Laboratory 1: Basic MCU development on STM32

Objectives

1. Understand the STM32 microcontroller
2. Introduce the basic of C programming using STM32CubeIDE
3. Understand the step debugging for MCU on Eclipse platform
4. Understand the System clock configuration
5. Understand the STM32Cube embedded software

Embedded System

"An embedded system is one that has embedded software and computer-hardware, which makes it a system dedicated for an application(s) or specific part of an application or product or a part of a larger system." – Raj Kamal

Microcontroller

A microcontroller (MCU) is a System-On-Chip (SoC) that contain a processor core, memory, and some peripherals Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialize in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).

https://en.wikipedia.org/wiki/Embedded_system

https://en.wikipedia.org/wiki/Microcontroller

STM32 32-bit ARM Cortex MCUs

The STM32 family of 32-bit Flash microcontrollers based on the ARM® Cortex®-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

The unparalleled and large range of STM32 devices, based on an industry-standard core and accompanied by a vast choice of tools and software, makes this family of products the ideal choice, both for small projects and for entire platform decisions.

http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus.html?querycriteria=productId=SC1169

STM32 applications

- Industrial
  - PLC
  - Inverters
  - Printers, scanners
  - Industrial networking
  - Solar inverters

- Medical
  - Glucose meters
  - Portable medical care
  - VPAP, CPAP
  - Patient monitoring

- Buildings and security
  - Alarm systems
  - Access control
  - HVAC
  - Power meters

- Appliances
  - 3-phase motor drives
  - Application control
  - User interfaces
  - Induction cooking

- Consumer
  - Home audio
  - Gaming
  - PC peripherals
  - Digital cameras, GPS

Coffee machine

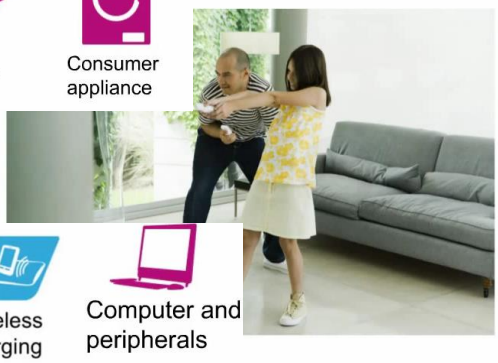
Blenders

Class B

Vacuum cleaner

Induction cooking

Consumer appliance



User interface

Gaming

USB

CEC

Digital TV

Wireless charging

Computer and peripherals

IoT

Battery charger

Industrial

Street light

Smart power

CEC

Power tools

Motor control

STM32Cube Embedded Software

With STM32Cube, STMicroelectronics provides a comprehensive software tool, significantly reducing development efforts, time and cost.

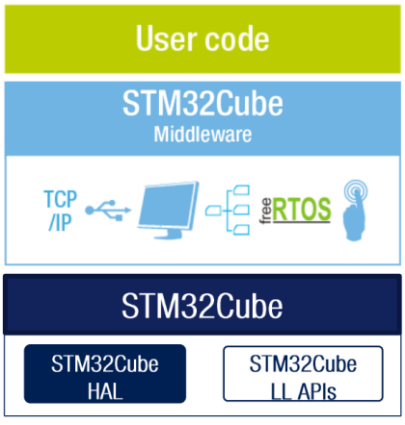
STM32Cube consists of (usable together or independantly):

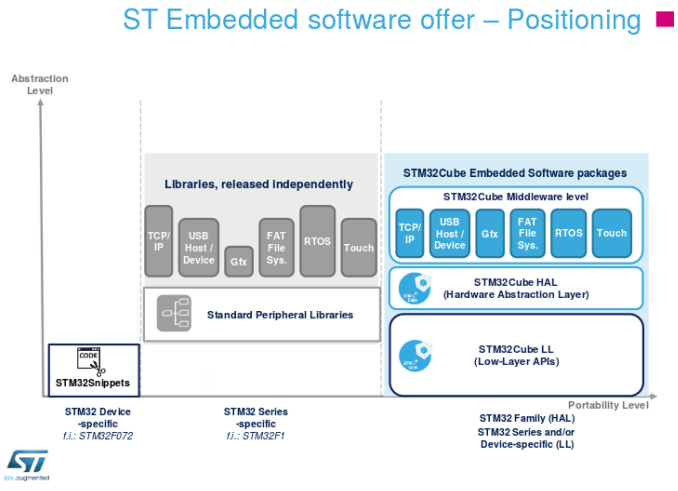
The STM32CubeMX, featuring

- Configuration C code generation for pin multiplexing, clock tree, peripherals and middleware setup with graphical wizards
- Generation of IDE ready projects for a integrated development environment tool chains
- Power consumption calculation for a user-defined application sequence
- Direct import of STM32 Cube embedded software libraries from st.com
- Integrated updater to keep STM32CubeMX up-to-date

STM32Cube embedded software libraries, including:

- The HAL hardware abstraction layer, enabling portability between different STM32 devices via standardized API calls
- The Low-Layer (LL) APIs, a light-weight, optimized, expert oriented set of APIs designed for both performance and runtime efficiency
- A collection of Middleware components, like RTOS, USB library, file system, TCP/IP stack, Touch sensing library or Graphic Library (depending on the MCU series)





http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software.html

STM32CubeMX – STM32Cube initialization code generator

STM32CubeMX is part of STMicroelectronics STMCube™ original initiative to ease developers life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

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

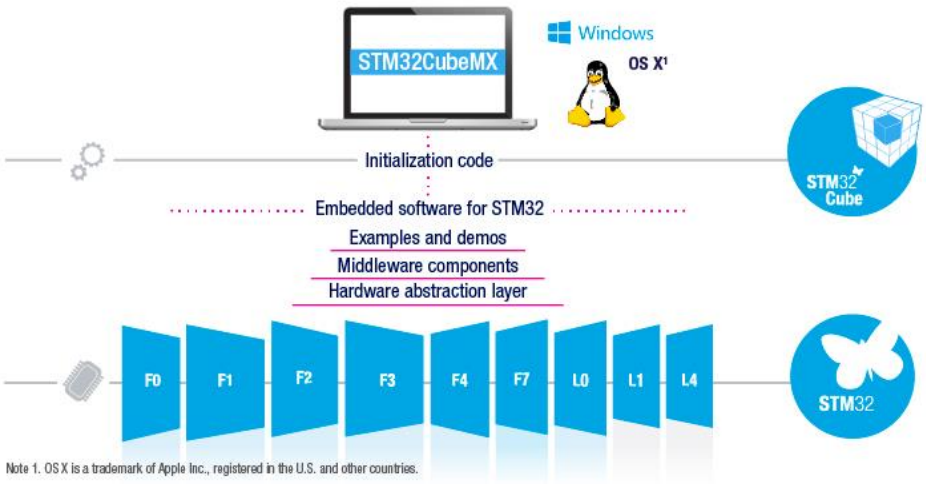
It also embeds a comprehensive software platform, delivered per series (such as STM32CubeF4 for STM32F4 series). This platform includes the STM32Cube HAL (an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio), plus a consistent set of middleware components (RTOS, USB, TCP/IP and graphics). All embedded software utilities come with a full set of examples.

