Programming using CubeIDE #1- Blinking a LED

Senura ENTC 16 **Apr '20**

Hi guys!.

Hope you are now able to set up the environment to start programming with STM32. If you have any questions or problems regarding our previous post (setting up the environment needed to begin with STM32) feel free to ask, because if you want to check your code in the real world that knowledge is a must.

Today, we will be looking at the key features of STM32CubeIDE and how a new project is created

Key Features

1.Integration of STM32CubeMX that provides services for:

- Project creation and STM32 microcontroller and microprocessor selection
- Pinout, clock, peripheral, and middleware configuration
- Generation of the initialization code

2.Based on ECLIPSE™/CDT, with support of ECLIPSE™ add-ons, GNU C/C++ for Arm® toolchain and GDB debugger

3.Additional advanced debug features including:

- CPU core, peripheral register, and memory views
- Live variable watch view
- System analysis and real-time tracing (SWV)
- CPU fault analysis tool

4.Support of ST-LINK (STMicroelectronics) and J-Link (SEGGER) debug probes

5.Import project from Atollic® TrueSTUDIO® and AC6 System Workbench for STM32 (SW4STM32)

6.Multi-OS support: Windows®, Linux®, and macOS®, 64-bit versions only

Project creation and STM32 microcontroller and microprocessor selection

Workspace and project creation :

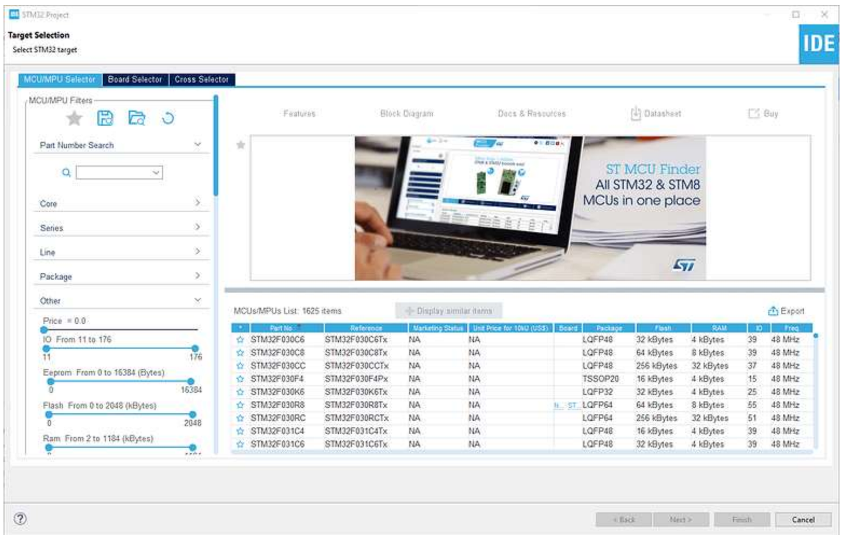
The first thing to be done when using CubeIDE is to create a project. Before that we will introduce you, the concept of workspace.

A workspace is a container that includes project folders or information about project folders, and a .metadata folder that contains information about the projects. A workspace is simply a folder on a hard drive, which can be located anywhere in the storage media. When STM32CubeIDE starts up, it asks which workspace must be used. This may be changed at any time by selecting [File]>[Switch Workspace] and navigating to another folder.

The easiest way to create a new embedded project is to use the STM32 Project wizard. It is selected through the [File]>[New]>[STM32 Project] menu command, and launches the embedded MCUFinder:

1 - Select the target MCU or board and click next (We have chosen STM32F103C8 MCU).

- This image shows you the target selection window. In light blue color in the upper left corner of this image, you can see now we are at the MCU/MPU selector tab.


