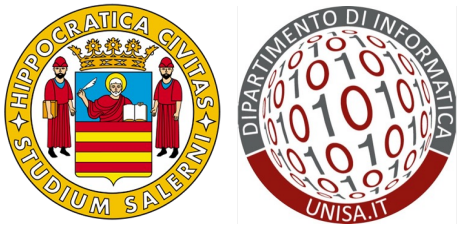
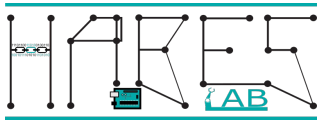




Hardware Assisted and
Blockchain Empowered Security Lab

LAB: Introduction to the programming of STM32 boards

Introduction

The world of embedded systems is complex but exciting, and **STM32** microcontrollers play a key role in today's electronics.

STM32CubeIDE is an important tool for developers who want to make the most of STM32 boards.

We'll explore STM32CubeIDE, a **development environment** that offers many powerful features while staying simple to use, making it great for both beginners and experienced developers.

What is STM32?

STM32 is a family of 32-bit microcontroller chips made by STMicroelectronics. These STM32 chips are available in **different series**, designed for many types of applications.

Key families include STM32F0, STM32F1, STM32F4, STM32H7, and STM32L0.

- STM32F series: General-purpose microcontrollers offering a good balance of performance and efficiency.
- STM32H series: High-performance models for demanding applications.
- STM32L series: Low-power devices designed for energy-efficient applications.

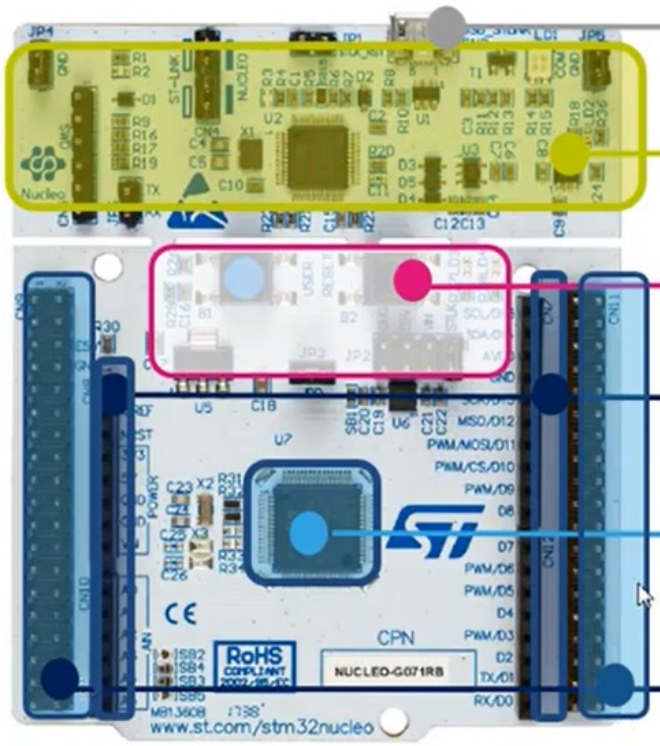
To easily prototype with STM32 MCUs, **Nucleo** boards are available. These development boards support various STM32 chips, making it simple to test and develop IoT applications such as wearables, industrial IoT systems, and smart home devices.

STM32 Cube IDE

The STM32Cube ecosystem is a comprehensive set of tools and software designed to simplify the development process for STM32 microcontrollers (MCUs).

- **STM32CubeMX**: This tool allows users to configure microcontroller peripherals and automatically generate initialization code.
- **STM32CubeIDE**: This integrated development environment combines code editing, compiling, and debugging into one tool. It simplifies writing and testing code, making the entire development workflow smoother.
- **STM32Cube Firmware Libraries**: These libraries provide pre-built middleware solutions such as USB, Real-Time Operating Systems (RTOS), and file systems like FATFS. They save time by offering reusable components for various functionalities.

Nucleo Board



Flexible board power supply :
through USB or external source

Integrated ST-Link/V2-1:
mass storage device flash programming

2 push buttons, 2 color Leds

Arduino extension connectors :
easy access to add-ons

One STM32 MCU flavor with 64 pins

Morpho extension headers :
direct access to all MCU I/Os

Nucleo Board

- Connect USB ST-LINK (**CN2**) to your PC
  - ST-LINK driver may be installed if this is the first time the board is plugged in.
- LD1 should be ON and solid RED (indicating board power available and ST-Link is functional)

Nucleo Board

NUCLEO-F401RE

Affordable and flexible platform to ease prototyping using a STM32F401RET6 microcontroller.

Overview

The STM32 Nucleo board provides an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller line, choosing from the various combinations of performance, power consumption and features

Table of Contents

1. Overview
2. Microcontroller features
3. Board features

https://os.mbed.com/platforms/ST-Nucleo-F401RE/

STM32 Cube IDE

STM32CubeIDE is an Integrated Development Environment (IDE) designed specifically for STM32 microcontrollers.

STM32CubeIDE combines various tools needed for programming STM32 microcontrollers, making it easier to manage projects and code.
Tools for debugging your code directly within the platform.

Cross-Platform Support, Windows, Linux, or macOS, STM32CubeIDE works seamlessly across all these operating systems, giving you flexibility in your development environment.