STM32CubeMX is a graphical tool that allows configuring STM32 microcontrollers very easily and generating the corresponding initialization C code through a step-by-step process.

Step one consists in selecting the STMicroelectronics STM32 microcontroller that matches the required set of peripherals.

The user must then configure each required embedded software thanks to a pinout-conflict solver, a clock-tree setting helper, a power-consumption calculator, and an utility performing MCU peripheral configuration (GPIO, USART, ..) and middleware stacks (USB, TCP/IP, ...).

Finally, the user launches the generation of the initialization C code based on the selected configuration. This code is ready to be used within several development environments. The user code is kept at the next code generation.



http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html

STM32CubeIDE

STM32CubeIDE is an advanced C/C++ development platform with IP configuration, code generation, code compilation, and debug features for STM32 microcontrollers. It is based on the ECLIPSE™/CDT framework and

GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the ECLIPSE™ IDE.
STM32CubeIDE integrates all STM32CubeMX functionalities to offer all-in-one tool experience and save installation and development time. After the selection of an empty STM32 MCU or preconfigured microcontroller from the selection of a board, the project is created and initialization code generated. At any time during the development, the user can return to the initialization and configuration of the IPs or middleware and regenerate the initialization code with no impact on the user code.
STM32CubeIDE includes build and stack analyzers that provide the user with useful information about project status and memory requirements.
STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyzer. (From ST website)

STM32F4

STM32F4 series of high-performance MCUs with DSP and FPU instructions

The ARM® Cortex®-M4-based STM32F4 series MCUs leverage ST's NVM technology and ST's ART Accelerator™ to reach the industry's highest benchmark scores for Cortex-M-based microcontrollers with up to 225 DMIPS/608 CoreMark executing from Flash memory at up to 180 MHz operating frequency.

STM32F4XX

The STM32F4XX lines are designed for medical, industrial and consumer applications where the high level of integration and performance, embedded memories and rich peripheral set inside packages as small as 10 x 10 mm are required.
The STM32F407/417 offers the performance of the Cortex™-M4 core (with floating point unit) running at 168 MHz.

Performance: At 168 MHz, the STM32F4XX deliver 210 DMIPS/566 CoreMark performance executing from Flash memory, with 0-wait states using ST's ART Accelerator. The DSP instructions and the floating point unit enlarge the range of addressable applications.

Power efficiency: ST's 90 nm process, ART Accelerator and the dynamic power scaling enables the current consumption in run mode and executing from Flash memory to be as low as 238 µA/MHz at 168 MHz.

Rich connectivity: Superior and innovative peripherals: Compared to the STM32F4x5 series, the STM32F4XX product lines feature Ethernet MAC10/100 with IEEE 1588 v2 support and a 8- to 14-bit parallel camera interface to connect a CMOS camera sensor.

http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f407-417.html?querycriteria=productId=LN11

STM32 F4

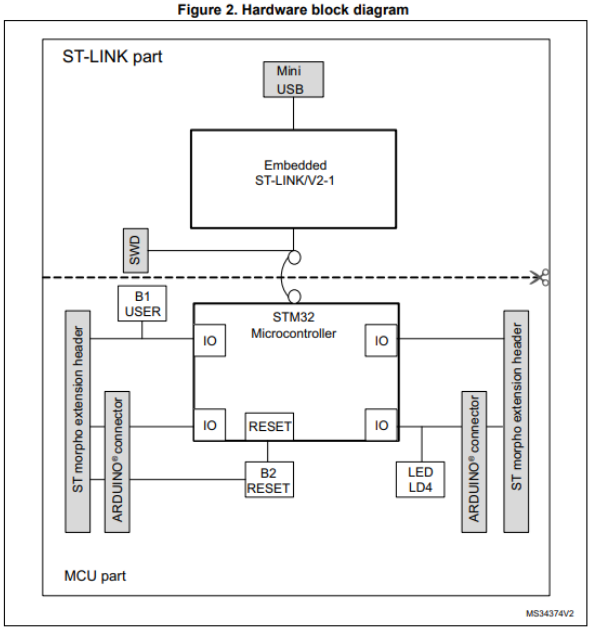
NUCLEO-F411

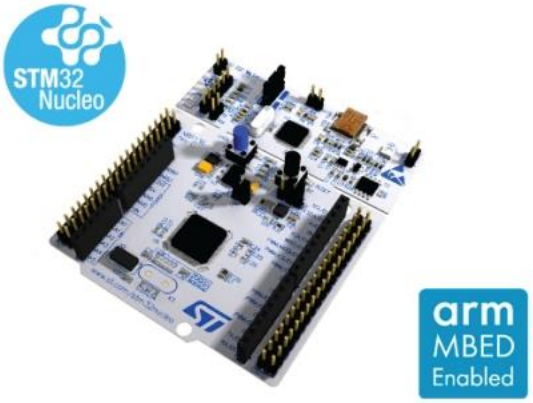
The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller.

Traditionally, Microcontroller board requires an external board for programming and debugging. In case of Nucleo board family, ST provides this functionality through the Mini-USB through ST-LINK protocol. See more details about the board here.

https://www.st.com/en/evaluation-tools/nucleo-f411re.html

Figure 2. Hardware block diagram





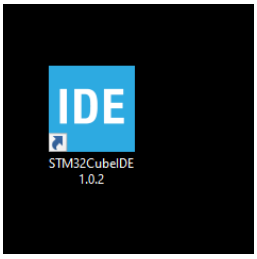
Digital Logic

Digital Logic is an alternative term for ***Digital Electronic (or Digital Circuits)***. Digital is a way for representing the electronic signals as High (True) and Low (False) rather than the real analog values (voltages). Thus, small changes in the analog levels (voltages), such as noise from circuits or communication channels, does not effect the digital signal. This allows electronic devices to switch to known states than producing specific values (voltages).

