
ELEC 3300 – CubeMX

Department of Electronic and Computer Engineering
HKUST

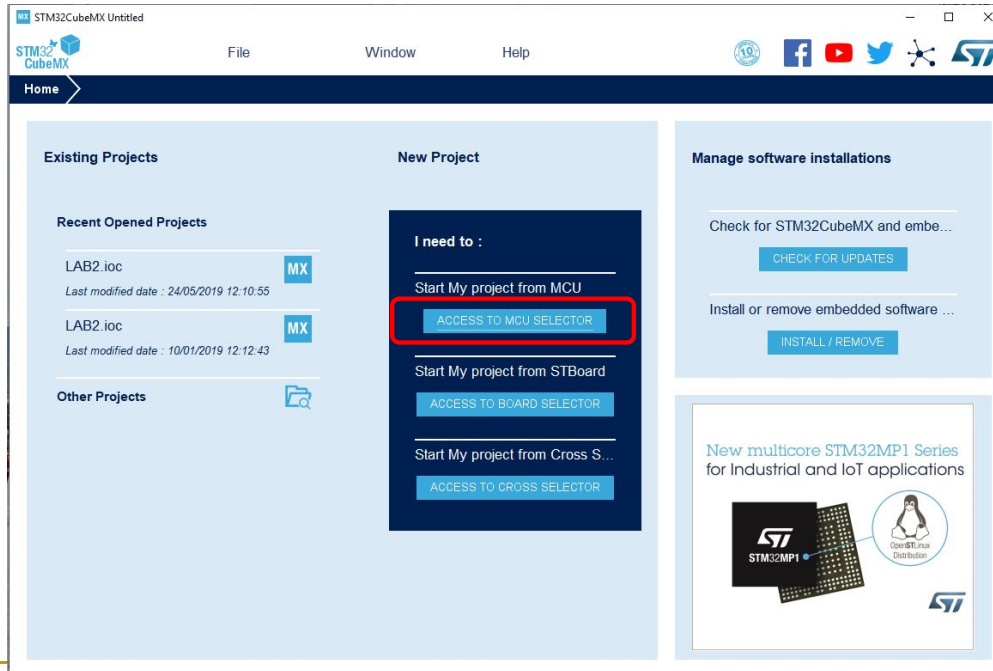
by WU Chi Hang 

About CubeMX

- CubeMX is a official software to generate the initialization code for all the platforms.
- <https://www.st.com/en/development-tools/stm32cubemx.html>
- In our LABs, you need to use it to generate the Project Template
- Please note that this tutorial is just only for your troubleshooting and simple demonstration of how can we use the CubeMX to connect to our MINI V3 Board.
- For complete integration and connection for each LAB, you need to check the Canvas for both resources of CubeMX and the Development Tools.

MCU Selector

- After installation, choose MCU Selector



Choose MCU

- Choose STM32F103VE, LQFP100, then Start Project

MCU/MPU Selector Board Selector Cross Selector

MCU/MPU Filters

Part Number Search

STM32F103VE

Core

Check/Uncheck All

☐ ARM Cortex-M3

Series

Line

Package

Check/Uncheck All

☐ LFBGA100

☐ LQFP100

Other

Price = 4.092

IO = 82

Eeprom = 0 (Bytes)

Flash = 512 (kBytes)

Ram = 64 (kBytes)

Freq = 72 (MHz)

Features Block Diagram Docs & Resources Datasheet Buy Start Project

STM32F103VE

Mainstream Performance line, ARM Cortex-M3 MCU with 512 Kbytes Flash, 72 MHz CPU, motor control, USB and CAN

ACTIVE Active Product is in mass production

Unit Price for 10kU (US\$) : 4.092

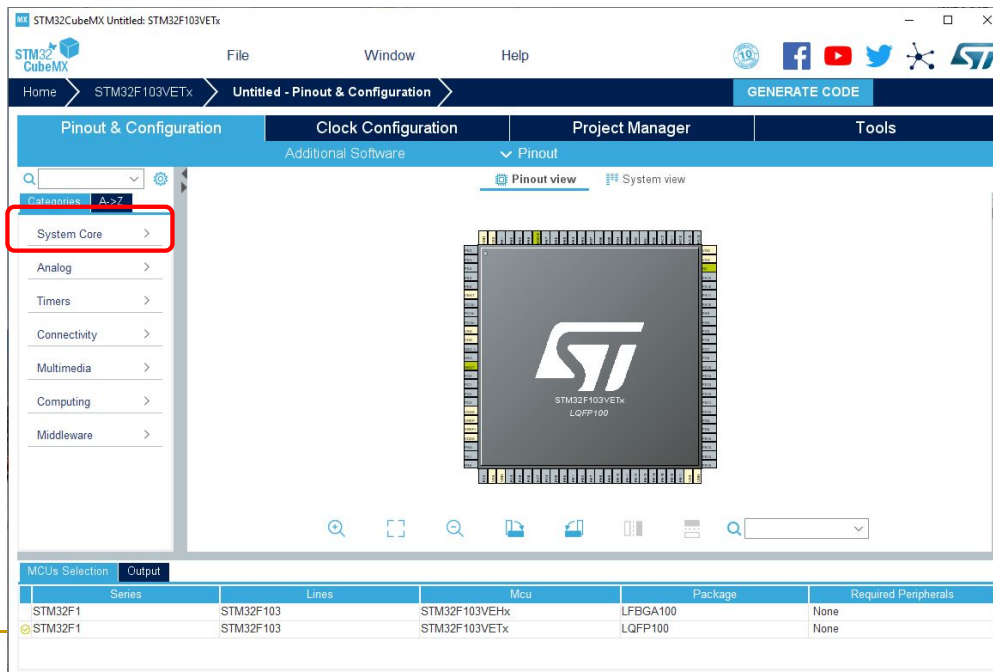
LQFP100

MCUs/MPUs List: 2 items

Part No.	Reference	Market	Unit Price for 10kU	Board	Package	Flash	RAM	IO	Freq	GPIOs	COR	USART	ADC	MAC	MPU	PSA	SWP	RT	PA
STM32F103	STM32F103VEHx	Active	4.092		LFBGA...	512 K...	64 KB...	82	72 ...	0.0	0	0	0	0	0	0	0	0	0
STM32F103	STM32F103VETx	Active	4.092		LQFP...	512 K...	64 KB...	82	72 ...	0.0	0	0	0	0	0	0	0	0	0