STM32 Project

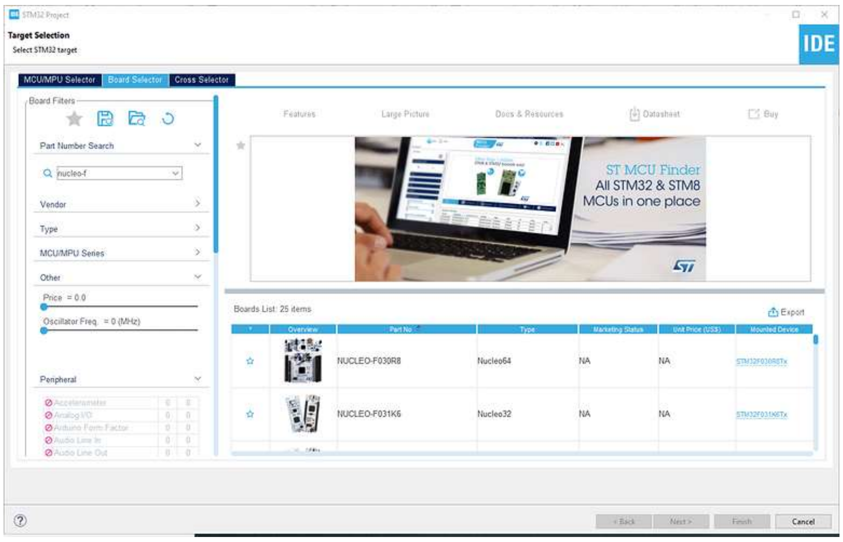
Target Selection

Select STM32 target

MCUs/MPUs List: 1625 items

*	Part No	Reference	Marketing Status	Unit Price for 10kU (US$)	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F030C6	STM32F030C6Tx	NA	NA		LQFP48	32 kBytes	4 kBytes	39	48 MHz
☆	STM32F030C8	STM32F030C8Tx	NA	NA		LQFP48	64 kBytes	8 kBytes	39	48 MHz
☆	STM32F030CC	STM32F030CCTx	NA	NA		LQFP48	256 kBytes	32 kBytes	37	48 MHz
☆	STM32F030F4	STM32F030F4Px	NA	NA		TSSOP20	16 kBytes	4 kBytes	15	48 MHz
☆	STM32F030K6	STM32F030K6Tx	NA	NA		LQFP32	32 kBytes	4 kBytes	25	48 MHz
☆	STM32F030R8	STM32F030R8Tx	NA	NA	N... ST...	LQFP64	64 kBytes	8 kBytes	55	48 MHz
☆	STM32F030RC	STM32F030RCTx	NA	NA		LQFP64	256 kBytes	32 kBytes	51	48 MHz
☆	STM32F031C4	STM32F031C4Tx	NA	NA		LQFP48	16 kBytes	4 kBytes	39	48 MHz
☆	STM32F031C6	STM32F031C6Tx	NA	NA		LQFP48	32 kBytes	4 kBytes	39	48 MHz

As you can see in the picture below, now we are in the Board Selection tab.


STM32 Project

Target Selection

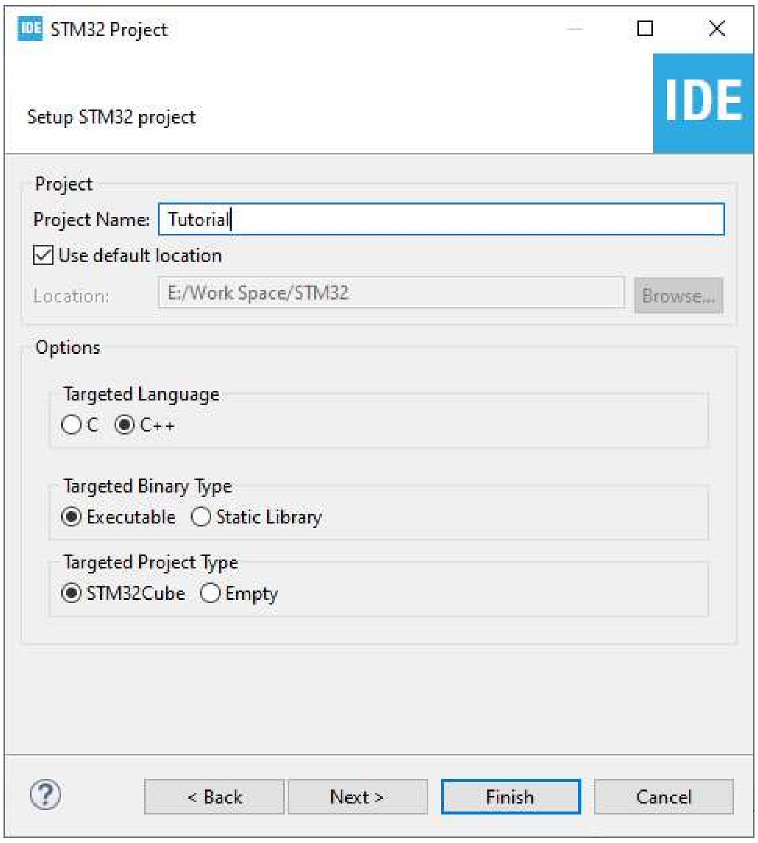
Select STM32 target

Part Number Search: nucleo-f

Boards List: 25 items

*	Overview	Part No	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-F030R8	Nucleo64	NA	NA	STM32F030R8Tx
☆		NUCLEO-F031K6	Nucleo32	NA	NA	STM32F031K6Tx

2 - Enter a project name and select the setting wanted for the project in the dialogue boxes. Normally we use C++ as our targeted language.