Usually, a signal level (voltage) near a reference ground level is called Low or False or 0. A signal level (voltage) near the supply voltage is called High or True or 1. With this notion in our hands, we can simply refer to our digital value as high or low.

Getting Started

1 Run System Workbench for STM32.



It will ask you to create a workspace. Choose work space that you would like your program to be in. **However, make sure that there is no space in any of the pathname.** For example

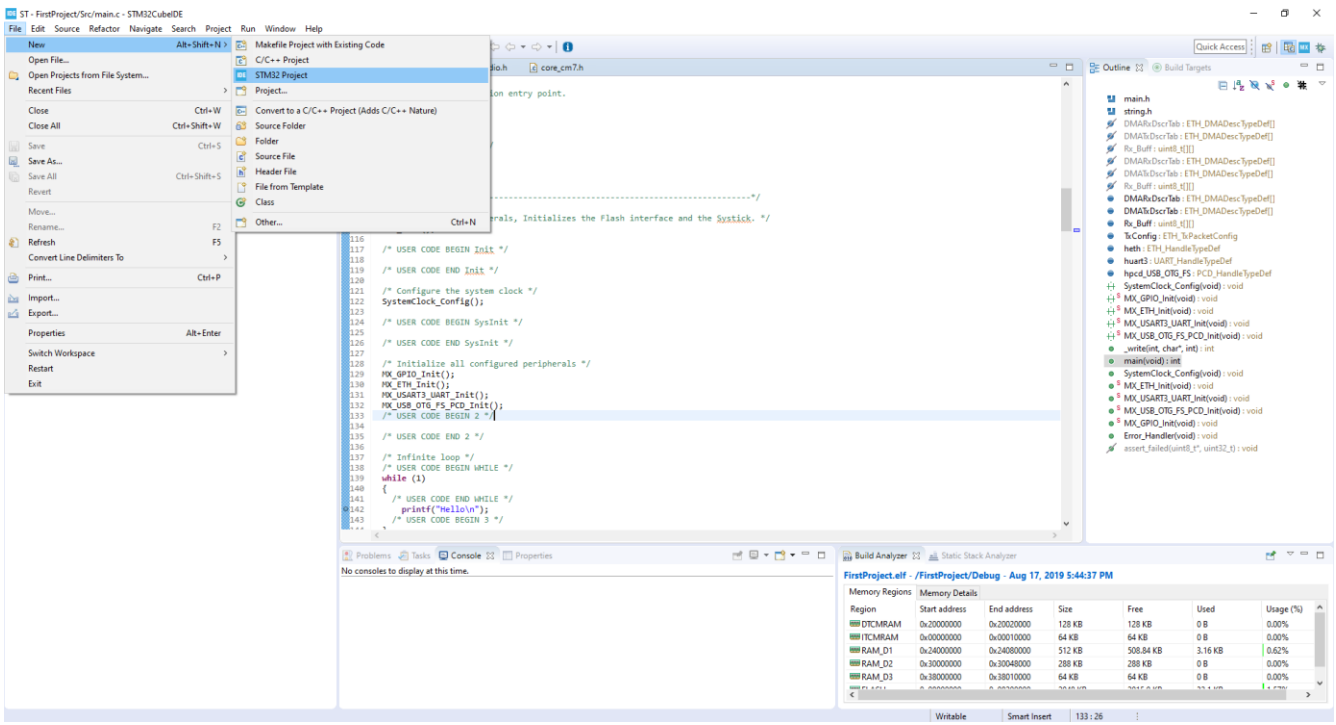
C:\ST\STM32

Is ok, but

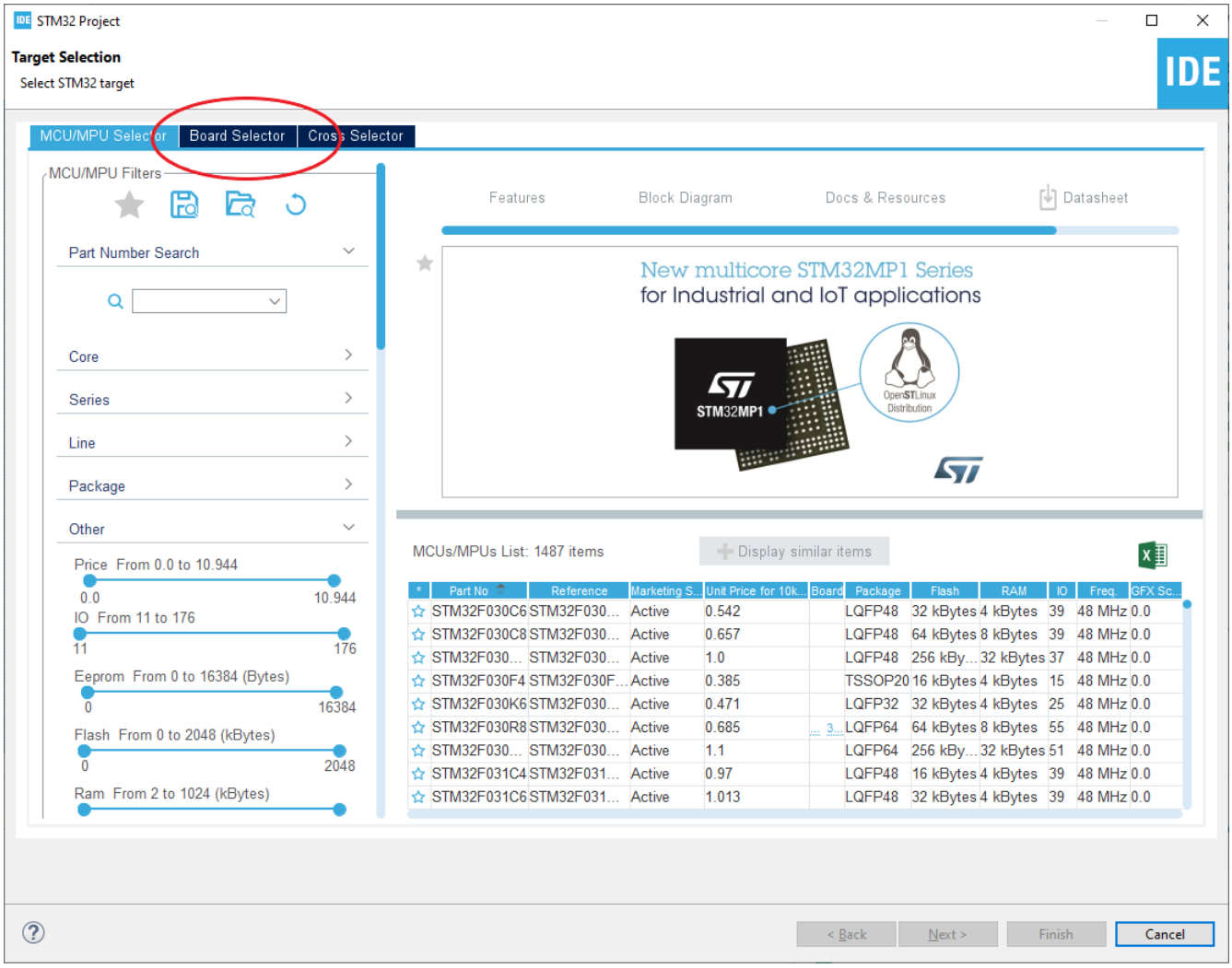
C:\my work\STM32

Will lead to some problems later on.