Set Clock

- You will go to this screen, first we need to set the clock, Expand System Core



Change Clock to Crystal

- Click RCC, enable the High Speed Clock and Low Speed Clock to
 - ❑ Crystal/Ceramic Resonator

STM32CubeMX Untitled: STM32F103VETx

File Window Help

Home STM32F103VETx Untitled - Pinout & Configuration

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

DMA

GPIO

IWDG

RCC

WWDG

Analog

Timers

Connectivity

Multimedia

Computing

Middleware

RCC Mode and Configuration

High Speed Clock (HSE) Crystal/Ceramic Resonator

Low Speed Clock (LSE) Crystal/Ceramic Resonator

☐ Master Clock Output

Configuration

Reset Configuration

Parameter Settings User Constants NVIC Settings **GPIO Settings**

Search Signals

Select Pins from table to configure them. Multiple selection is Allowed.

Pin Name	Signal on Pin	GPIO output	GPIO mode	GPIO Pull-u	Maximum s	User Label	Modified
OSC_IN	RCC_OSC_IN	n/a	n/a	n/a	n/a		<input type="checkbox"/>
OSC_OUT	RCC_OSC_OUT	n/a	n/a	n/a	n/a		<input type="checkbox"/>
PC14-OSC3	RCC_OSC3	n/a	n/a	n/a	n/a		<input type="checkbox"/>
PC15-OSC3	RCC_OSC3	n/a	n/a	n/a	n/a		<input type="checkbox"/>

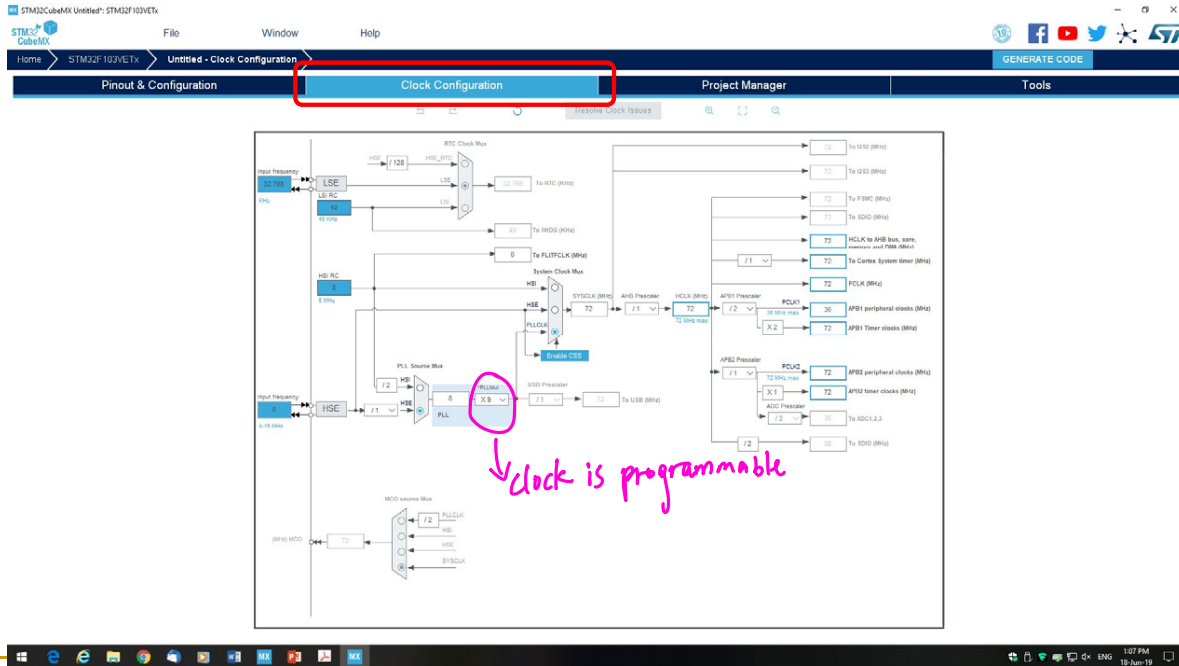
MCUs Selection Output

Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F103	STM32F103VETx	LFBGA100	None
STM32F1	STM32F103	STM32F103VETx	LQFP100	None

