Cube F4 HAL package

- Documents
- Drivers
- Middleware
- Projects
- Utilities

Complete projects for STM32 boards

STM32Nucleo, Discovery kits, Eval Boards

Projects



STM32F-Discovery

Templates

Examples

Applications

Demonstrations

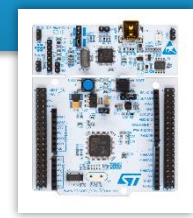
Empty project only with startup file prepared for modification

Simple examples for STM32 Peripherals (GPIO, USART, ...)

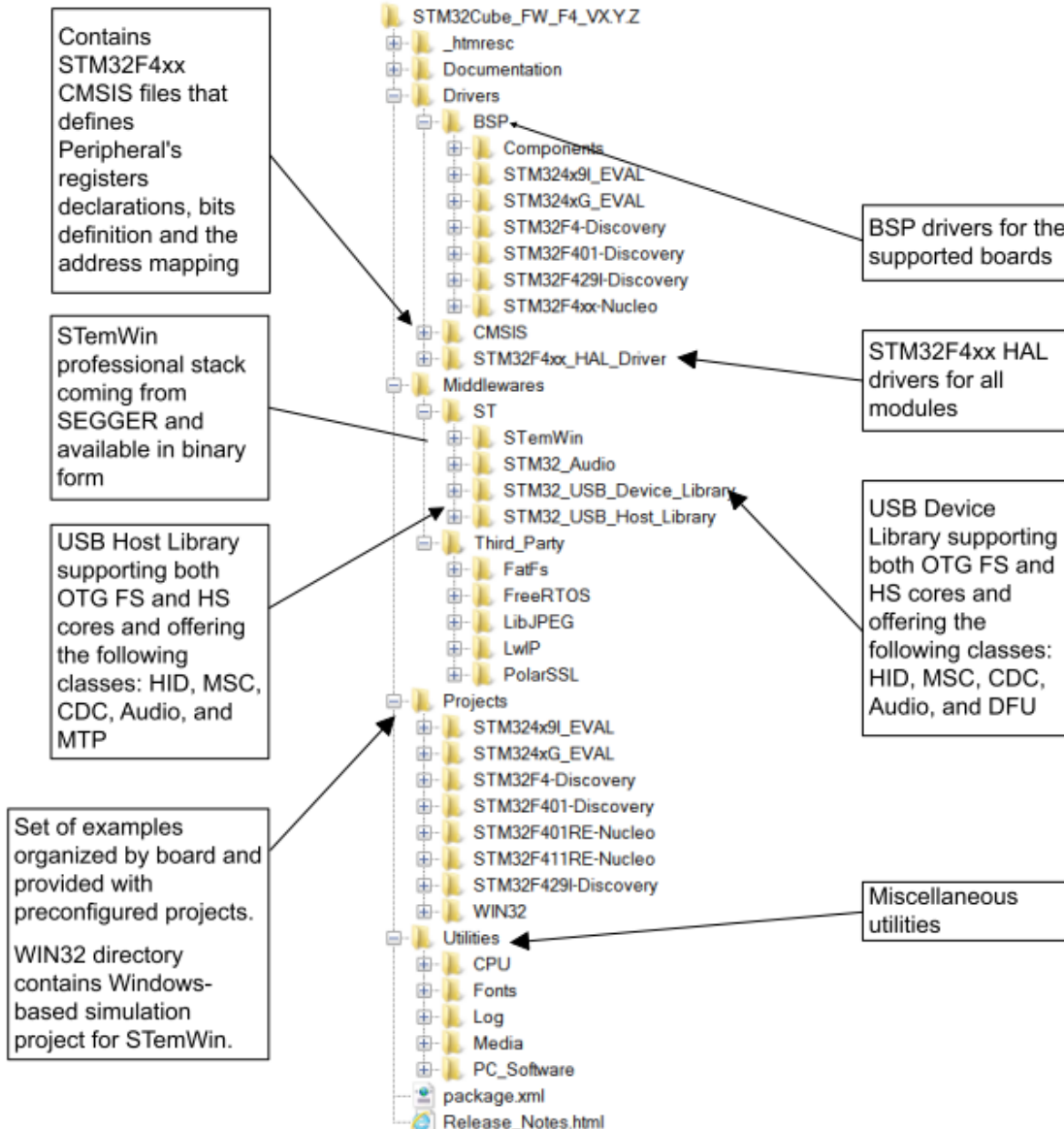
Advanced examples which use Middlewares (USB virtual com port)