2 Choose File -> New -> STM32 Project



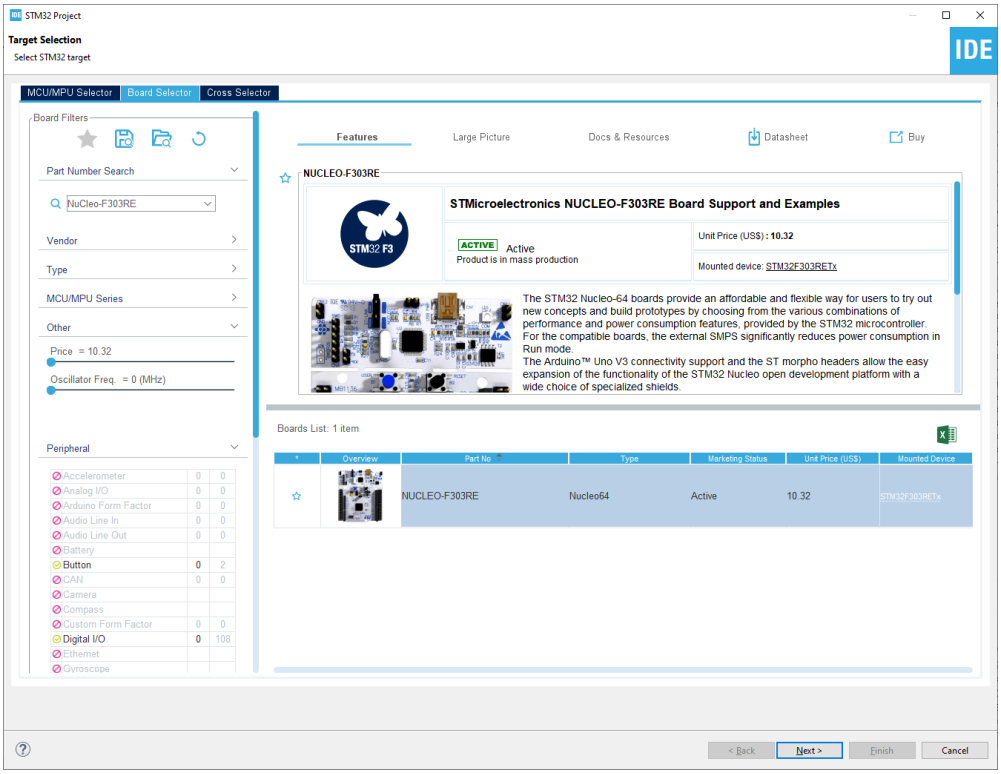
This window will appear



Choose "Board Selector"

3 In "Board Selector" Put part number of the board you are using in the "Part Number Search"

For those of you who has, NUCLEO Board (The white one) Put in "NUCLEO-F411RE". Note that if you have a slightly different model, you must put in the correct number as written on the board. For example, if somehow, you have this NUCLEO-F303RE, you must choose that name



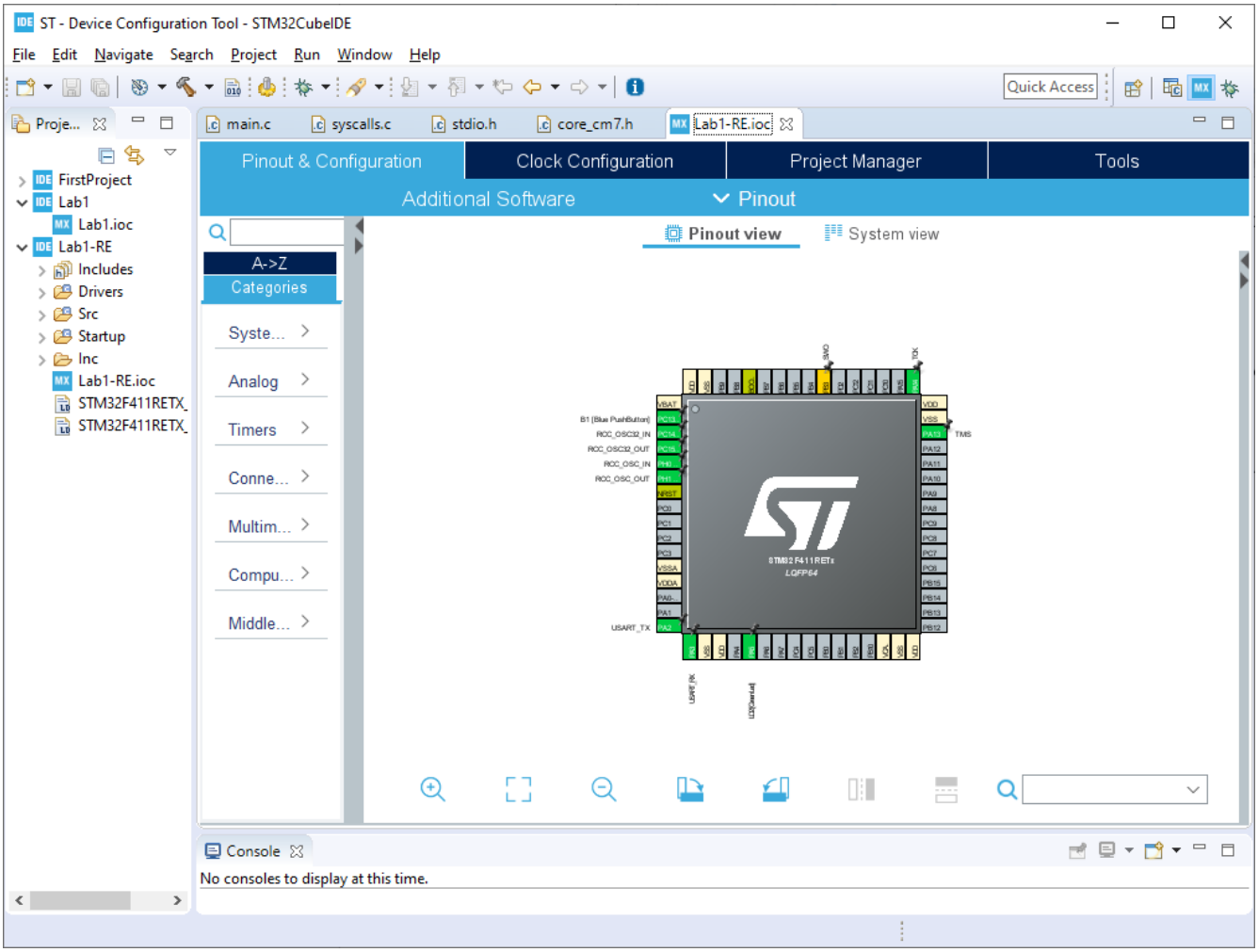
For those of you who do not have the board yet, and use the green board (STM32F4-Discovery) Put in "STM32F4DISCOVERY"