STM32 Cube IDE

https://www.st.com/en/development-tools/stm32cubeide.html

Creating a Project in STM32CubeIDE

1. Launch STM32CubeIDE and select File > New > STM32 Project.

2. The Target Selection window appears. Here, you can search for your STM32 microcontroller or STM32 board by part number or name. Once selected, click Next.

3. Configure your project settings, including project name and location. Configure GPIO PB0 as output.

4. Ctr+S then generate the project/Code (click 'Yes') and write code following code.

5. Attached STM32 Board the Build project and click 'Run'.

Understanding project structure

Upon creating a new project, STM32CubeIDE presents a structured view of your project's files and resources. Familiarizing yourself with this structure is crucial for efficient development.

1. **Inc** and **Src** folders contain your project's header and source files, respectively.
2. The Drivers folder houses the **HAL** (Hardware Abstraction Layer) and **LL** (Low Layer) drivers, facilitating hardware abstraction for easier coding.
3. .**cproject** and .project files are Eclipse-based configuration files used by STM32CubeIDE.
4. The MX Project (.**ioc**) file is where STM32CubeMX stores the configuration of your STM32 microcontroller or microprocessor.

Adding user code

To add your code, navigate to the **Src/main.c** file.

STM32CubeIDE provides markers such as */* USER CODE BEGIN */* and */* USER CODE END */* to indicate safe areas where you can write your custom code without it being overwritten by the STM32CubeMX code generator.

Build project

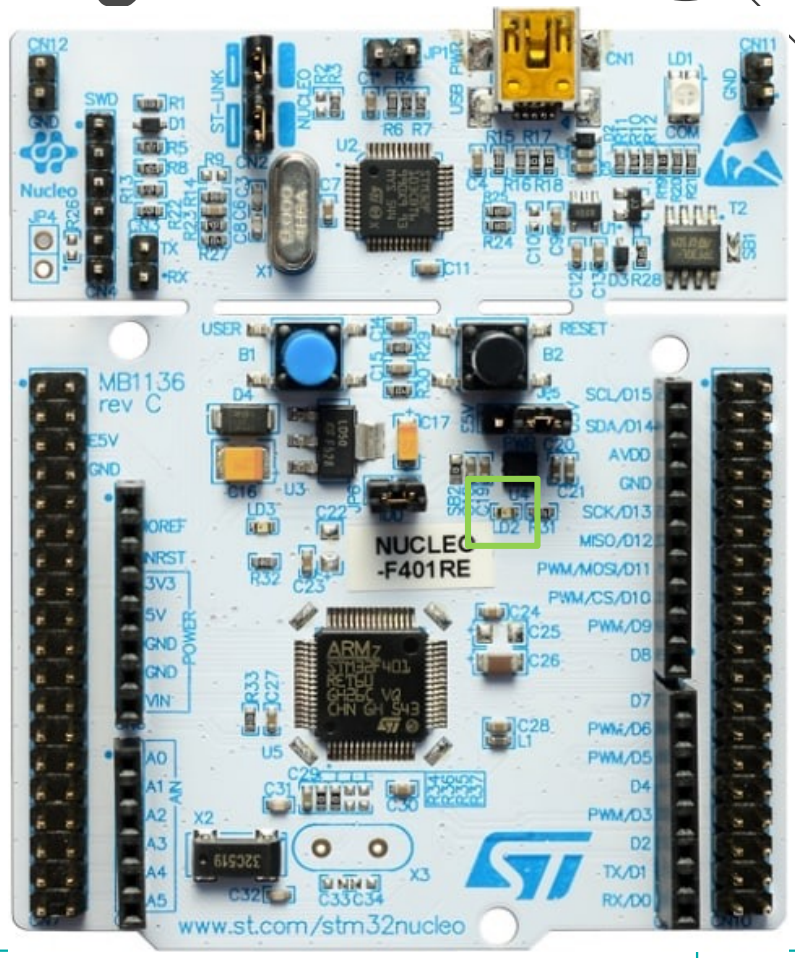
To build your project, click on the Build button (represented by a hammer icon) in the toolbar.

STM32CubeIDE compiles your project, and any errors or warnings will be displayed in the Problems view. Resolve any issues before proceeding.