Demonstration project combine multiple Middlewares together

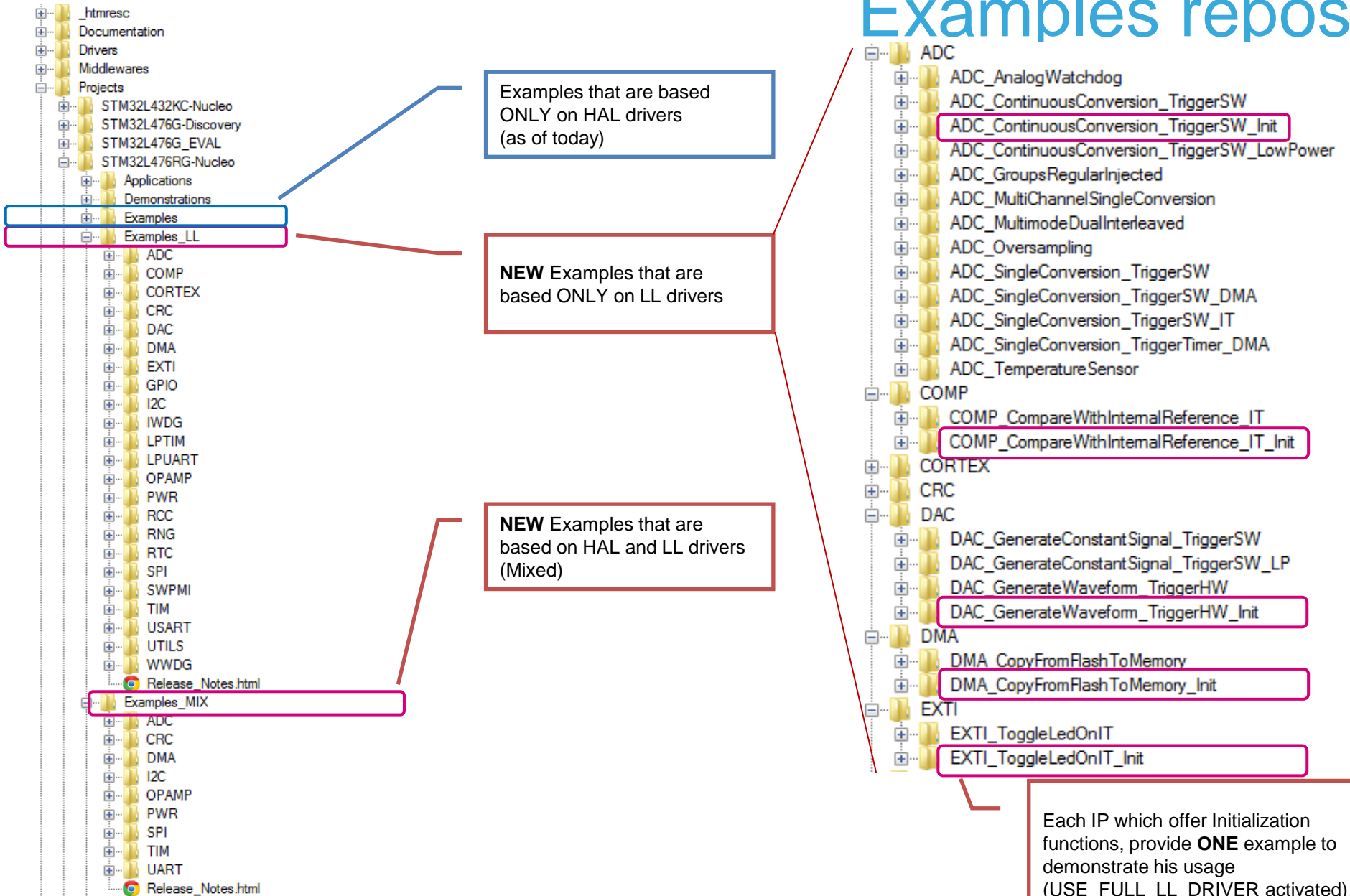
....