STM32 Project

Setup STM32 project

Project

Project Name: Tutorial

☑ Use default location

Location: E:/Work Space/STM32

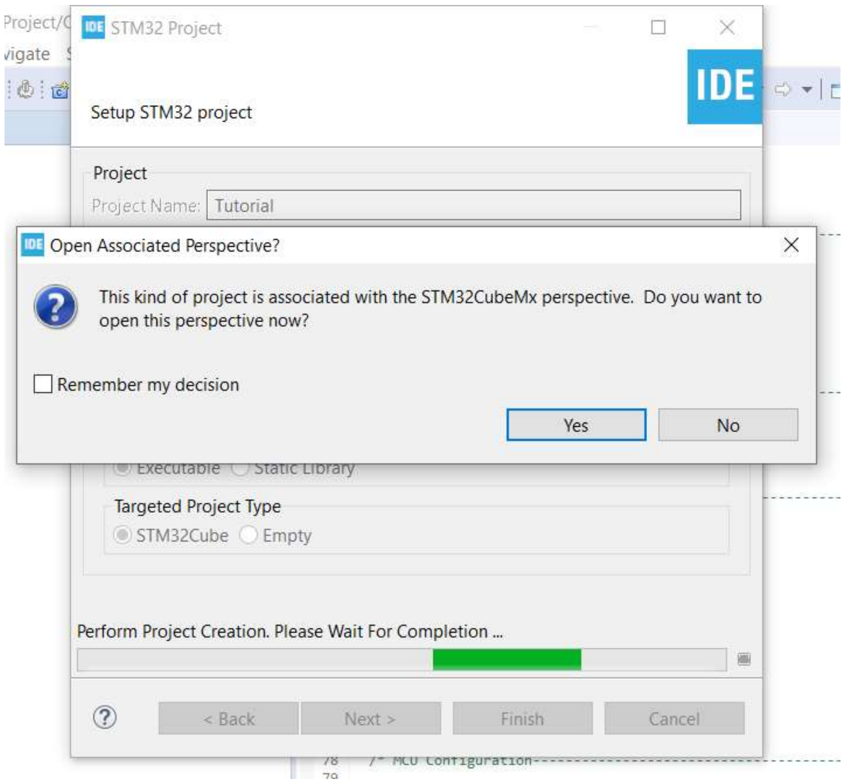
Options

Targeted Language: ○ C [✓ C++]

Targeted Binary Type: [✓ Executable] ○ Static Library

Targeted Project Type: [✓ STM32Cube] ○ Empty

3 - Click on [Finish] and a window as shown below will pop out in your computer. Just click [Yes] and you are done creating your project. By clicking Yes will bring up the STM32CubeMX perspective for configuring the peripherals, clock, middleware, and the power consumption.


Open Associated Perspective?

This kind of project is associated with the STM32CubeMx perspective. Do you want to open this perspective now?

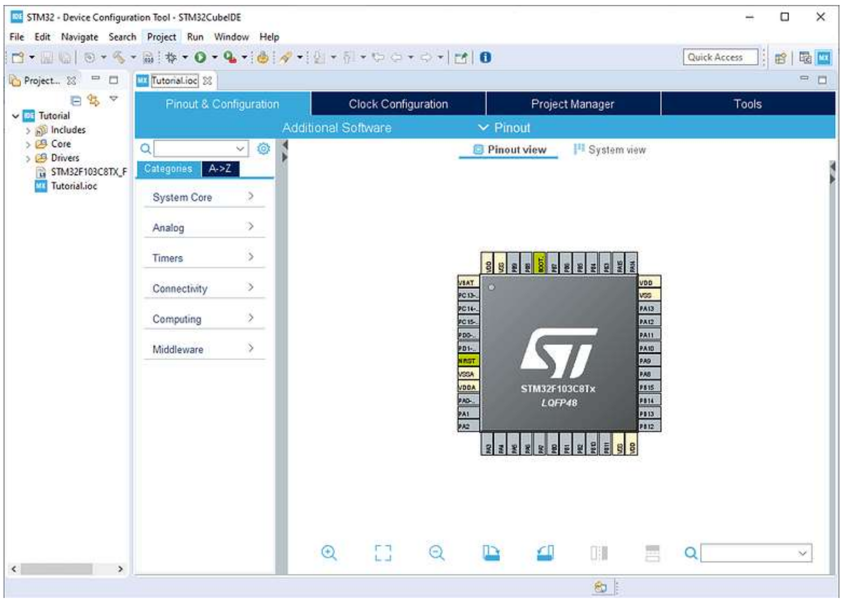
☐ Remember my decision

Perform Project Creation. Please Wait For Completion ...

Now, you have chosen your microcontroller/board and done creating a project.

Pinout, clock, peripheral, and middleware configuration

In the CubeMX perspective, there are mainly 4 tabs and now we're in the Pinout & Configuration tab. As you can see in the left side of the following image, we have some dropdown lists under the Pinout & Configuration tab.