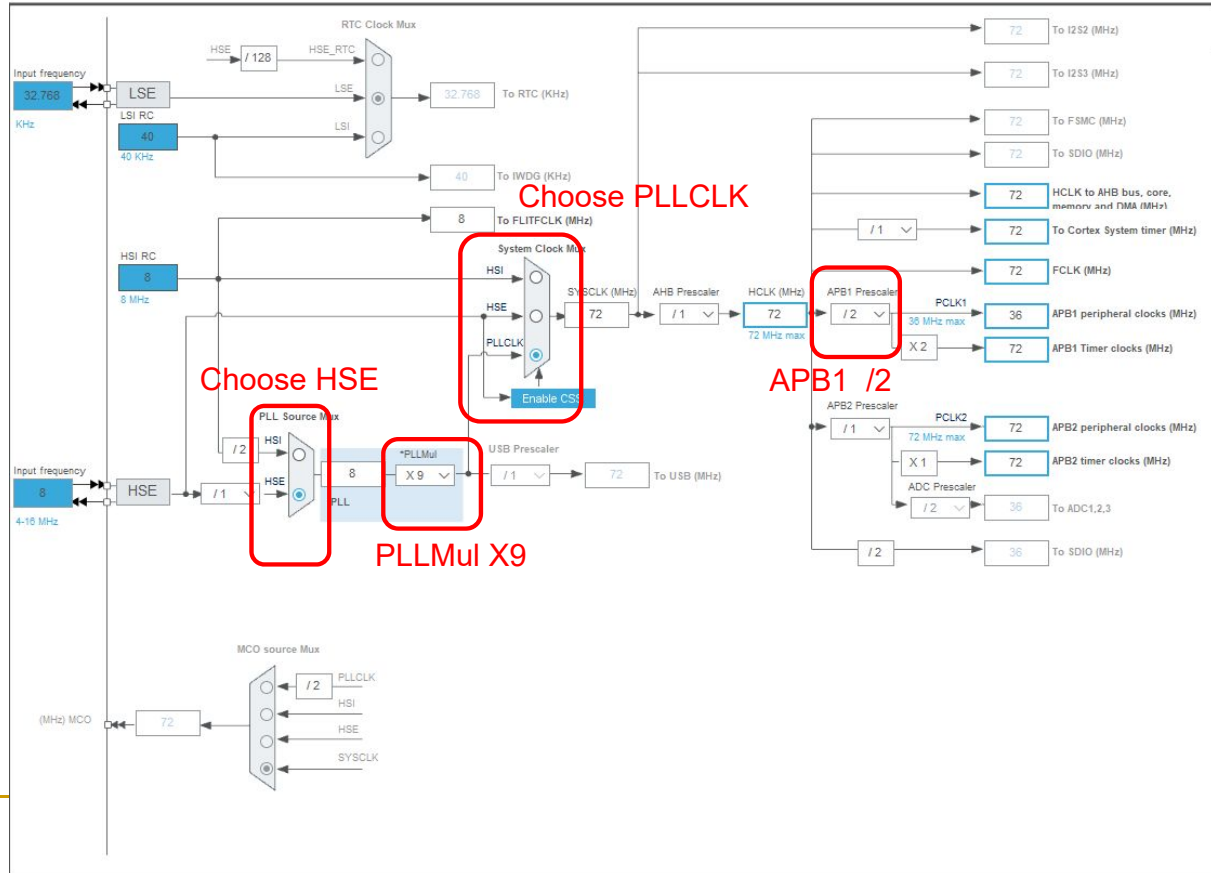
clock needs to input to the cortex m3 cpu