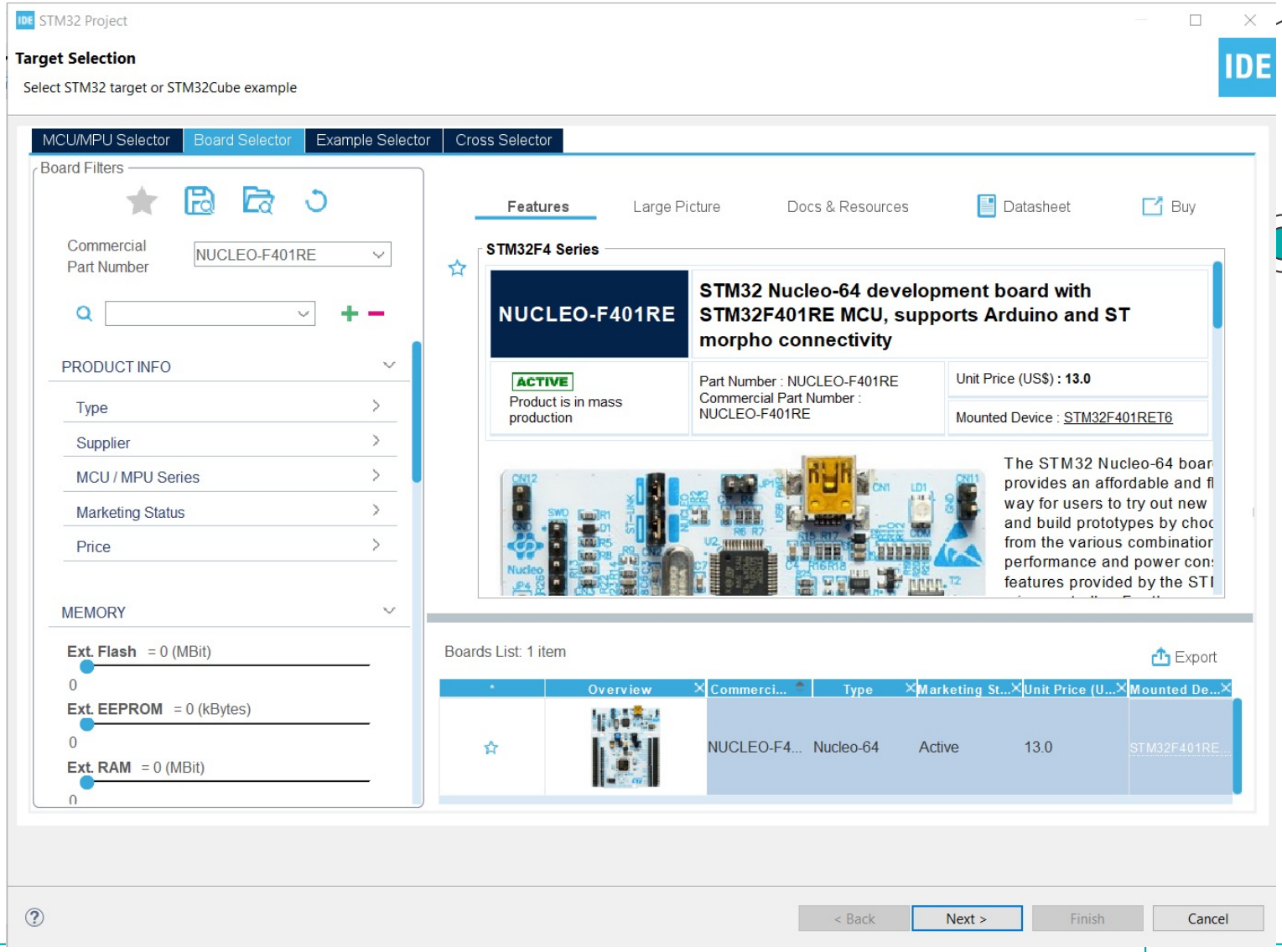
Once the build is successful, you are ready to flash the program to your STM32 board.