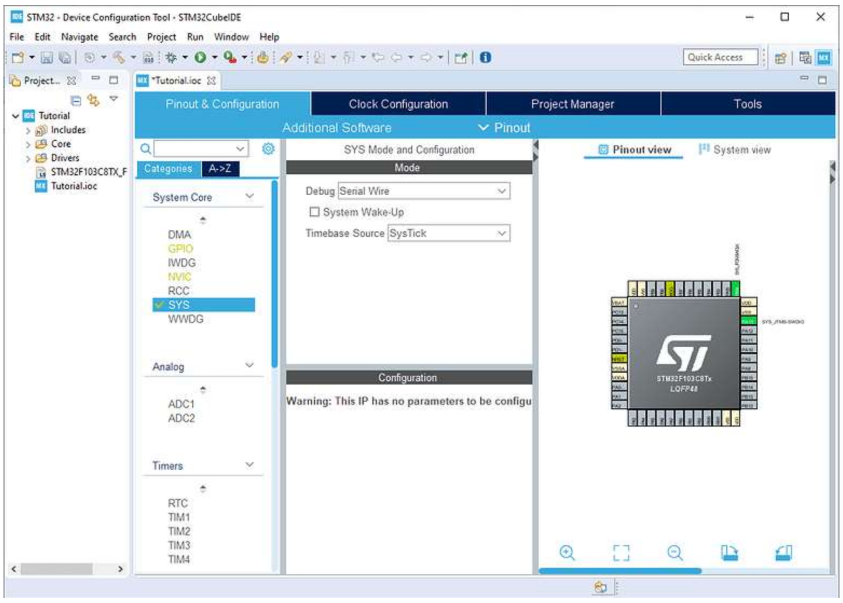


In this guide we'll be using STLink to program our MCU. So that we have to select Serial Wire as our debugging method in CubeIDE.

To do that,

- Click SYS on the System Core drop down list.
- Chose Serial Wire from the Debug drop down list

Now, the two pins(SWDIO, SWCLK) needed to program your MCU using STLink have been configured automatically and you can see them in the Pinout view as in the following image.



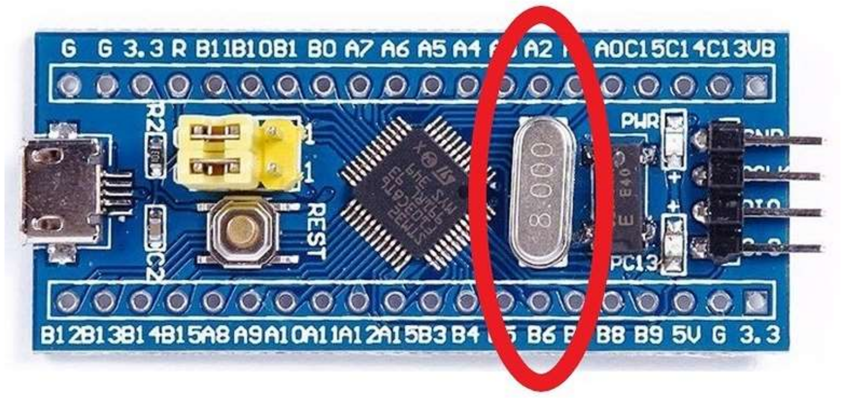
Debug Serial Wire
System Wake-Up
Timebase Source SysTick

Warning: This IP has no parameters to be configu

Clock configuration

If you are using a STM32 Board most probably there will be an external clock connected to your MCU. But if you are directly using a MCU, then there will be no external clock unless you have connected one. In both of these cases you need to know the clock rate of the clock you are using.

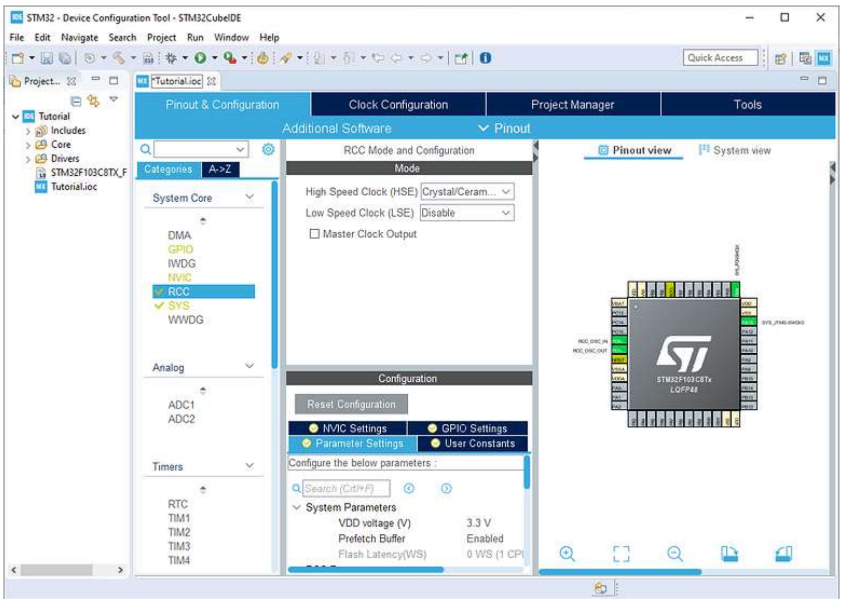
In our case(STM32F103C8 Board) we have an 8MHz external clock.


G G 3.3 R B11B10B1 B0 A7 A6 A5 A4 A2 A0C15C14C13VB
PWR
REST
PC13
B12B13B14B15A8A9A10A11A12A15B3B4 B6 B8 B9 5V G 3.3
8.000

It should be mentioned that almost every STM32 MCU consists of an internal clock too. The reason we are using an external clock is to get a high precision clock.

You can configure your clock by following these steps.