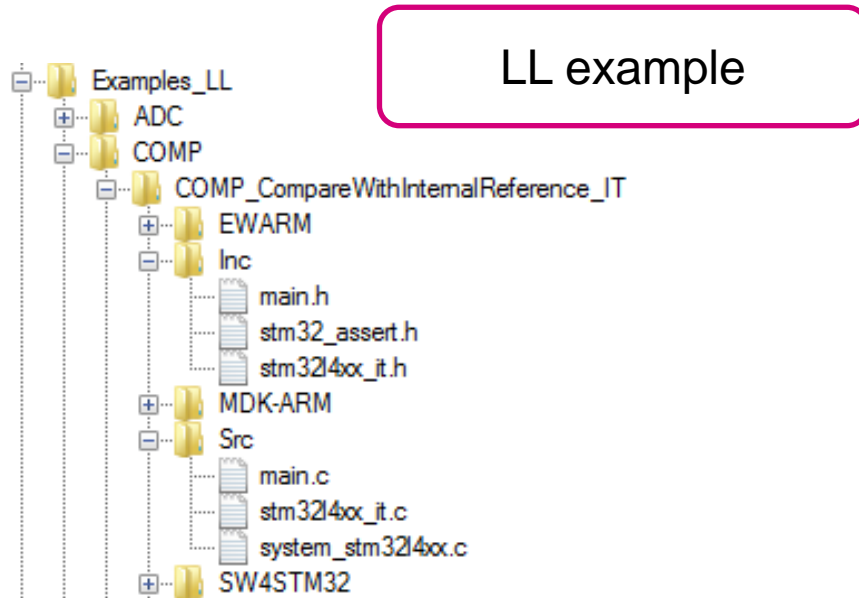
# STM32Cube FW Package Organization



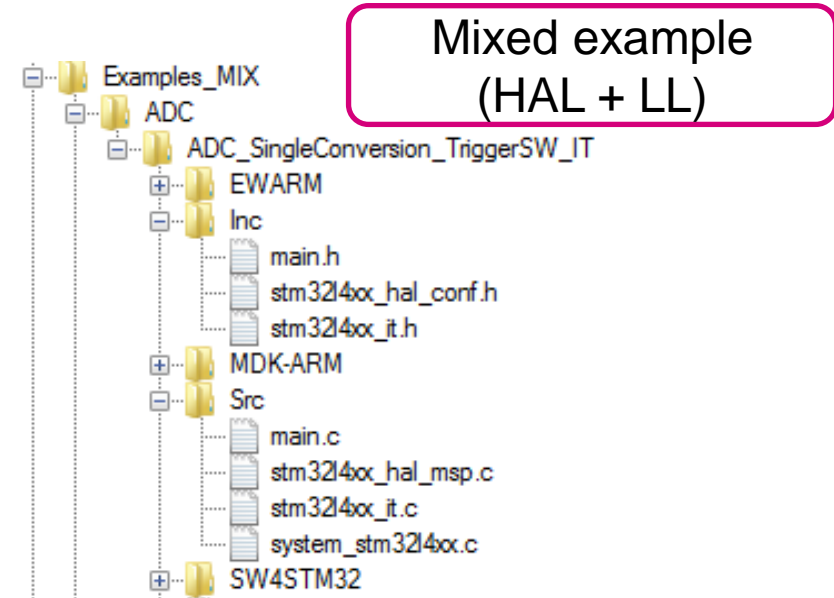
# Examples repository







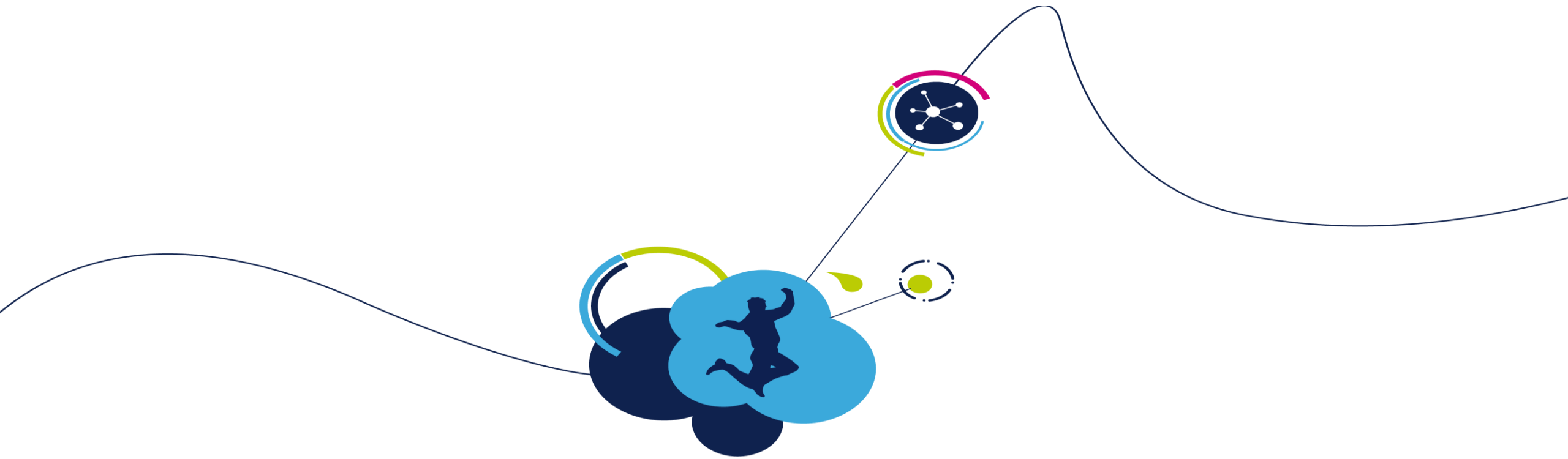
LL example



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.

- 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.