Clock Configuration

- Go to Clock Configuration

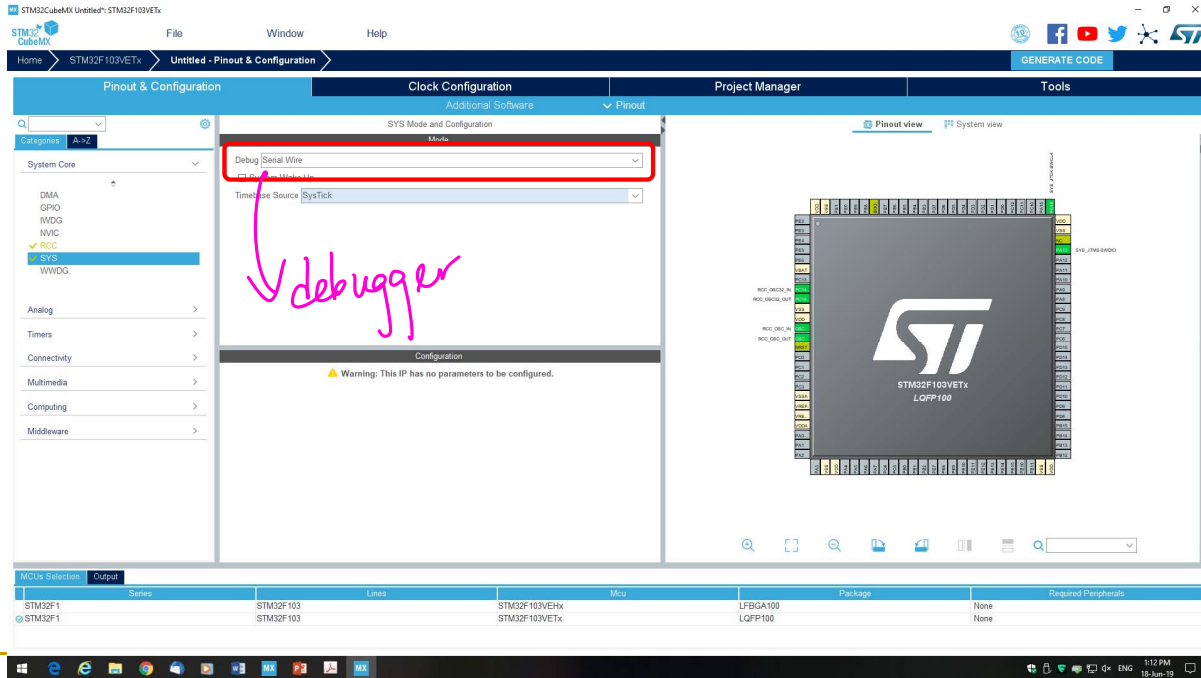


Clock Configuration



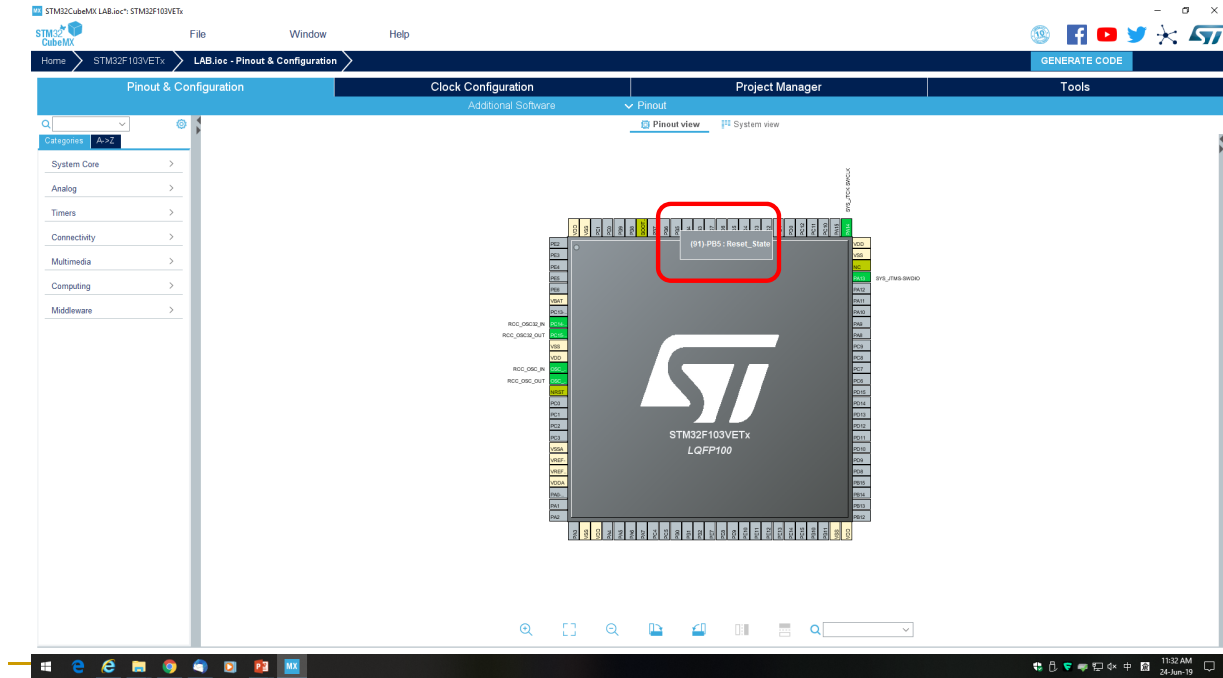
Communicate with Debugger

- Go to Pinout & Configuration, in SYS, Choose Serial Wire for Debug



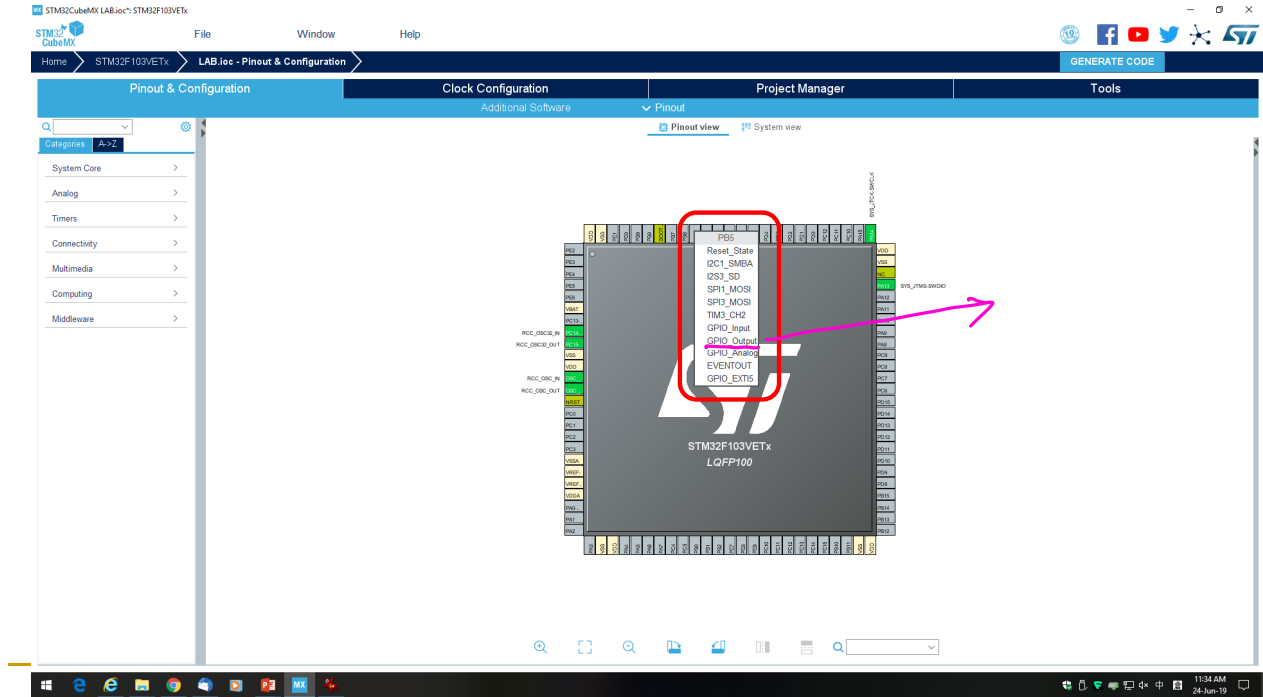
Define PB.5

- We now try to Define PB.5, go to PB.5



Define PB.5

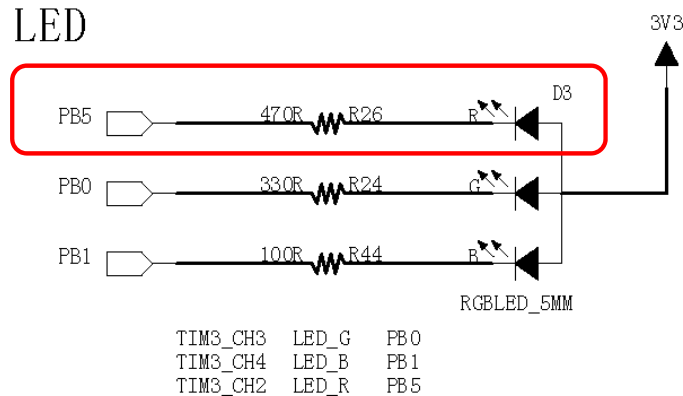