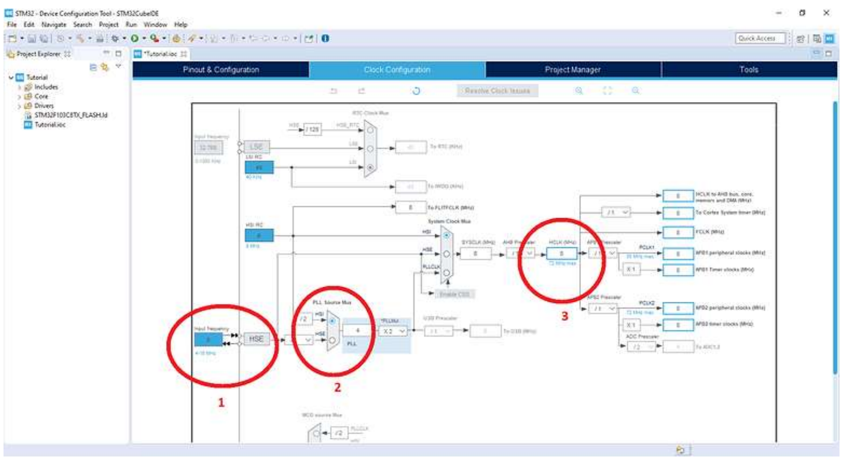
- Click RCC on the System Core drop down list.
- Select Crystal/Ceramic Resonator from the High Speed Clock dropdown list



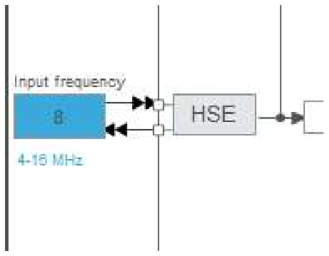
High Speed Clock (HSE) Crystal/Ceram...
Low Speed Clock (LSE) Disable
Master Clock Output

System Parameters
VDD voltage (V) 3.3 V
Prefetch Buffer Enabled
Flash Latency(WS) 0 WS (1 CPU

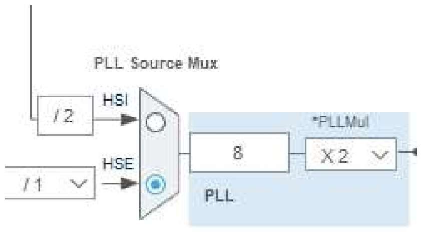
- Next, Click on the Clock Configuration tab. In here you can see the clock tree of your MCU.



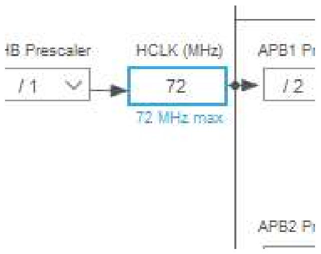
1. Input your clock frequency (In our case we didn't change anything, default clock is 8MHz in STM32F103C8)



2. Change the PLL Source Mux selector to HSE so that you can use the external clock source.

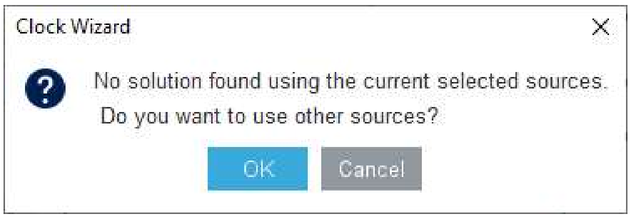


3. Input the desired clock rate that you want to get. (For the demonstration purposes here we have use the maximum rate)



Note - When the clock rate is higher, the power consumption is also higher.

After completing these steps, you will get a pop up window like this. Just click [OK].

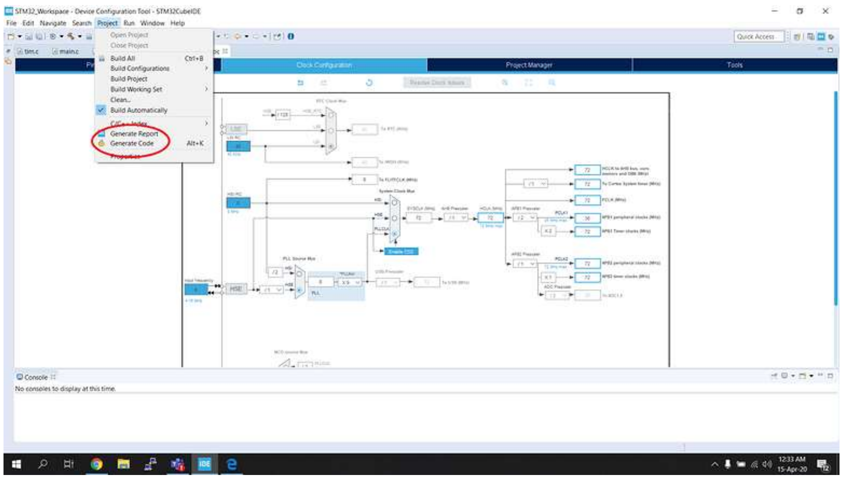

Clock Wizard

No solution found using the current selected sources.
Do you want to use other sources?