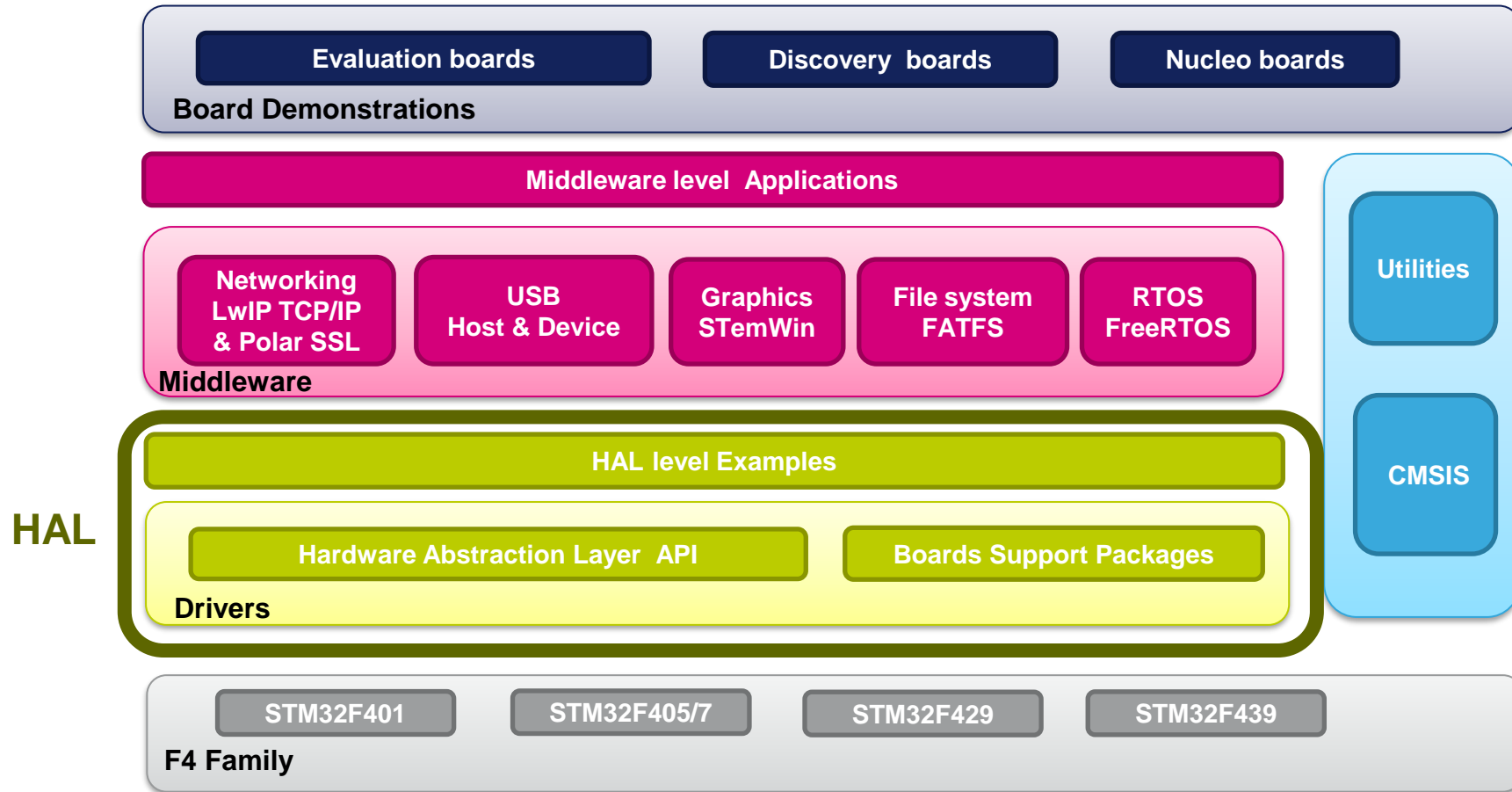
# STM32Cube Hardware Abstraction Layer (HAL)