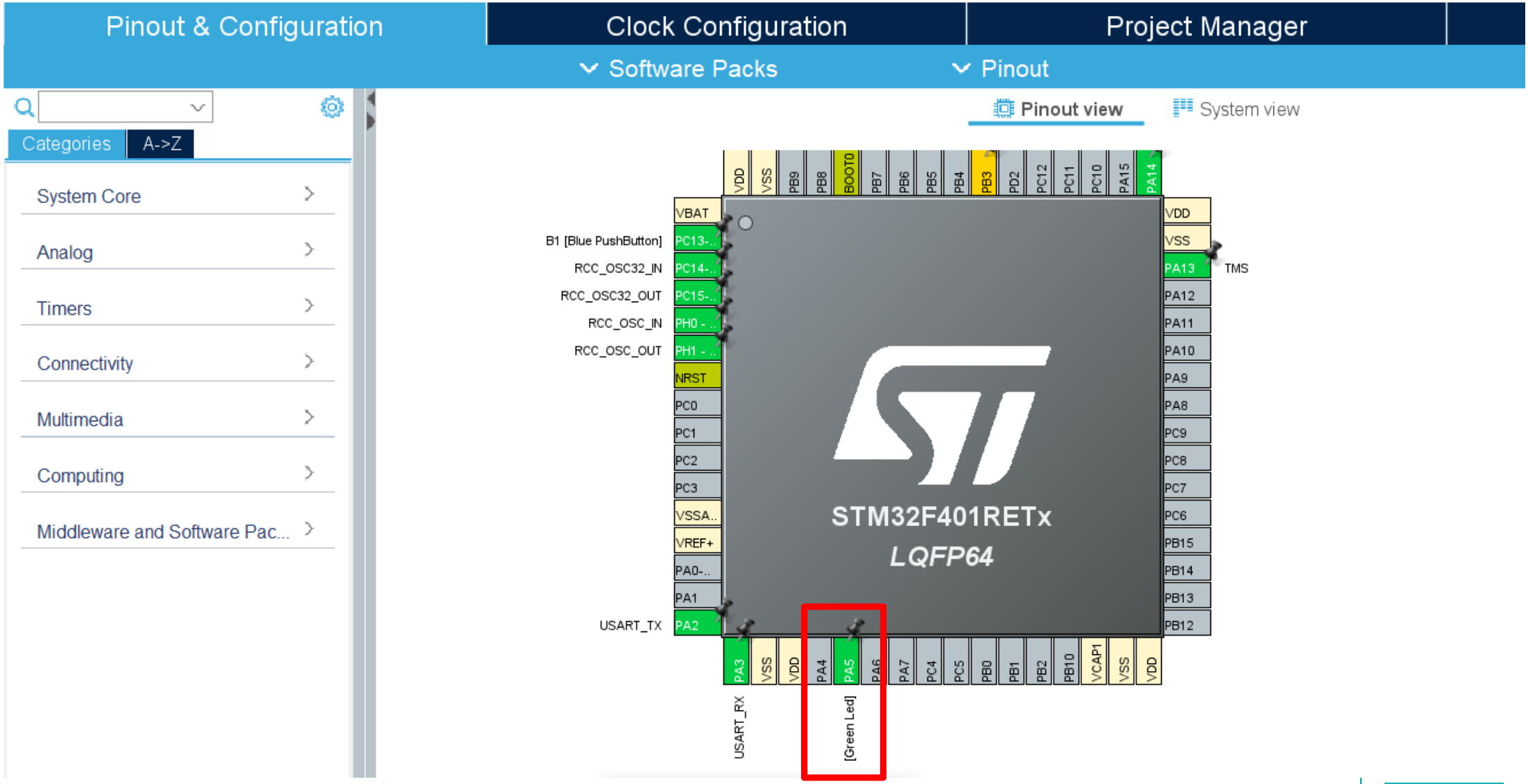
Led toggling using HAL library

Toggle green led -> LD2

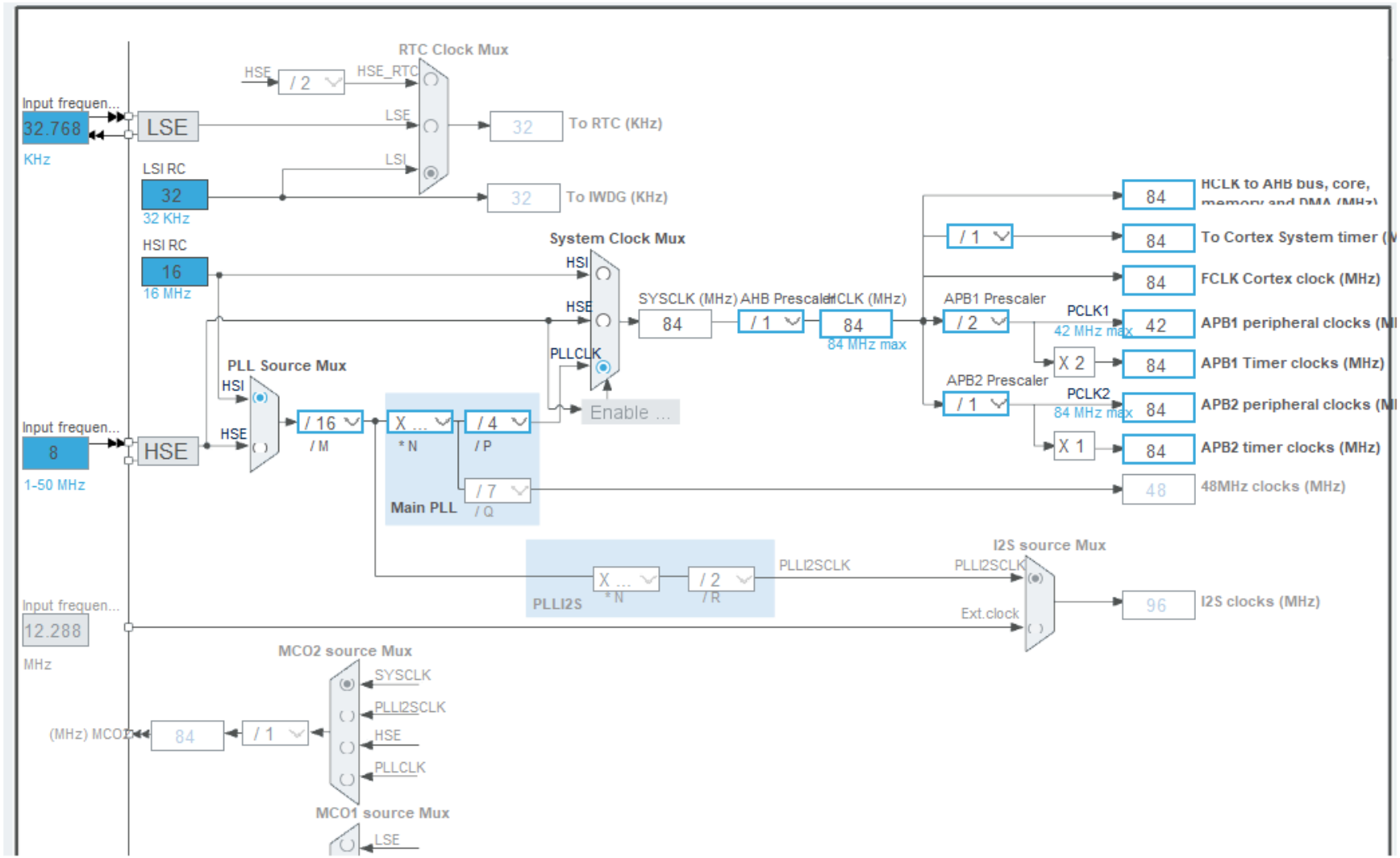


Create project

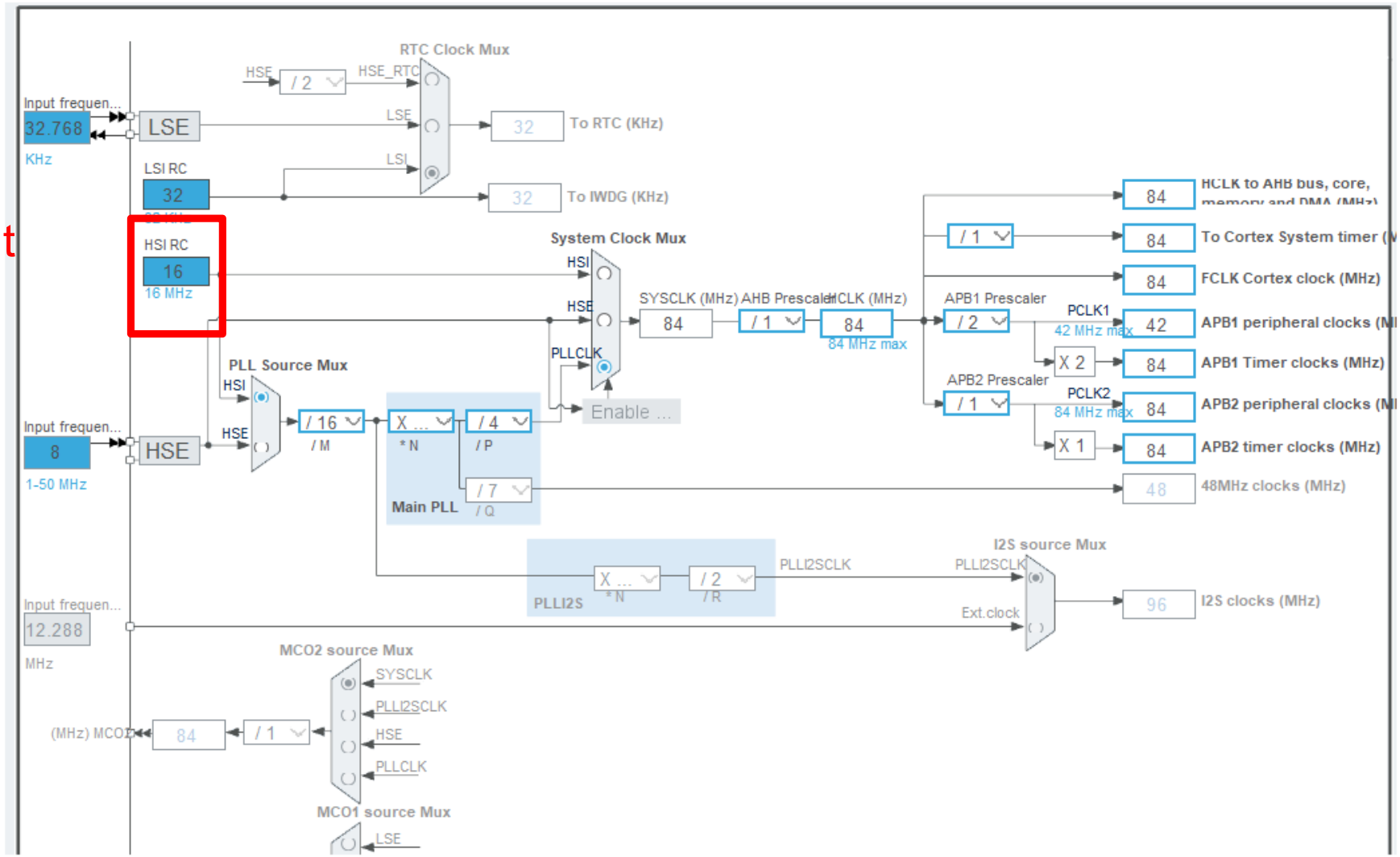



Pinout & Configuration
Clock Configuration
Project Manager
Software Packs
Pinout
Pinout view
System view
Categories
A->Z
System Core
Analog
Timers
Connectivity
Multimedia
Computing
Middleware and Software Pac...
STM32F401RETx
LQFP64
VDD
VSS
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PD2
PC12
PC11
PC10
PA15
PA14
VBAT
B1 [Blue PushButton]
PC13-..
RCC_OSC32_IN
PC14-..
RCC_OSC32_OUT
PC15-..
RCC_OSC_IN
PH0 - ..
RCC_OSC_OUT
PH1 - ..
NRST
PC0
PC1
PC2
PC3
VSSA..
VREF+
PA0-..
PA1
USART_TX
PA2
PA13
TMS
PA12
PA11
PA10
PA9
PA8
PC9
PC8
PC7
PC6
PB15
PB14
PB13
PB12
PA3
VSS
VDD
PA4
PA5
PA6
PA7
PC4
PC5
PB0
PB1
PB2
PB10
VCAP1
VSS
VDD
USART_RX
[Green Led]

ion | Clock Configuration | Project Manager



Default clock



Pinout & Configuration | Clock Configuration | Project Manager

Project

Code Generator

Advanced Settings

Project Settings

Project Name	t3
Project Location	C:\Users\Franco\STM32CubeIDE\workspace_1.15.1
Application Structure	Advanced ☐ Do not generate the main()
Toolchain Folder Location	C:\Users\Franco\STM32CubeIDE\workspace_1.15.1\t3\
Toolchain / IDE	STM32CubeIDE ☑ Generate Under Root