Hit Next and Choose a project name. Let's call it "Lab1"

When it is asking to "initialize all peripherals with their default mode? " choose yes. It may ask about perspective change. Please choose yes as well.

If you have not setup the software before, this process will take a while to download and install.

4. Now you will see the following page.

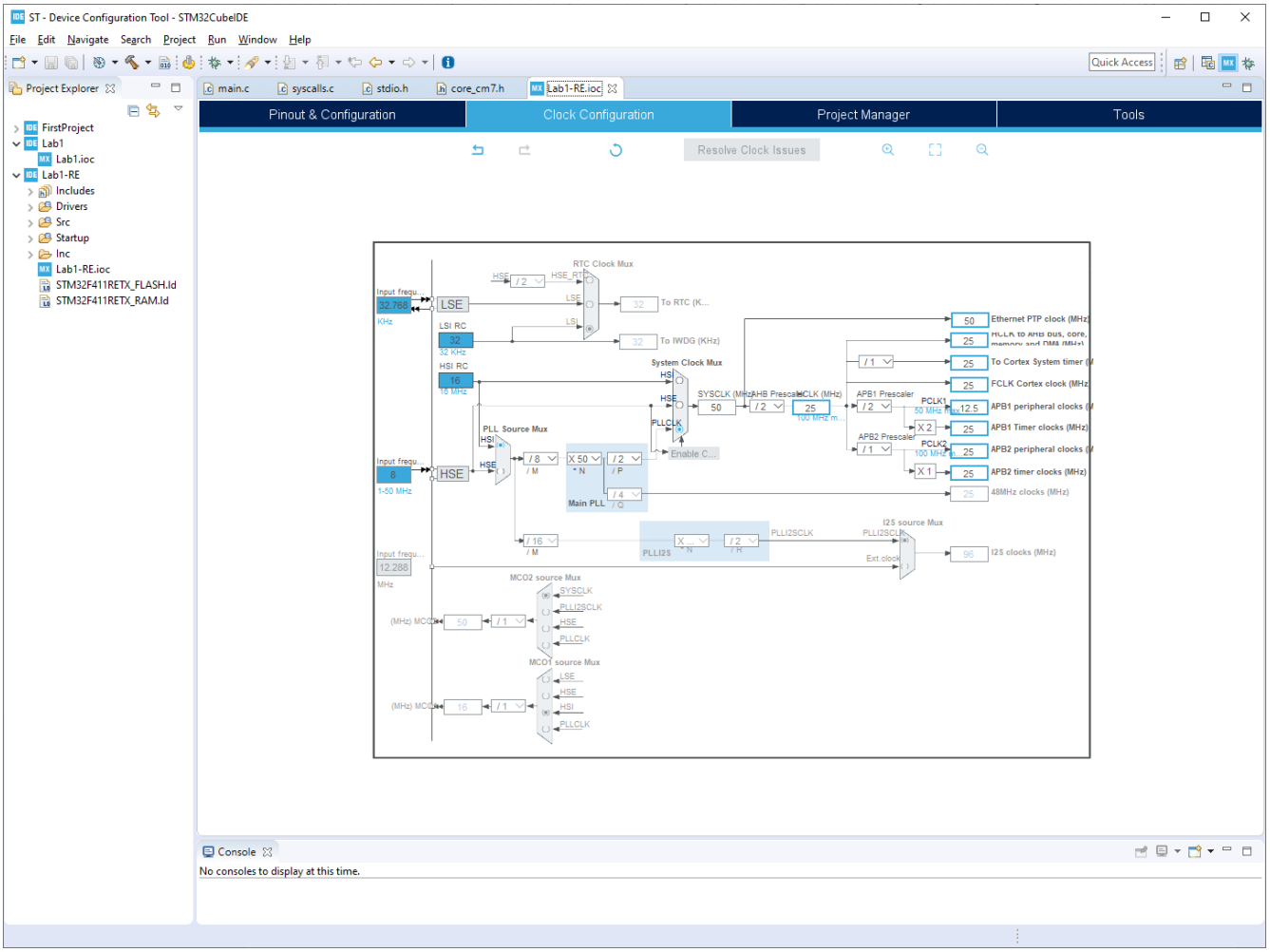


4 On Pinout Tab, you can choose mode of the PINs you would like to set. For this Lab, we will be trying to set LED light to blink. If you have STM32F4-Discovery Board, LEDs indicators is connected to PINs PD12-PD15 has been selected (Green highlighted).