What happens here is CubeIDE software automatically configures the clock tree for your desired rates.

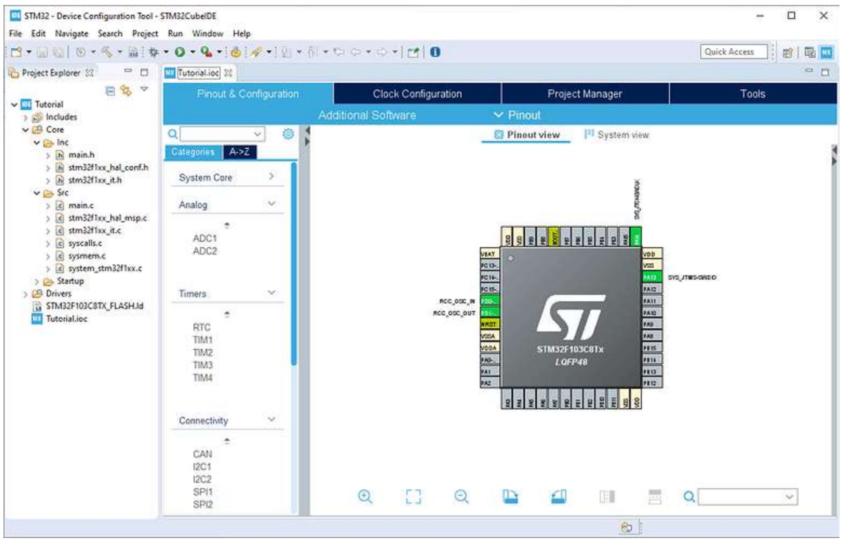
Now you can save your configurations by pressing [CTRL] + [S] or by clicking generate code in the Project tab.

You can generate your code anytime with no limitations(You will have to regenerate the code every time when you change any configuration parameter) .



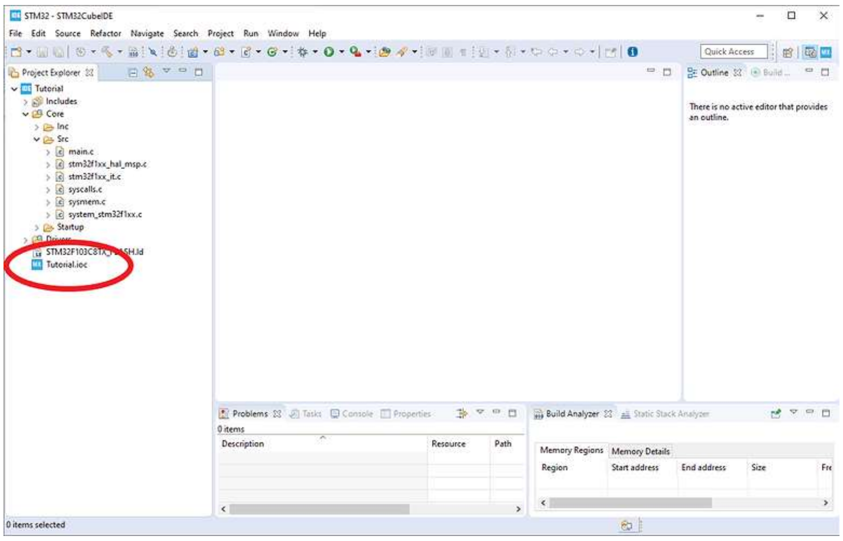
Now the code has been generated for your configurations.

You can find the generated .h(header) files in Core -> Inc in the Project Explorer window and .c(source) files in Core -> Src in the Project Explorer window.



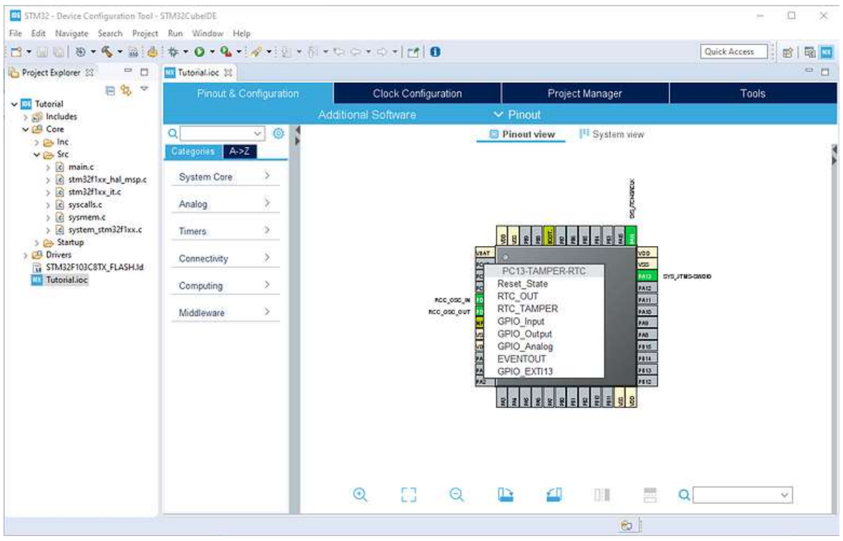
Blinking a LED using CubeIDE

If your pin configuration is not opened due to any reason, you can get it by double clicking [YOUR_PROJECT_NAME] **.ioc** file in the project explorer window.