- Left Click, you will see a list



About PB.5

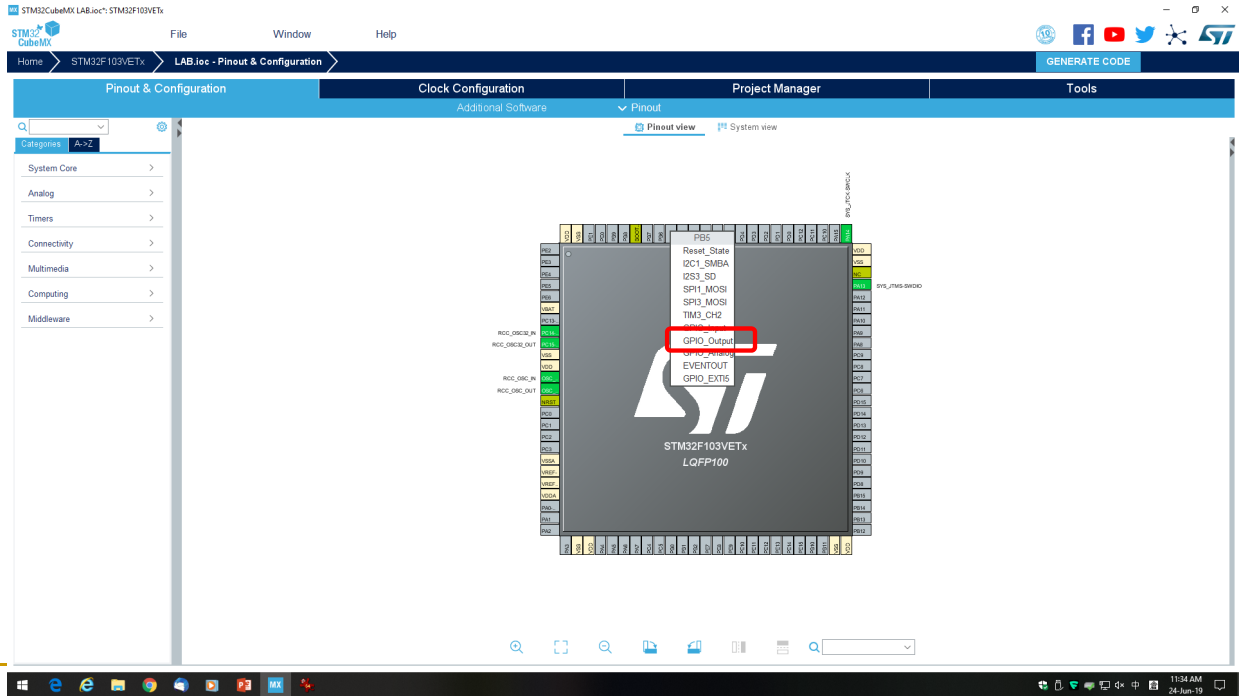
GPIO - output

- Since PB.5 in the development board is connected to the Red LED. We want to define it to be Output and with Push Pull, so that we can turn on and off.
- Note that the LED is LOW activated, we do not want it to turn on once we download the program, as a result, we need to initialize it to a HIGH state at the beginning