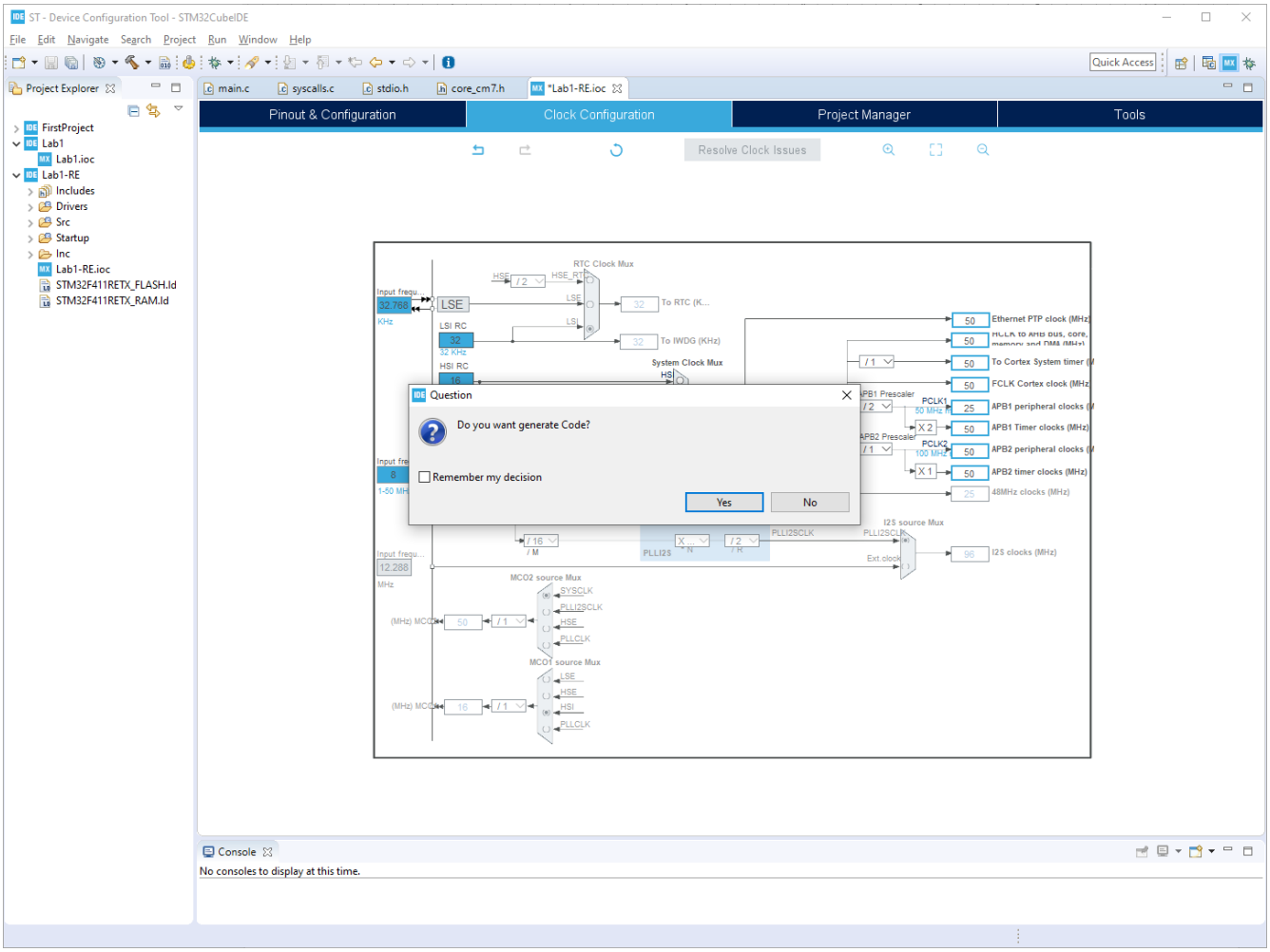
If you have Nucleo-Board, LED is connected to either PA5 or PB13. (Most likely PA5)

If you click at the name of the PIN, you can set the PIN to be different types. For the purpose of driven LED, it is a GPIO_Output. If we are to set it for input, then we can use GPIO_Input.

5 Microcontroller is typically not just one system, but many sub-systems combined together. Each of sub-systems may operate at a different clock speed. For your purpose, the main speed of the CPU is HCLK. Please try to set HCLK to 25



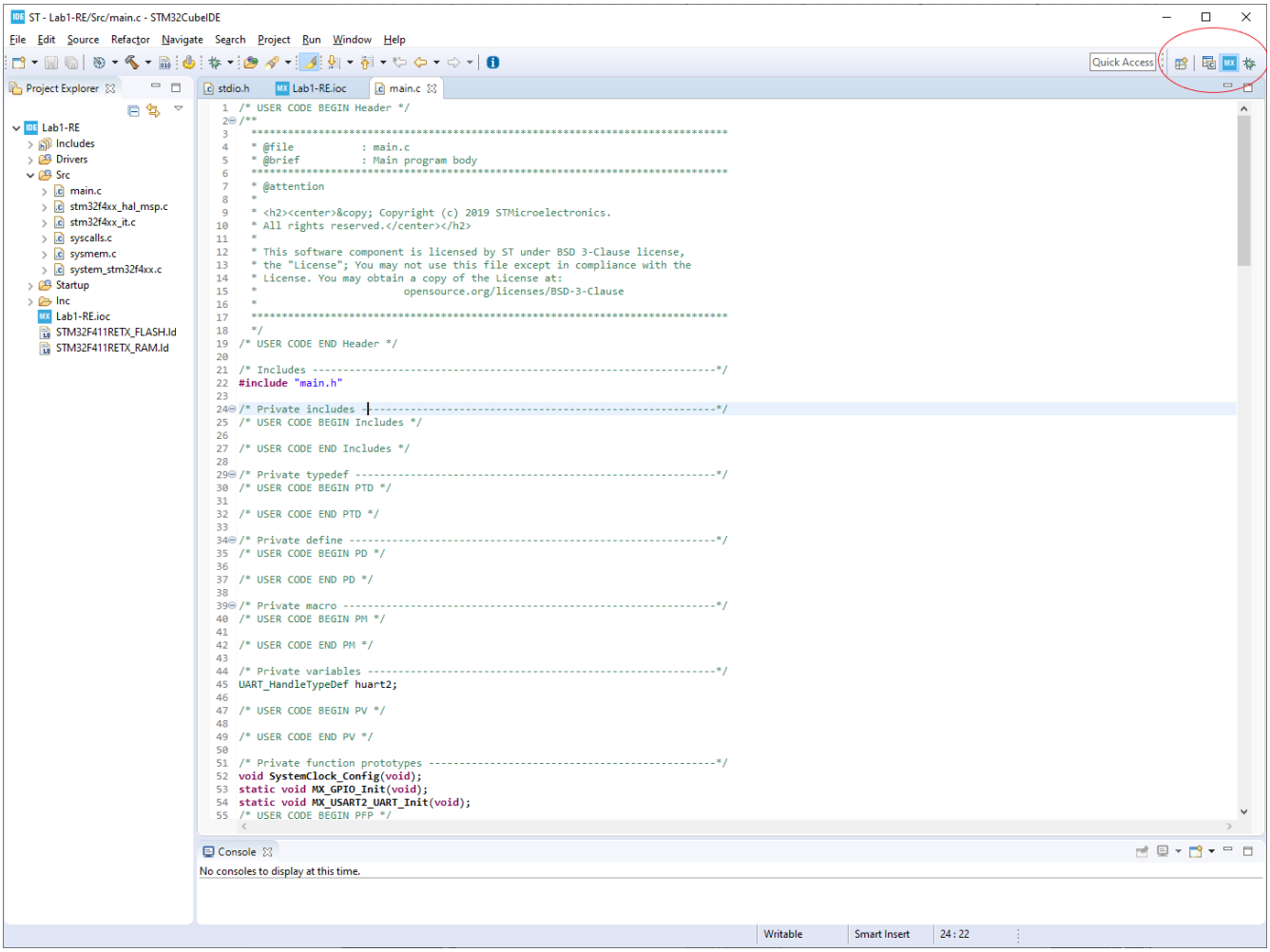
6 Click File -> Save, it will ask you the following question:



Choose Yes

7 Now, the program will generate new code.

In the left Windows pane, you can see a list of file in "Project Explorer." If you do not have "Project Explorer" You may try changing the project perspective by change using the icon on the top right corner. Expand the **project name-> Src**, then double click **main.c**



This code is generated from the tool you were using in the previous section

Scroll down and you will see that there are

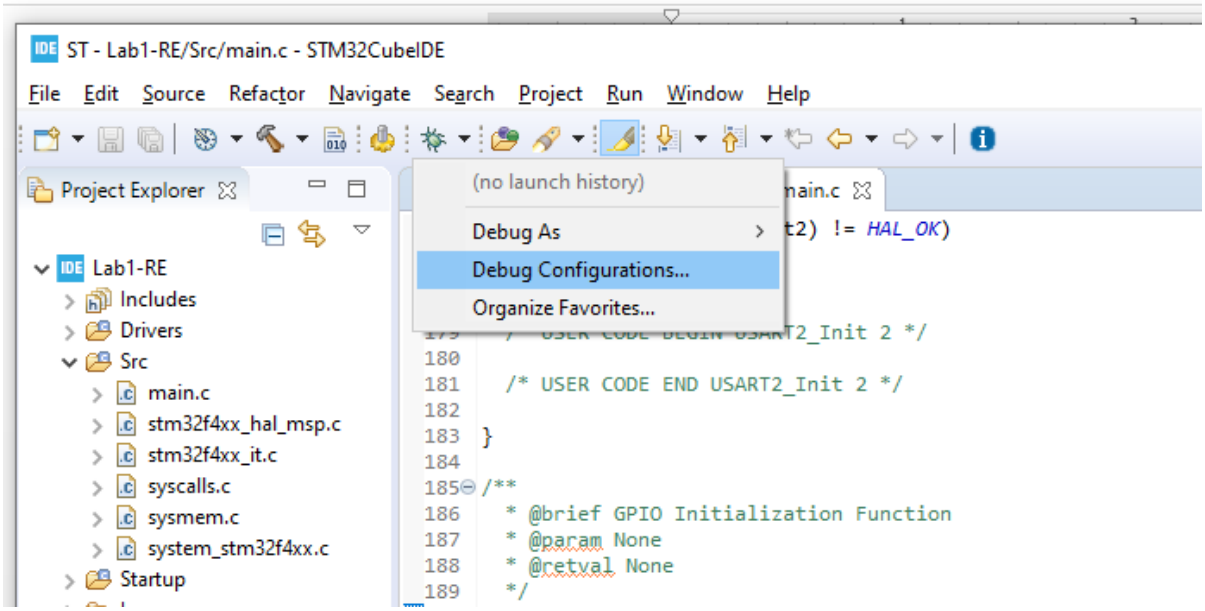
```
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

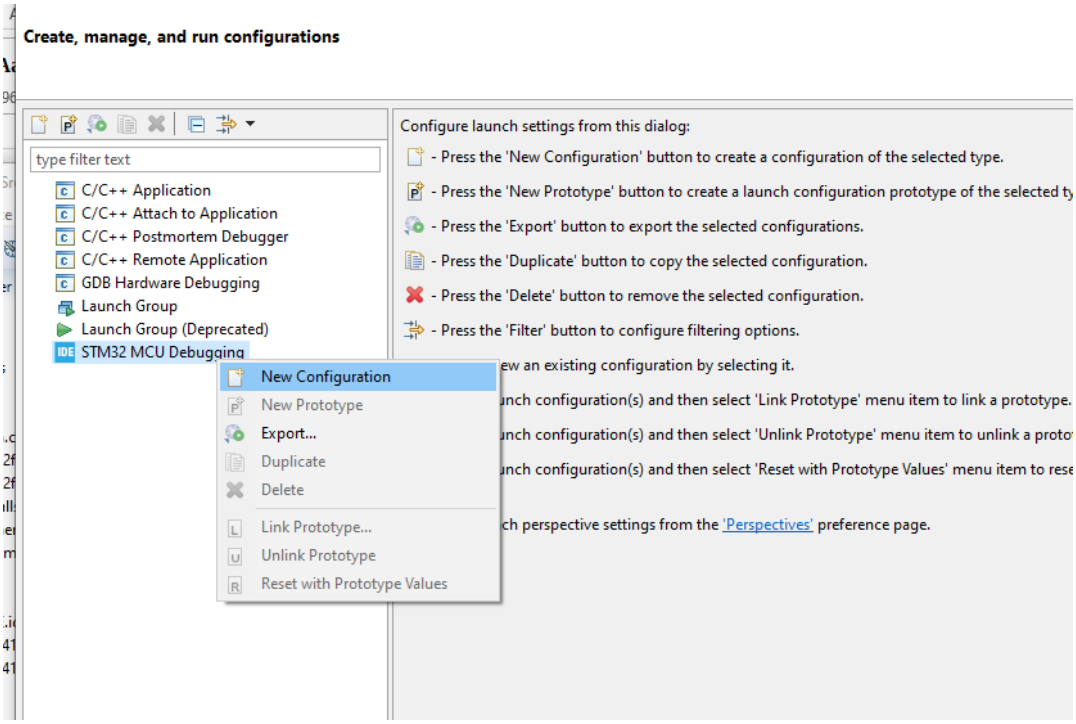
This tag is important for the code generator. If you want to be able to use the code generator to change any setting later on, all your code **must be** in between the comment of USER CODE BEGIN …… END

8 To compile the program, choose Project -> Build all

9 To upload the program to STM32 and start debugging,

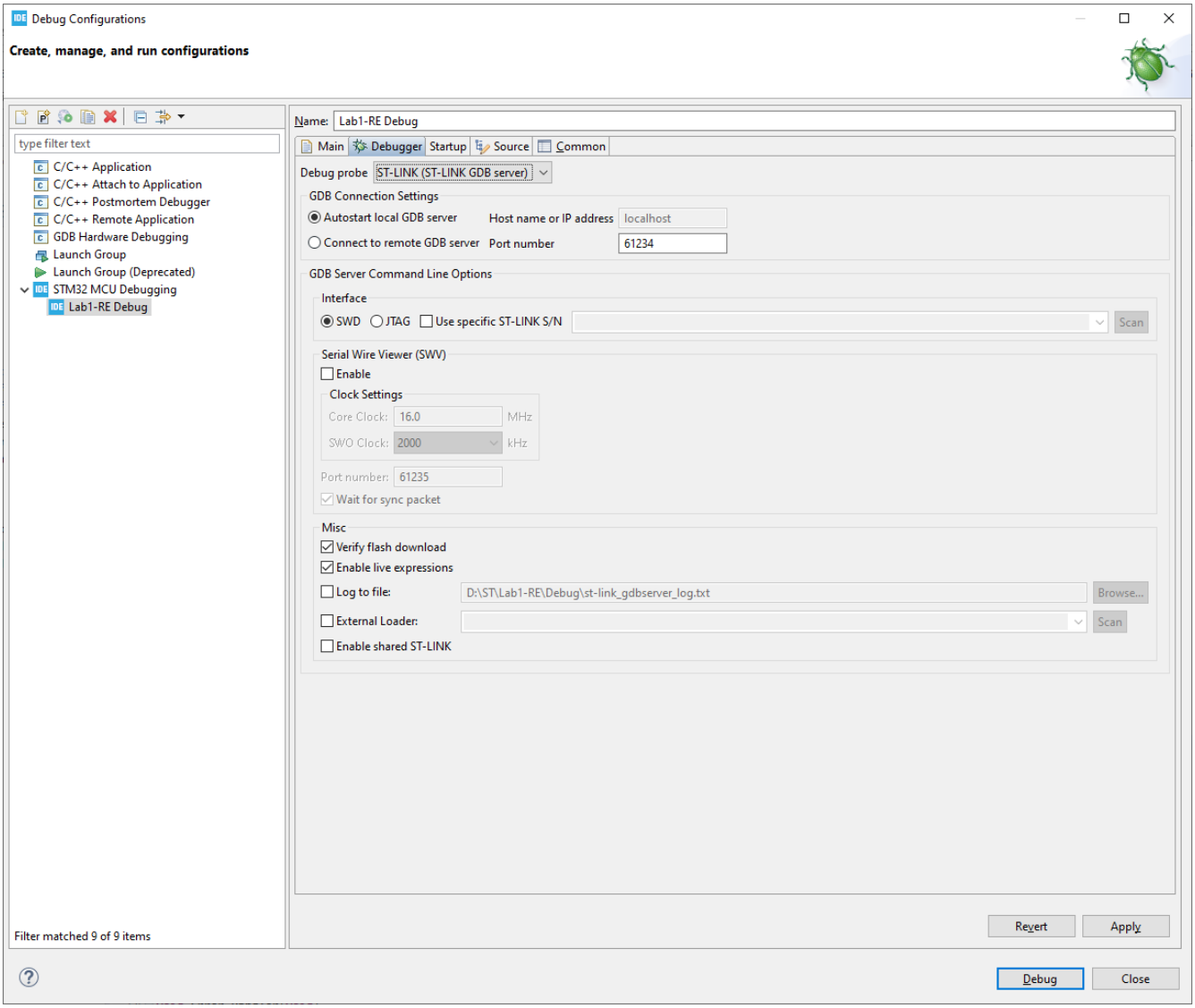


Then right click at "STM32 MCU Debugging"



In this page, choose "Search Project", then Hit OK.

Now, switch to "debugger plane"



If you are using the board STM32F4Discovery from the lab, they are likely to be pre-install debugger as J-Link, if not, it would be ST-Link. You can change the debugger probe between ST-Link and J-Link here.

You can then hit "Debug"

Lab Exercises

1 Create a new STM32 Project by following the Getting Started guide, and insert new main function code as below.