Linker Settings

Minimum Heap Size	0x200
Minimum Stack Size	0x400

Thread-safe Settings

Cortex-M4NS

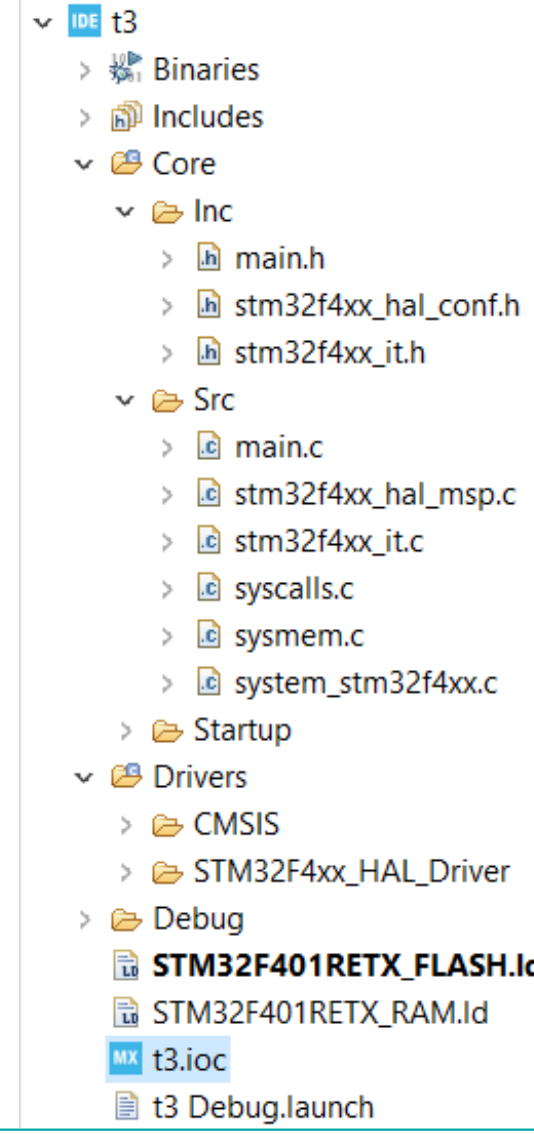
☐ Enable multi-threaded support

Thread-safe Locking Strategy	Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference	STM32F401RETx
Firmware Package Name and Version	STM32Cube FW_F4 V1.28.1 ☑ Use latest available version

Project structure



HAL Library

The **HAL** (Hardware Abstraction Layer) library is an official set of libraries provided by STMicroelectronics for STM32 microcontrollers. It simplifies the process of configuring and using the hardware peripherals by abstracting away the low-level register manipulations.

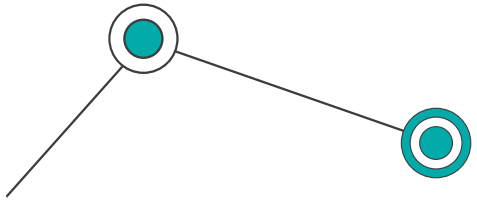
- Hardware Abstraction: It abstracts the hardware details and makes your code more portable across different STM32 microcontrollers.
- Ease of Use: It simplifies the development process by providing higher-level APIs to configure and control the hardware.
- Modularity: HAL is divided into modules, each representing a peripheral or subsystem (GPIO, SPI, I2C, UART, etc.).
- Compatibility: It supports all STM32 series (F0, F1, F2, F3, F4, F7, L0, L1, L4, H7, etc.), and once a project is written using HAL, migrating to a different STM32 family is easier.

Key HAL Libraries/Modules

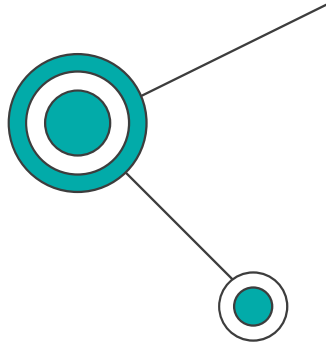
HAL is structured into different modules that handle various peripherals and functions:

Ex:

- GPIO (General Purpose Input/Output) Functions for configuring and controlling GPIO pins as inputs, outputs, or alternate function pins.Example: **HAL_GPIO_WritePin(), HAL_GPIO_ReadPin().**
- UART (Universal Asynchronous Receiver/Transmitter)Functions for sending and receiving data over serial communication.Example: **HAL_UART_Transmit(), HAL_UART_Receive().**
- SPI (Serial Peripheral Interface)Functions to configure and use the SPI peripheral for communication with other devices.Example: **HAL_SPI_Transmit(), HAL_SPI_Receive().**



Main



```c
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();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
  }
```