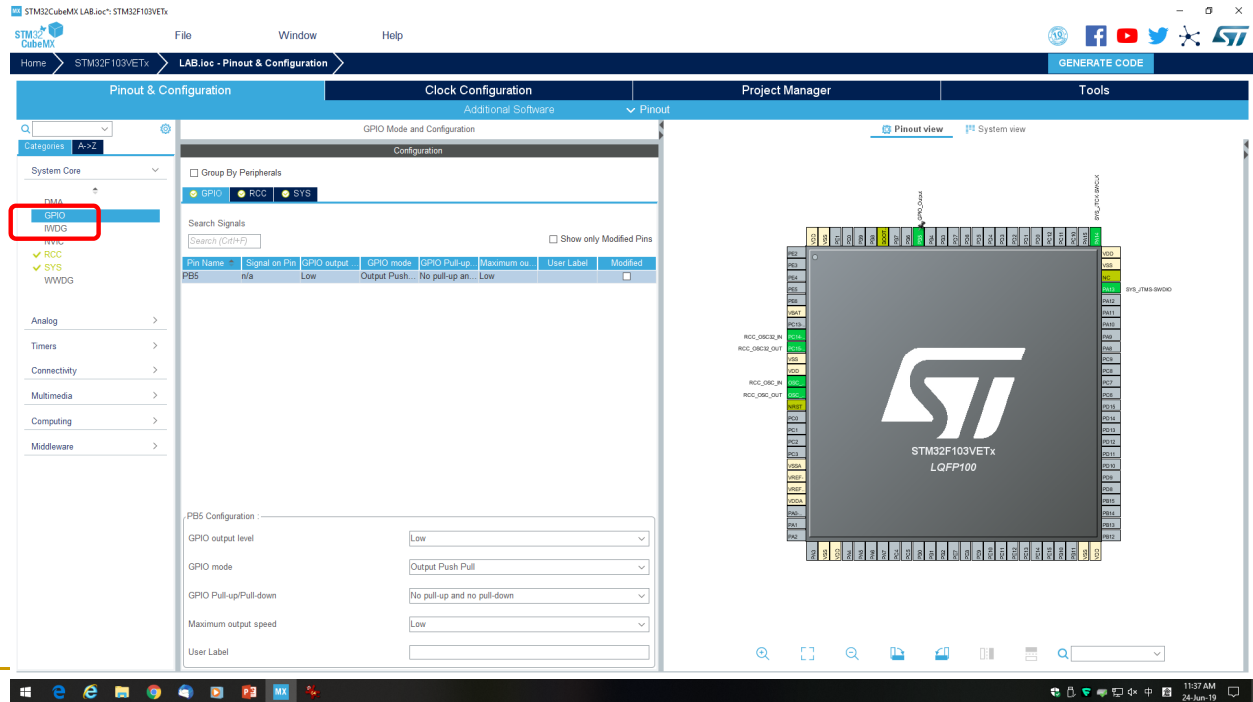
Define PB.5

■ Choose GPIO_Output



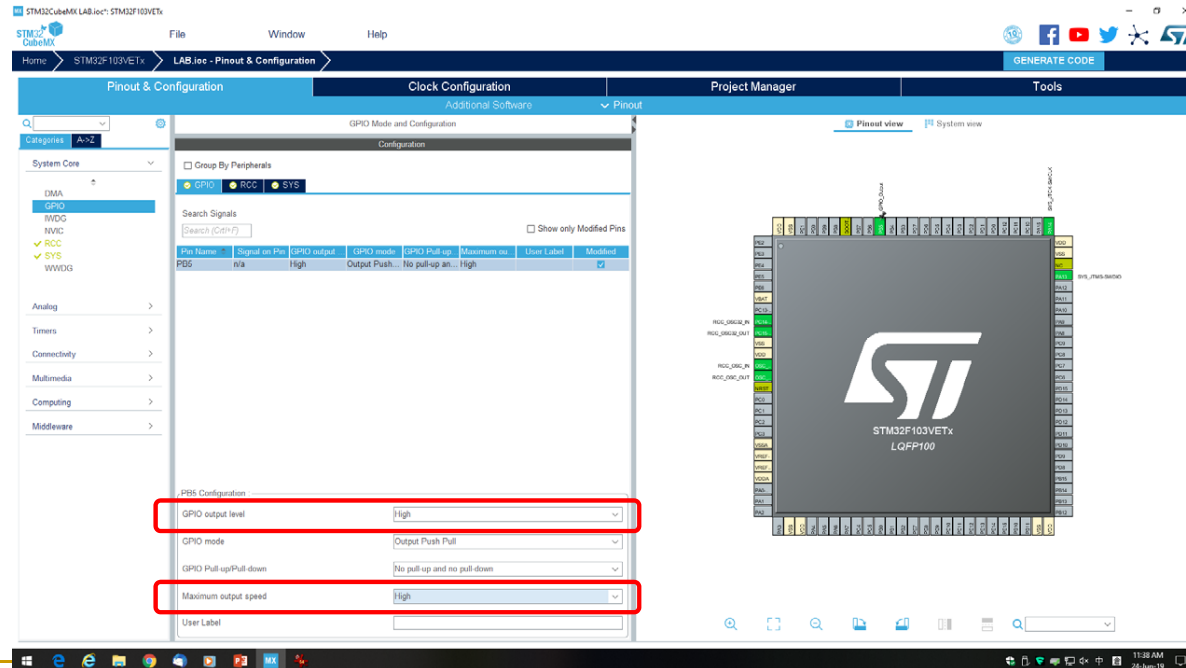
Define PB.5

- On left, click GPIO, you can then modify the parameters



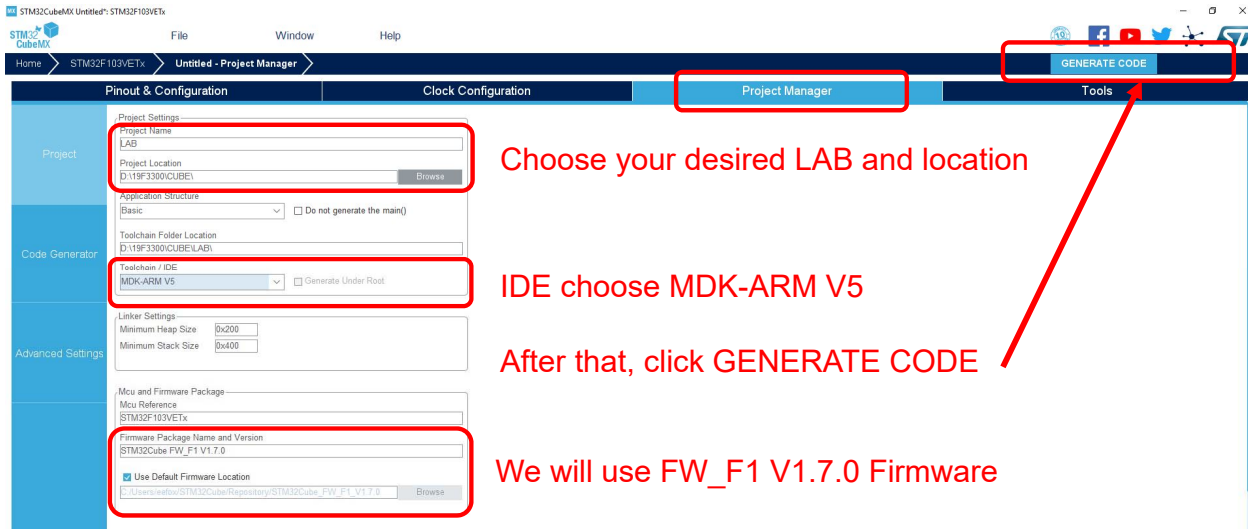
Define PB.5

- Modify Output Level → High, Maximum output speed → High