# HAL general concepts

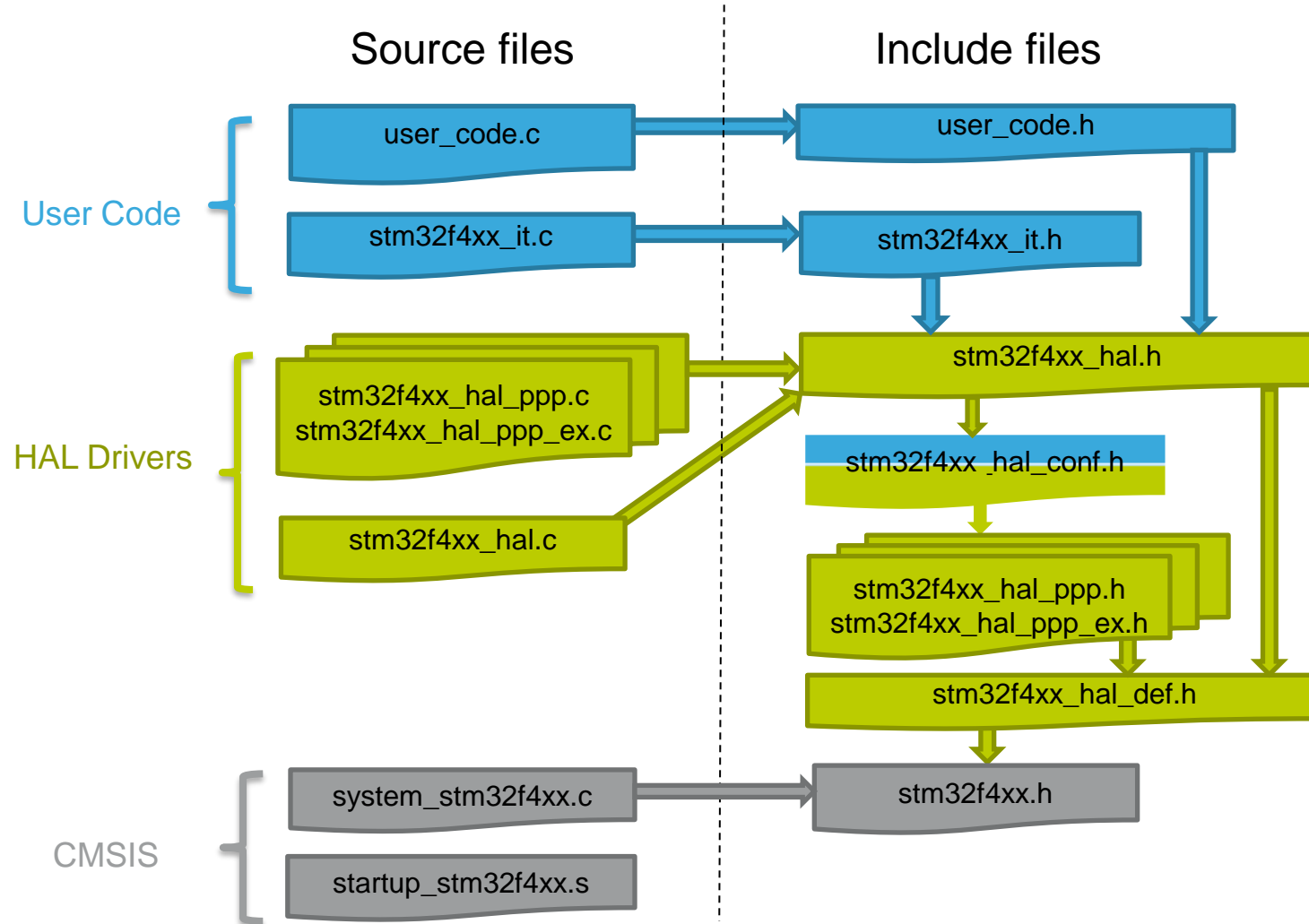
# HAL general concepts

## The HAL in STM32Cube FW package



# HAL general concepts

## HAL based project organization



# 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

48

- 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`

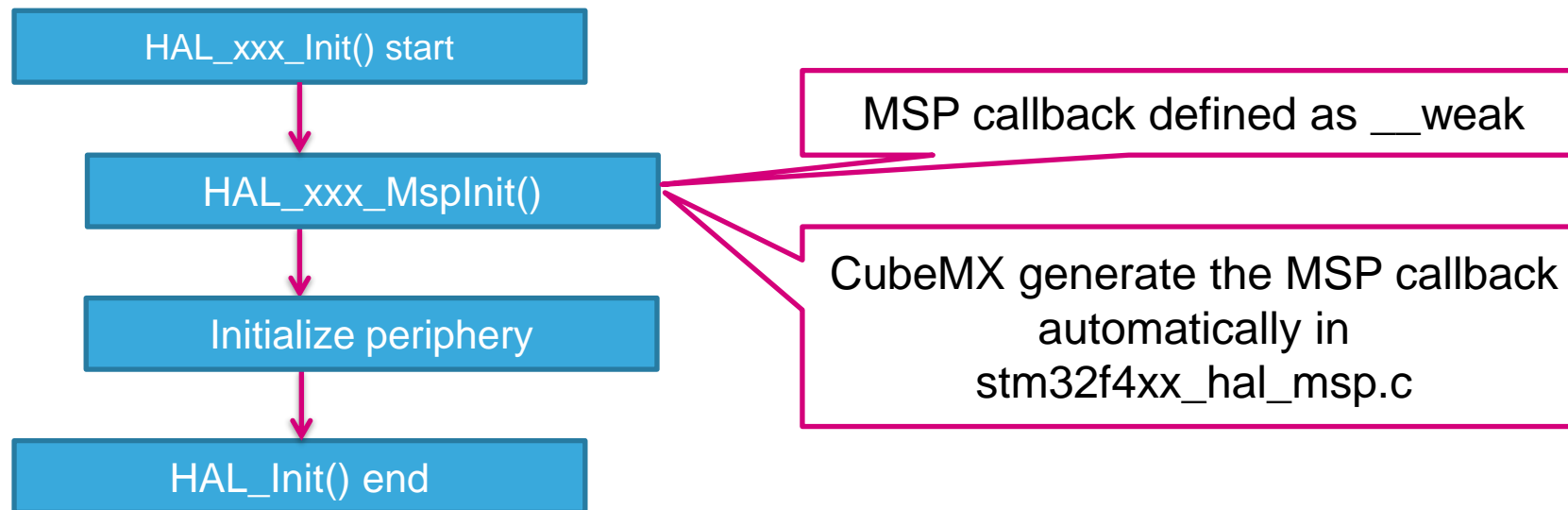
```
/**
 * @brief Tx Transfer completed callbacks
 * @param hspi: pointer to a SPI_HandleTypeDef structure that contains
 *             the configuration information for SPI module.
 * @retval None
 */
__weak void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi)
{
    /* NOTE : This function Should not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



# HAL general concepts

## Init functions

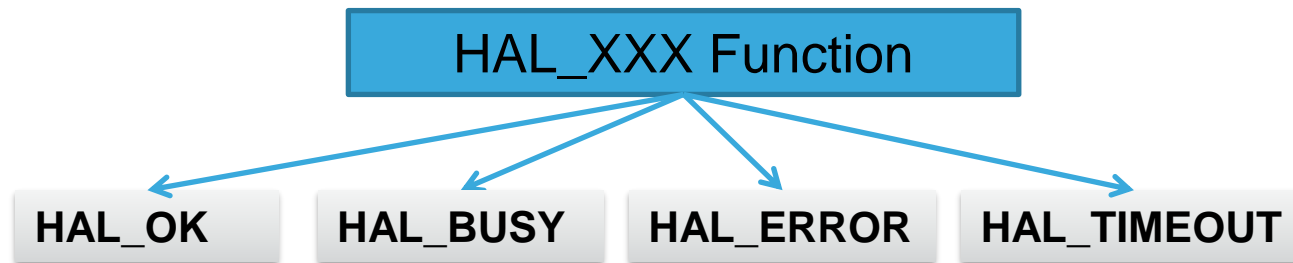