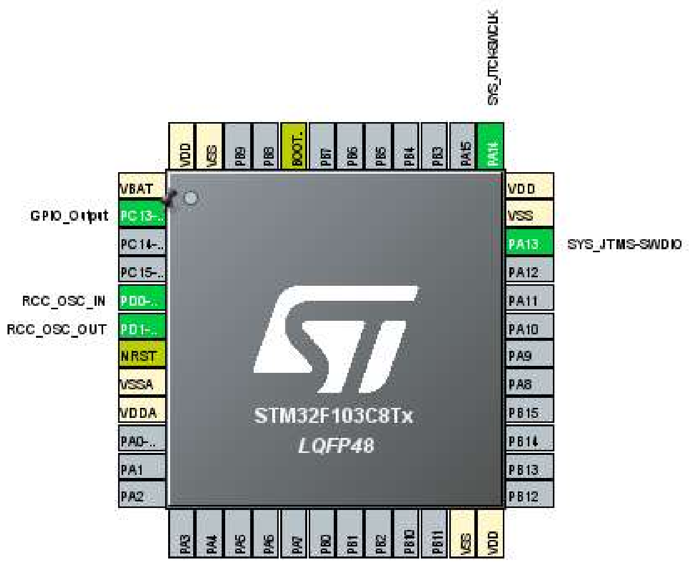


In Bluepill, PC13 pin has a LED connected to it. Here what we gonna do is toggle that LED's state so that LED will blink. When the state is LOW, LED will be lighted up and when the state is HIGH, the LED will be off.

As you can see in the following picture, first open your .ioc file so that you can see the pinout view. Then click on PC13. You will get a list like in the following picture. Click on GPIO_Output, so that PC13 pin will be configured as an output pin.



After that your pinout view will look like the following (Pin has been turned to green colour).



Now you have to generate the code again as we have mentioned earlier in this post.

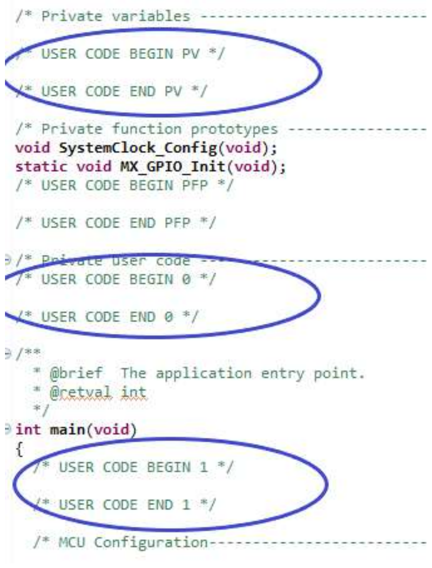
Before moving further there’s an important thing that you must know. Like in Arduino programming you cannot just put your pin name in CubeIDE when you are dealing with GPIO pins. Normally in a MCU, pins are grouped into several ports for the ease of use(Port is a collection of several pins).

If we look at this PC13 pin, here the port is “PORT C” and the “13” is the pin number.

Now let’s begin with coding.

- Find and double click on the main.c file to open it. [path - (Core -> Src -> main.c)]

- You must remember another thing when coding in CubeIDE. You can only write code segments in between the space where it says USER CODE BEGIN and USER CODE END.
  If you have written your code in other than these spaces, your written code will disappear when you generate your code again. See the following image if you are confused with this.



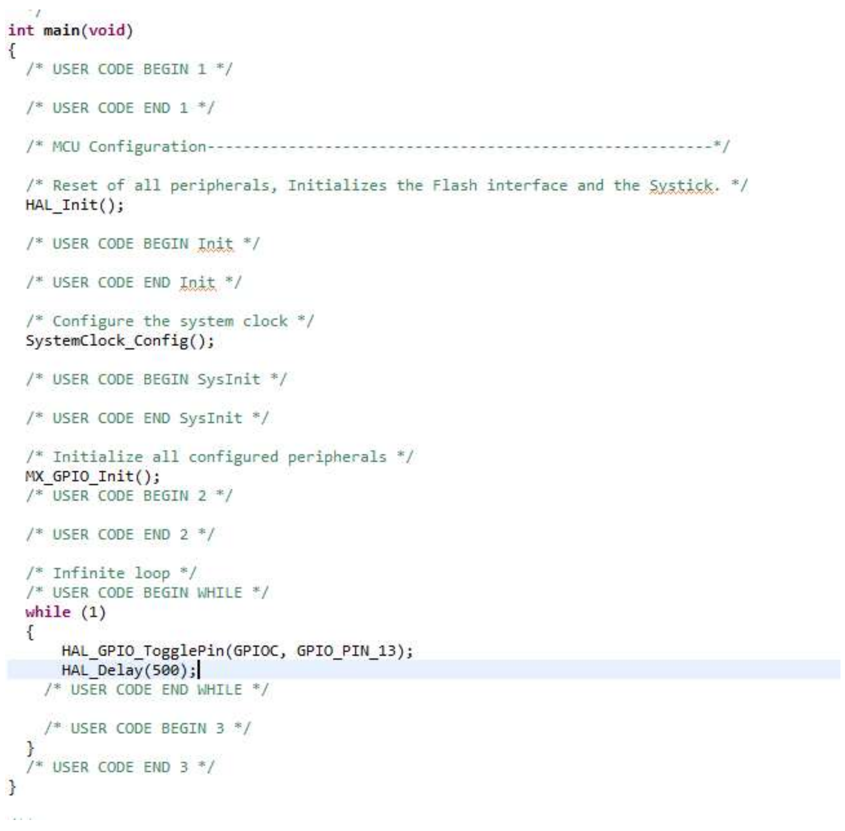
```
/* Private variables ------------------------
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes ---------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ------------------------
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration------------------------
```