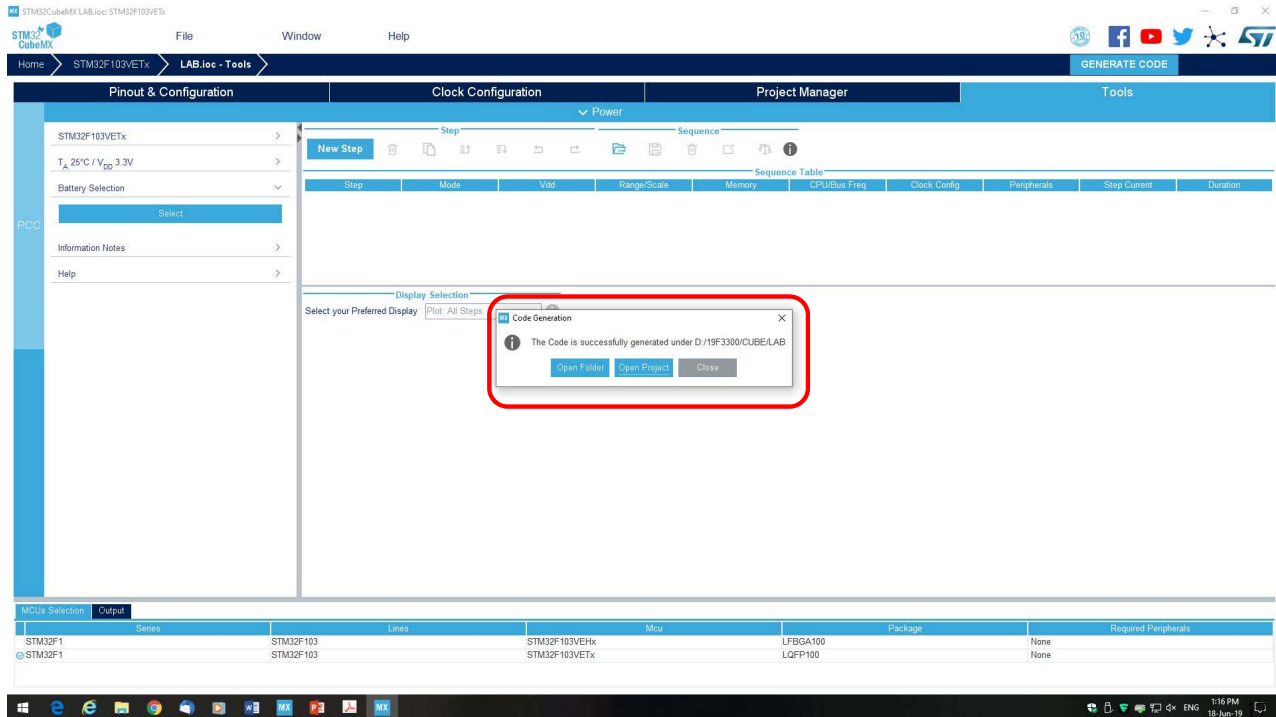
Generate Code

- Now, we can generate the Project Template, go to Project Manager
 - For the first time of generating code, user will be asked to confirm downloading "Firmware Pack STM32 Cube FW_F1 V1.7.0"