- HAL\_XXX\_Init() functions
- Inside init function are written data from input parameters/structure into registers
- Before the register write is processed HAL\_XXX\_MspInit callback is called

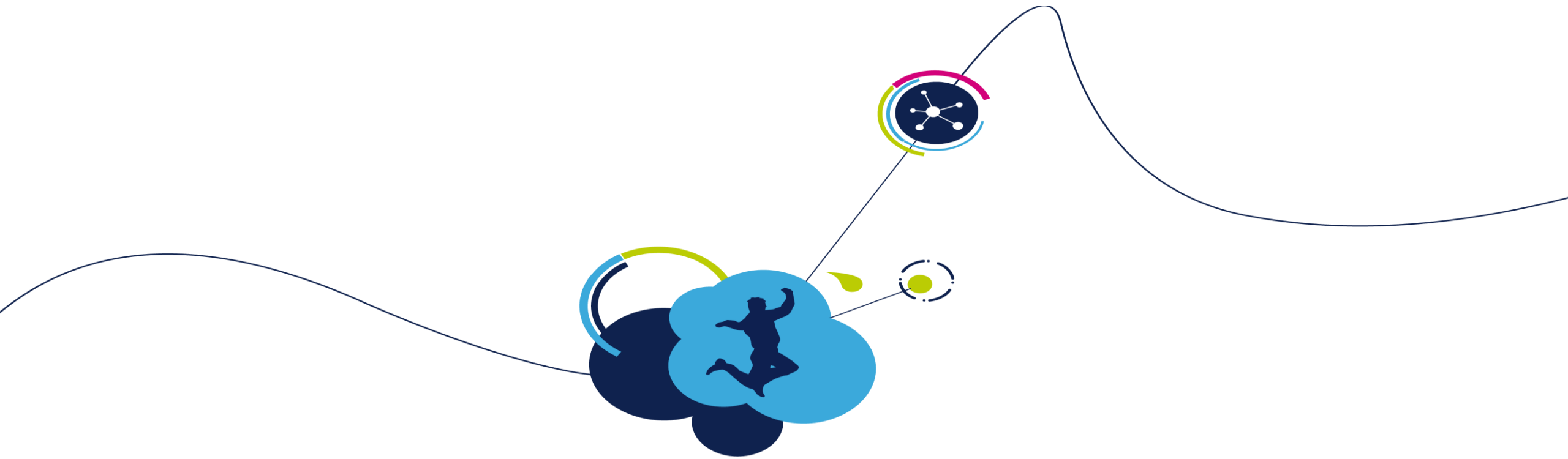


# HAL general concepts

## HAL API returns parameters

- HAL API can return a value of enumerated type HAL\_StatusTypeDef:
  - HAL\_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



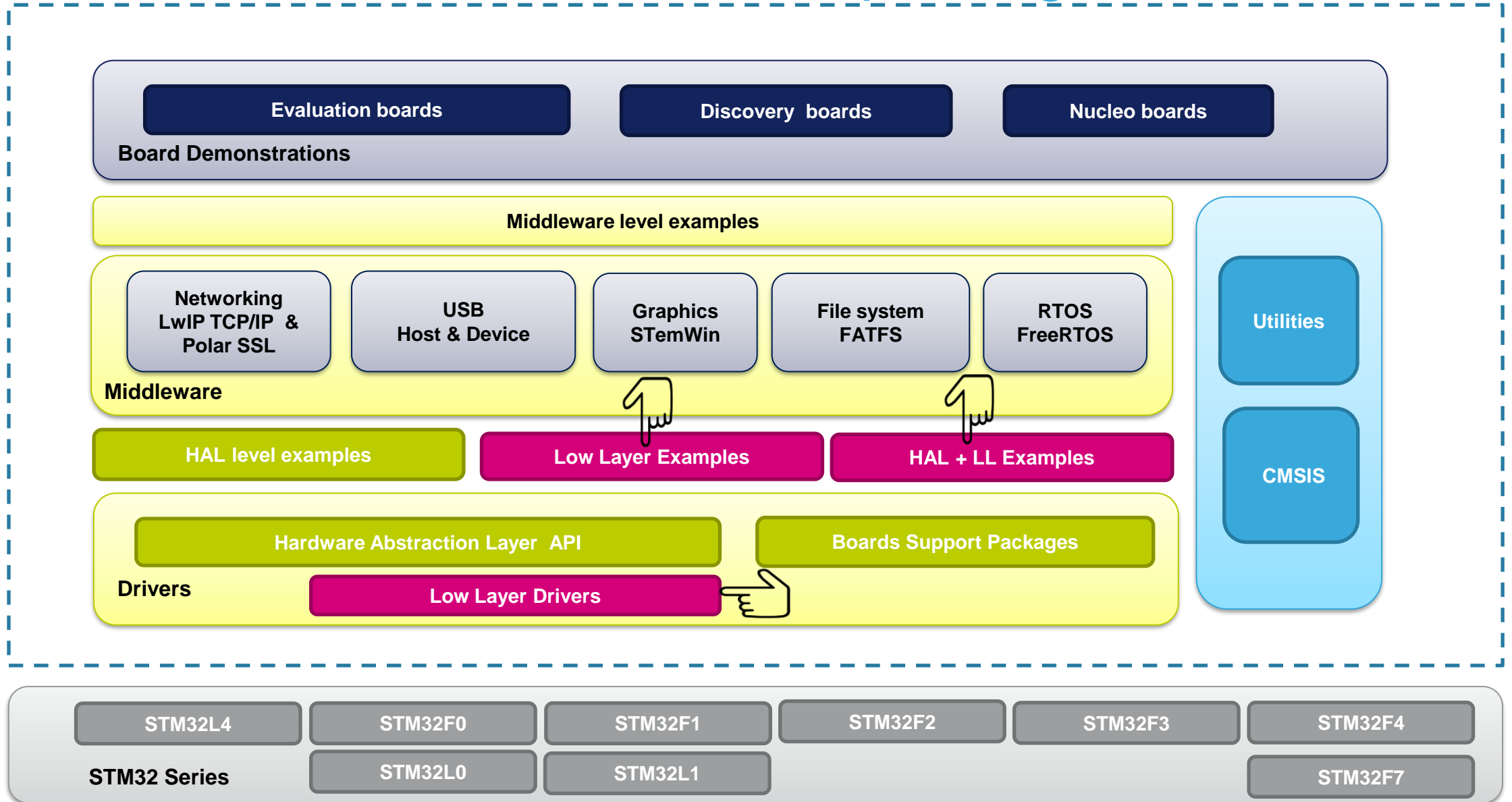


Low Layer drives

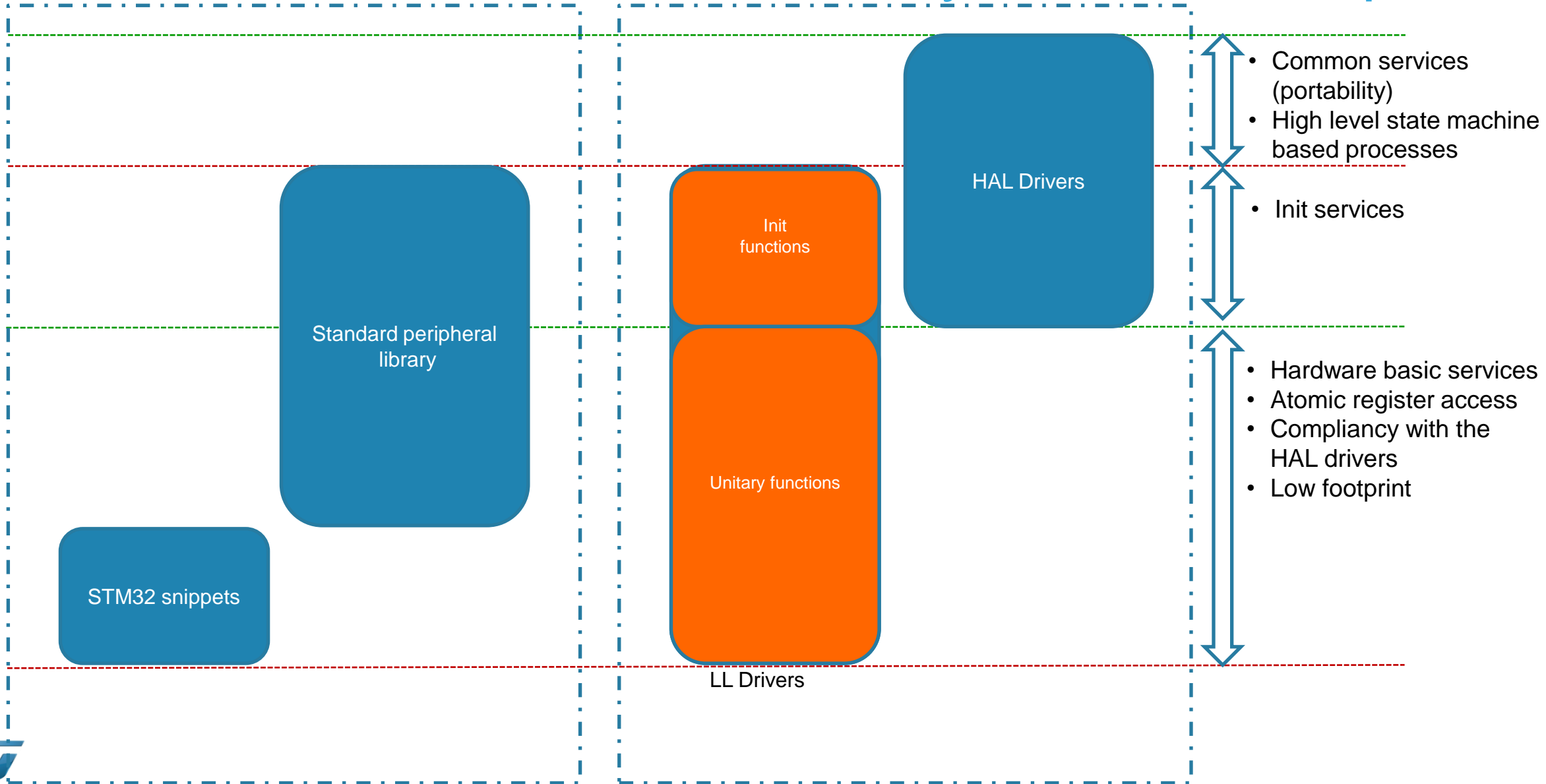
- **February 2014: 1<sup>st</sup> Release of STM32CubeV1 solution, based on F4 Series**
- After this release, we received some **concerns about the HAL**, mainly:
  - Big code called at peripheral initialization