- For this example, we will write our code segment in the while loop inside the main function. Write the following code sample in the CubeIDE.

  HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
  HAL_Delay(500);

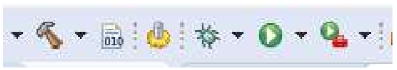
  Here, HAL_GPIO_TogglePin() function will change the state(HIGH or LOW) of the PC13 pin. This function needs two input parameters. They are the PORT and the PIN number of the GPIO(General Purpose Input Output) pin that you are using.

  HAL_Delay() function will delay your program in milliseconds depending on the input you give to it. In this example it is 500 milliseconds.



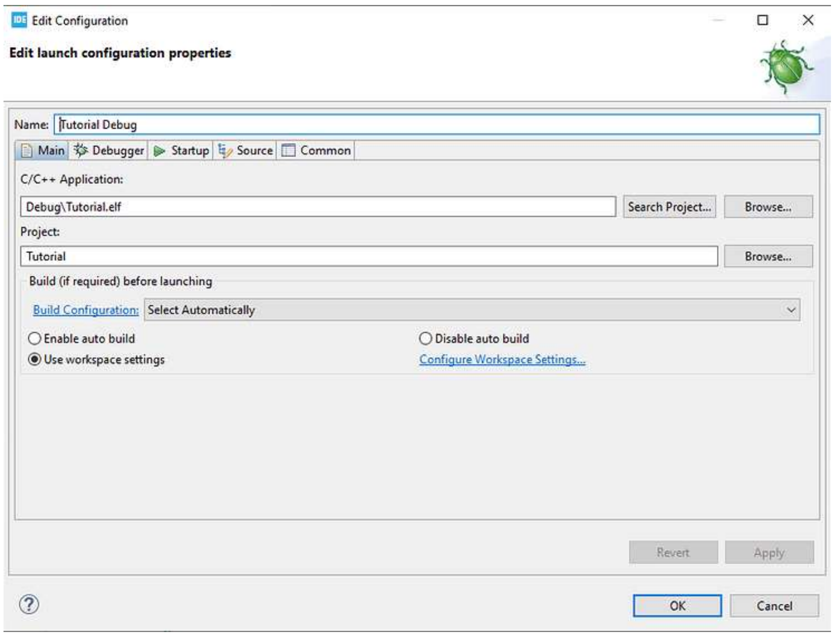
```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* 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();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13);
      HAL_Delay(500);
    /* USER CODE END WHILE */

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

- Now you have to build your program to check whether there's any error in your code. To do this just click on the “Hammer” icon in the toolbar or select [Projects]->[Build Project].



If there's no error in your code, you are ready to upload your code onto your MCU.

- Now connect your STM32 Board to the computer using a STLink.
- Click on the bug icon(debugging) in the toolbar to upload your code. The following window will appear then. For now, just click OK and your code will be uploaded and a new perspective will be opened.



Edit Configuration

Edit launch configuration properties

Name: Tutorial Debug

Main | Debugger | Startup | Source | Common

C/C++ Application:

Debug\Tutorial.elf

Project:

Tutorial

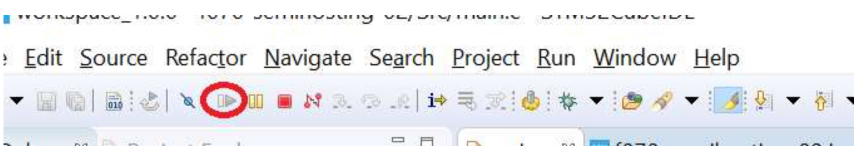
Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings Configure Workspace Settings...

- Click on the play button as outlined in the following picture, so that the code will start to run on your STM32 Board. And LED will start to blink. 😊


Edit Source Refactor Navigate Search Project Run Window Help

Congratulations, now you have completed your first program in CubeIDE.

Way more to go. Cheers! 💪