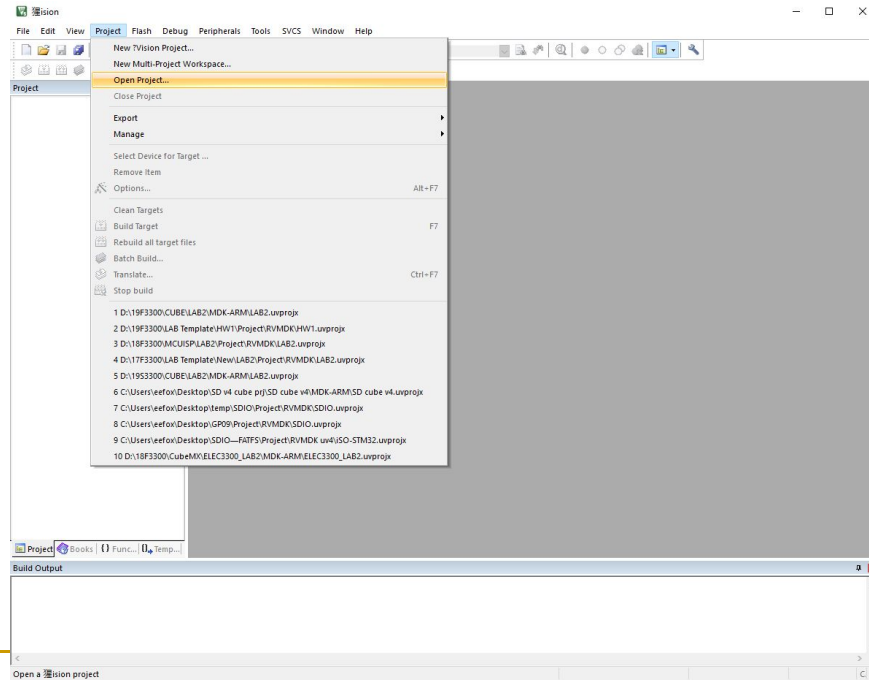
Generate Code

- If successful, it will create a folder



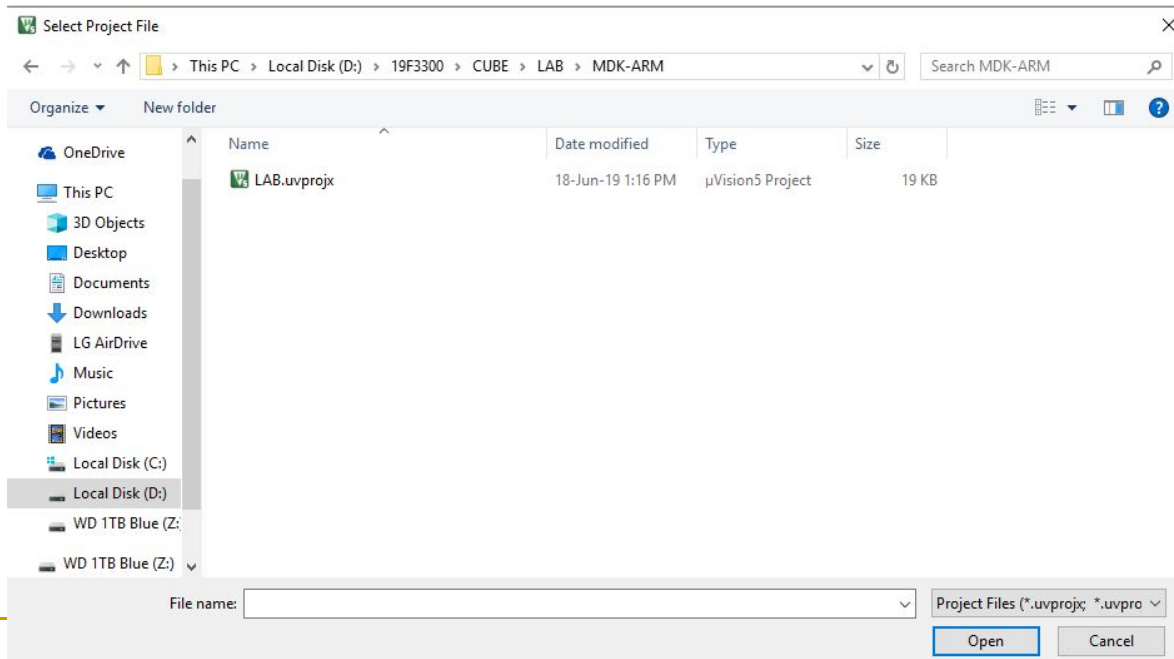
Open the Project

- Open Keil, Go Project → Open Project... Navigate to your folder



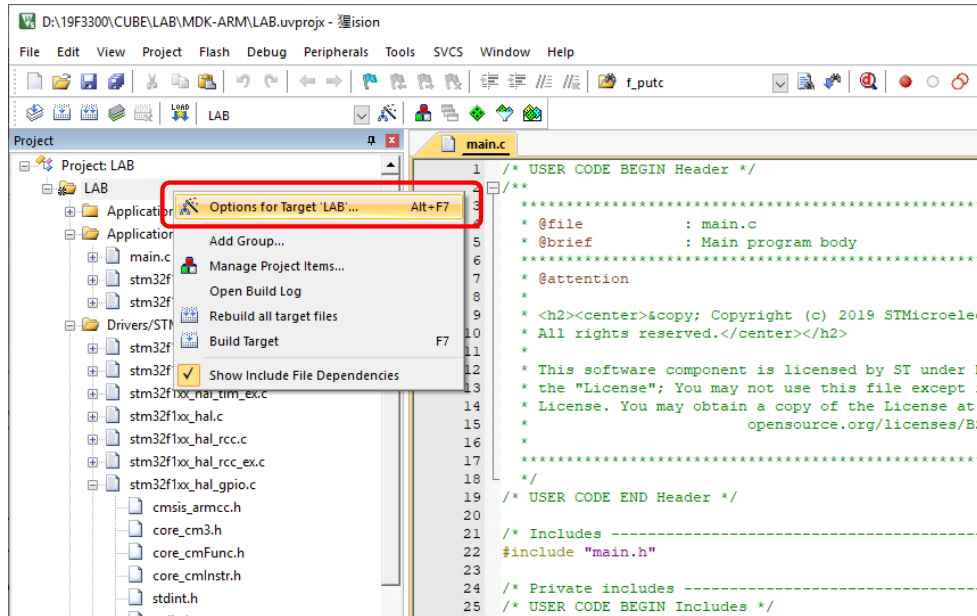
Open the Project

- There will be a MDK-ARM folder, under that, there will be a project file with uvprojx, in this example is LAB.uvprojx,



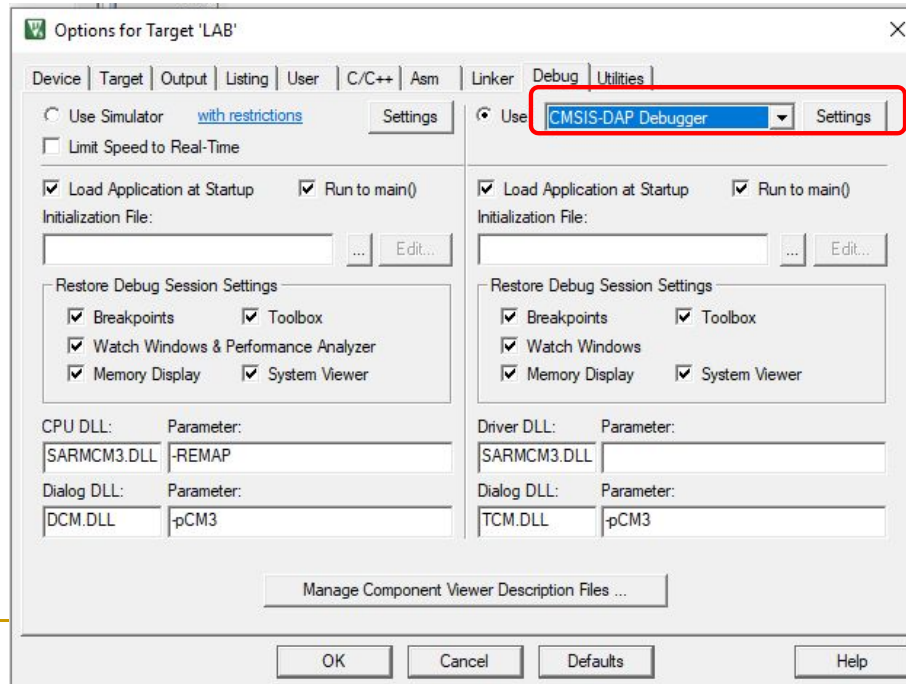
Modify Debugger Setting

- On the Project pane, right click on the Project and click on Options for Target



Modify Debugger Setting

- In Debug Window, change the debugger to CMSIS-DAP Debugger



Compile your Project

- Go to Project → Rebuild your Project File, there should be no error.

Add your Implementation Code

- Double Click the main.c, you should see a while(1) loop, as below

```
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    /* USER CODE END WHILE */  
  
    /* USER CODE BEGIN 3 */  
}  
/* USER CODE END 3 */  
}
```

Content inside Red will be retained
after re-generation of code

Content outside Red box will be
deleted after re-generation of code

Add your Implementation Code

- You can add the code between the USER CODE BEGIN and END

```
/* Infinite loop */
```

```
/* USER CODE BEGIN WHILE */
```

```
while (1)
```

```
{
```

```
    Code added here will be retained after re-generation of code
```

```
    because it is in between /* USER CODE BEGIN WHILE */ and /* USER CODE END WHILE */
```

```
/* USER CODE END WHILE */
```

```
    Code added here WILL BE DELETED after re-generation of code
```

```
    because it is in between /* USER CODE END WHILE */ and /* USER CODE BEGIN 3 */
```

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

```
    Code added here will be retained after re-generation of code
```

```
    because it is in between /* USER CODE BEGIN 3 */ and /* USER CODE END 3 */
```

```
}
```

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

```
}
```

Add your Implementation Code

- Add the code in Red, compile the code and run, you should see the Red LED Blinking.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET);
        HAL_Delay(1000);
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET);
        HAL_Delay(1000);
    }
    /* USER CODE END 3 */
}
```

END