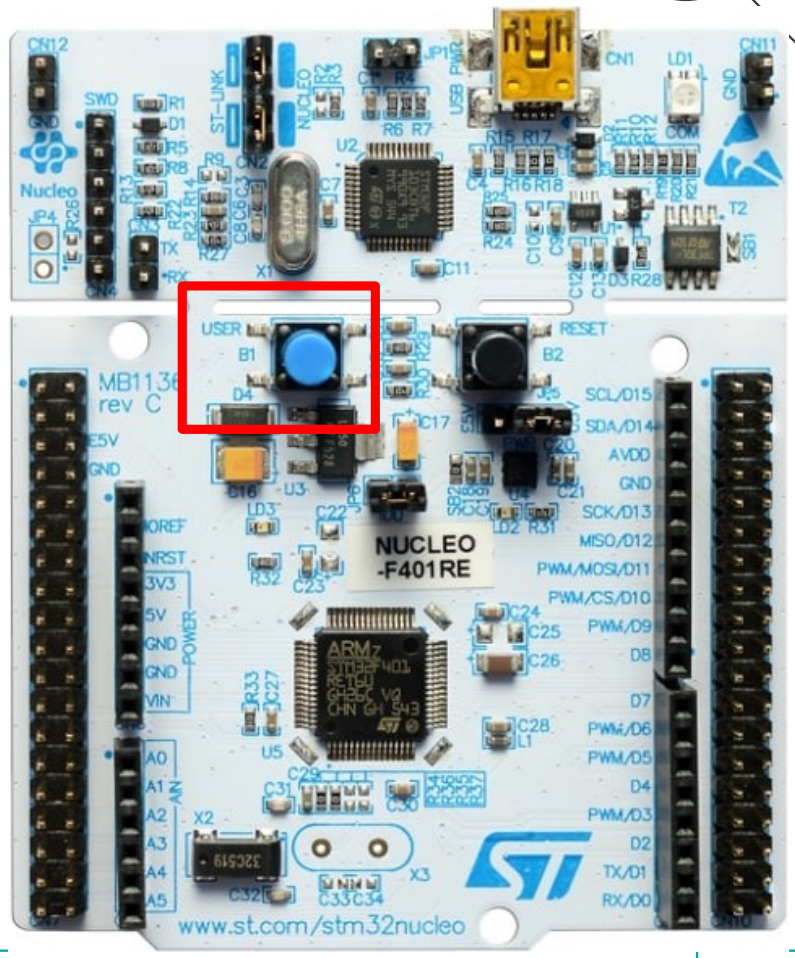
Main

```
96   /* Infinite loop */
97   /* USER CODE BEGIN WHILE */
98   while (1)
99   {
100      HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5);
101      HAL_Delay (500);
102    /* USER CODE END WHILE */
103
104    /* USER CODE BEGIN 3 */
105  }
106  /* USER CODE END 3 */
```