  - Need to go-through full peripheral initialization even if slight modification is needed 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**

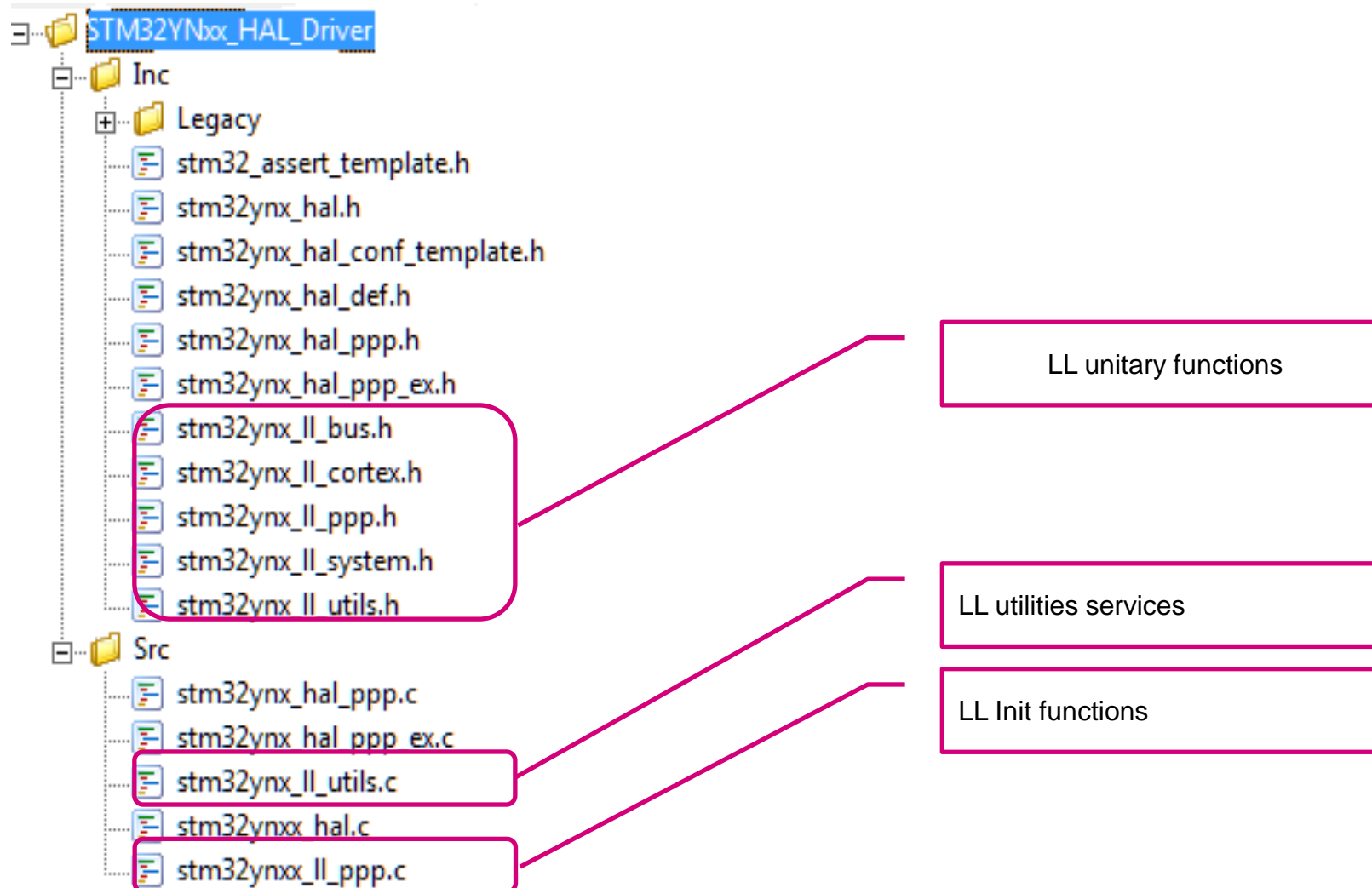
# STM32Cube FW package block view



# Low Layer Drivers scope



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



- 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**

```
61  /**
62   * @brief Main program
63   * @param None
64   * @retval None
65   */
66  int main(void)
67  {
68      /* Configure the system clock to 80 MHz */
69      SystemClock_Config();
70
71      /* -2- Configure IO in output mode */
72      Configure_GPIO();
73
74      /* Toggle IO in an infinite loop */
75      while (1)
76      {
77          LL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
78
79          /* Insert delay 250 ms */
80          LL_mDelay(250);
81      }
82  }
```

```
89  __STATIC_INLINE void Configure_GPIO(void)
90  {
91      /* Enable the LED2 Clock */
92      LL_AHB2_GRP1_EnableClock(LL_AHB2_GRP1_PERIPH_GPIOA);
93
94      /* Configure IO in output push-pull mode to drive external LED2 */
95      LL_GPIO_SetPinMode(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_MODE_OUTPUT);
96      LL_GPIO_SetPinOutputType(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_OUTPUT_PUSHPULL);
97      LL_GPIO_SetPinSpeed(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_SPEED_LOW);
98      LL_GPIO_SetPinPull(LED2_GPIO_PORT, LED2_PIN, LL_GPIO_PULL_NO);
99  }
```



Thank you!