```
int main(void)
{

  /* USER CODE BEGIN 1 */
    uint32_t i,j;
```

```
  /* 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 */
  MX_GPIO_Init();

  /* USER CODE BEGIN 2 */
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
            int j;
            GPIOA->ODR ^= 1<<5;
            for(j=0;j<2000000;j++);


  /* USER CODE BEGIN 3 */

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

2 Using debugging feature, set Breakpoint at "for loop", then debug on STM32F4 Discovery Board.

2.1 From STM32F4XX Reference manual, what are these GPIO registers

2.1.1 GPIOD_MODER

port mode register

2.1.2 GPIOD_OTYPER

port output type register

2.1.3 GPIOD_OSPEEDR

port output speed register

2.1.4 GPOD_PUPDR

port pull-up / pull down register

2.1.5 GPIOD_ODR

port output data register

2.2 Using step/suspend/resume debugging feature on SW4STM32, what is the value of GPIOD_MODER and GPIOD_ODR from the start of debugging and breakpoint. What are the relations to LEDs' Discovery Board.

	start	breakpoint
GPIOA_MODER	0xa8000000	0xa8000400
GPIOA_ODR	0xa8000000	0x20

The LED toggled.

3 Create a new project with 4 times speed of System Clock using "Clock Configuration" on STM32CubeMX. What are the value of PLLP, PLLN and PLLM register, before and after set the new speed. (Look at RCC register on STM32F407 Reference manual)

HCLK = 21

PLLP1 = 0x0
PLLP2 = 0x0

PLLN		PLLM5	
8	0x0	4	0x0
7	0x1	3	0x1
6	0x1	2	0x0
5	0x0	1	0x0
4	0x0	0	0x0
3	0x0		
2	0x0		
1	0x0		
0	0x0		

HCLK = 84

PLLP1 = 0x0
PLLP2 = 0x0

PLLN		PLLM5	
8	0x0	4	0x1
7	0x1	3	0x0
6	0x1	2	0x0
5	0x0	1	0x0
4	0x0	0	0x0
3	0x0		
2	0x0		
1	0x0		
0	0x0		

4 Use a STM32 Cube embedded software libraries instead of direct register assignation and for loop delay (using functions from stm32f4xx_hal.c and stm32f4xx_hal_gpio.c in STM32F4xx_HAL_Driver) to create a same behavior as example code on 1.

```
HAL_GPIO_TogglePin(GPIOA, 1<<5);
HAL_Delay(1000);
```