Port PIN

Edit: toggle button

Blue button-> User button



Edit: toggle button

Configuration

Group By Peripherals

SYS | USART | NVIC | GPIO | Single Mapped Signals | RCC

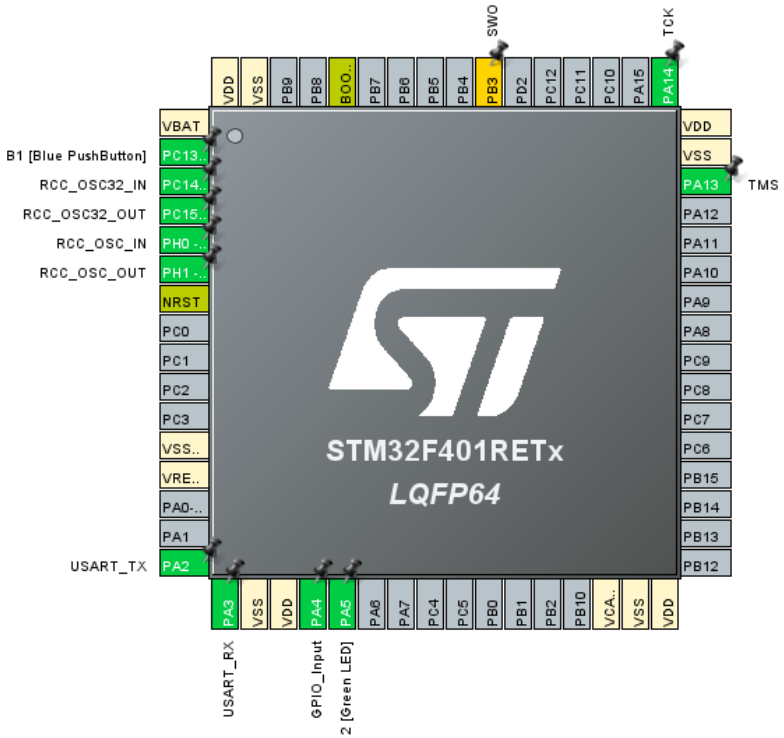
Search Signals

Search (Ctrl+F) ☐ Show only Modified Pins

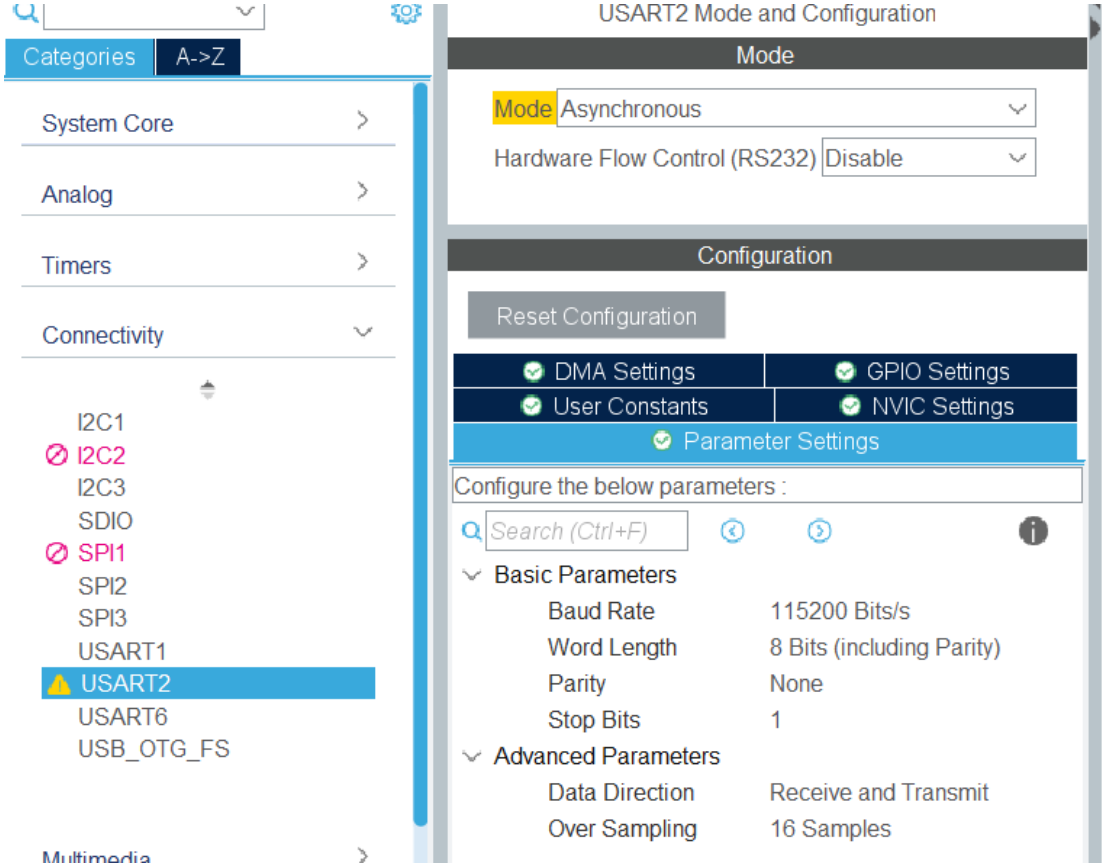
Pin N...	Signal o...	GPIO out...	GPIO m...	GPIO Pu...	Maximu...	User Lab...	Modified
PA4	n/a	n/a	Input mo...	No pull-u...	n/a		☐
PA5	n/a	Low	Output P...	No pull-u...	Low	LD2 [Gr...	☑
PC13-A...	n/a	n/a	External ...	No pull-u...	n/a	B1 [Blue ...	☑

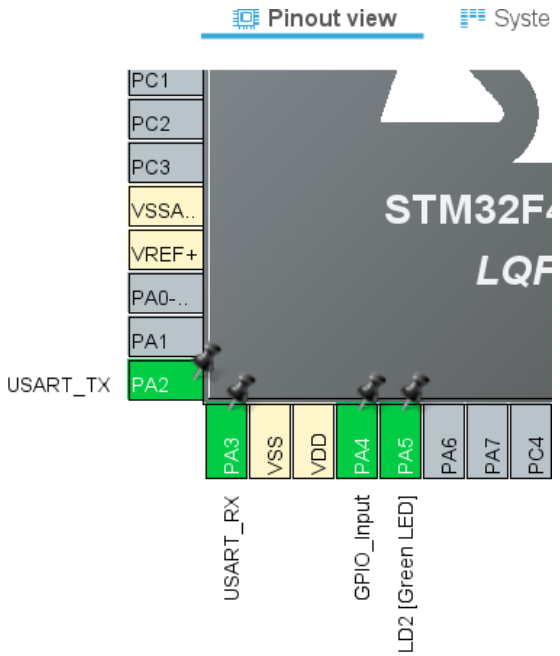
SYS | USART | NVIC | GPIO | Single Mapped Signals | RCC

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
EXTI line[15:10] interrupts	☑	0	0



Usart communication





Edit: toggle button

```
41
42 /* Private variables ---------------
43
44 /* USER CODE BEGIN PV */
45 UART_HandleTypeDef huart2;
46 uint8_t flag=0;
47 /* USER CODE END PV */
```

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

    if(flag==1){
        HAL_GPIO_TogglePin (GPIOA, GPIO_PIN_5);
        flag=0;
    }
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

```
238
239 /* USER CODE BEGIN 4 */
240 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
241     flag=1;
242 }
243 /* USER CODE END 4 */
```

Usart communication