



Title: Getting Start with STM32 using STM32CubeIDE and HAL

Prepared by:
PM Ir Dr Tee Kian Sek
Dr Chew Chang Choon
FKEE, UTHM

Date: April 2024

*A short course to promote competency skills
in electronics and embedded systems*

Table of Contents

1	Reminder.....	1
2	About the instructors.....	1
3	A Brief on ARM.....	2
3.1	ARM architecture family	2
3.2	Design and Licensing.....	2
3.3	Why popular?.....	2
3.4	Market Share in Processor	3
3.5	Competitors.....	3
4	Family of STM32 MCU.....	4
4.1	STM32 MCU Naming Convention.....	5
4.2	ST Reference.....	6
4.3	Competitors.....	6
5	Introduction To STM32 MCU Training.....	7
5.1	Key Elements (To Highlight).....	7
5.2	List of Market Parts	7
6	Knowing The Development Board	10
6.1	Schematic.....	10
6.2	Boot Configuration	12
6.3	The MCU – STM32F103C8T6.....	12
6.4	Pin Definitions	12
6.5	Device Features And Peripherals	13
6.6	ST Reference.....	14
7	Programming Software	15
7.1	STM32 – Hardware Abstract Layer (HAL).....	15
7.2	C Language	15
8	Installing STM32CubeIDE	16
8.1	First Glance	16
8.2	ST reference	17
8.3	Important skills	17
9	First Project - Create A New Project.....	18
9.1	Create A New Project.....	18
9.2	File Structure.....	19
9.3	Build – main.c.....	19
9.4	Pinout & Configuration.....	20
9.5	Run – Upload To MCU	22

9.6	Re-configure Pinout	23
10	Practices	24
10.1	GPIO- LED	24
10.1.1	Diagram.....	24
10.1.2	Practices	25
10.2	GPIO- LEDs Blink.....	27
10.2.1	Diagram.....	27
10.2.2	Practices	28
10.3	GPIO- LED-Buttons	30
10.3.1	Diagram.....	30
10.3.2	Practices	31
10.4	GPIO- EXTI.....	33
10.4.1	Diagram.....	33
10.4.2	Practices	33
10.5	UART in Polling Mode	35
10.5.1	Diagram.....	35
10.5.2	Practices	37
10.6	UART With Interrupt	40
10.6.1	Diagram.....	40
10.6.2	Practices	44
10.7	UART With DMA.....	47
10.7.1	Diagram.....	47
10.7.2	Practices	50
10.7.3	References on UART	53
11	STM32CubeIDE – Shortcut Keys	54
12	Tools.....	55
12.1	Serial App – CoolTerm	55
12.1.1	Self-test With UART-to-USB.....	56
12.1.2	Self-test With Two Units of UART-to-USB.....	56
12.2	Flashing Using UART Port.....	57
12.2.1	Procedures to Write.....	58
12.2.2	Procedures to Read/Read ChipInfo.....	60
12.3	ST Tools	60
13	STM32 ST-LINK Utility.....	61
13.1	Write HEX / Bin File to MCU	61
13.2	Read HEX / BIN File and Erase Full Chip	62
13.3	Why not STMCubeProgrammer?	62

1 Reminder

1. Before attending the short course, please install STM32CubeIDE on your personal computer. Refer to the installation guide for instructions. See **Installing STM32CubeIDE**.
2. The training kit, including MCU, ST-Link V2, parts, and accessories, is **not included** in the training fee. You will need to purchase the market parts yourself as listed in the List of Market Parts.

2 About the instructors



PM Ir Dr Tee Kian Sek

<https://community.uthm.edu.my/staff/people/tee>



Dr Chew Chang Choon

<https://community.uthm.edu.my/chewcc>

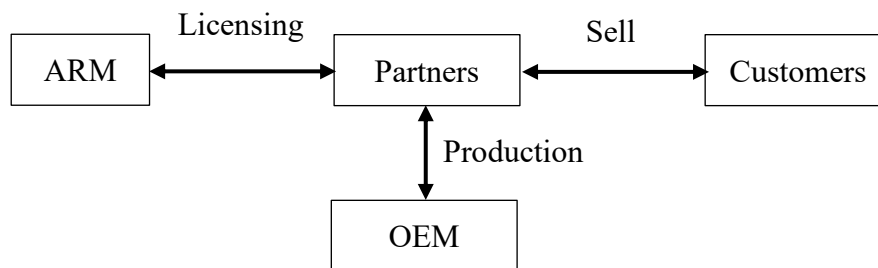
3 A Brief on ARM

3.1 ARM architecture family

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

ARMv1, ARMv2, ARMv3... ARMv9

3.2 Design and Licensing



Partners:

1. Broadcom
2. Apple
3. ST
4. ARM licenses IP to over 1,000 global partners (including Samsung, Apple, Microsoft).

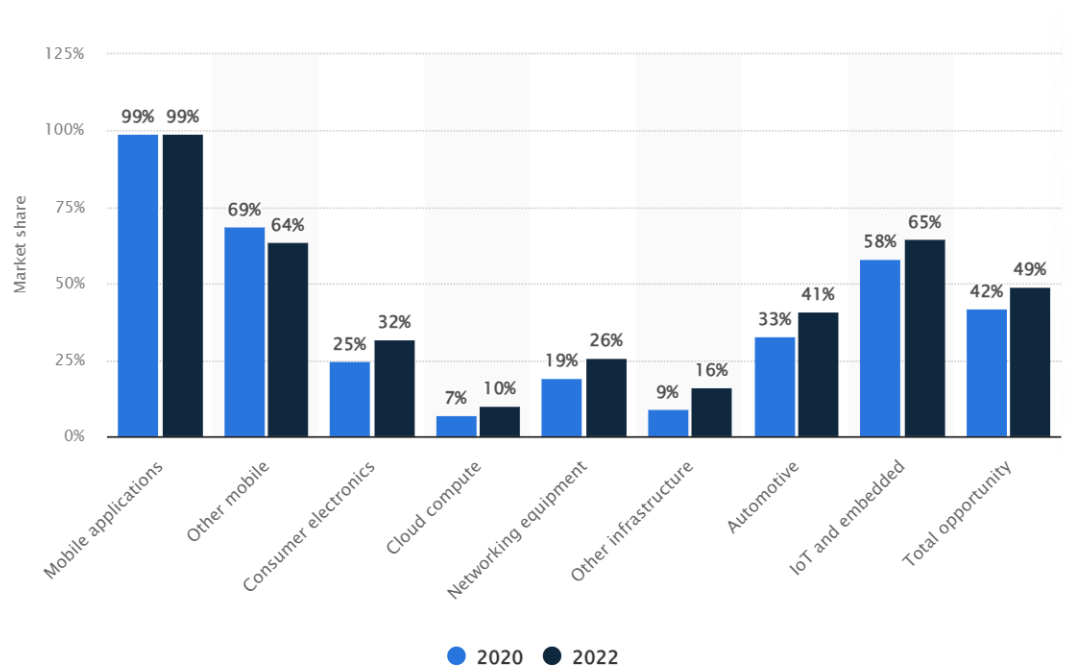
<https://www.arm.com/partners/catalog/results#sort=date%20descending&numberOfResults=12>

3.3 Why popular?

1. Low energy consumption, low cost, high performance
2. Support 16/32 instruction sets
3. Many partners
4. Rich ecosystem
5. Many more reasons...

https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers.html

3.4 Market Share in Processor



Source: <https://www.statista.com/statistics/1132112/arm-market-share-targets/>

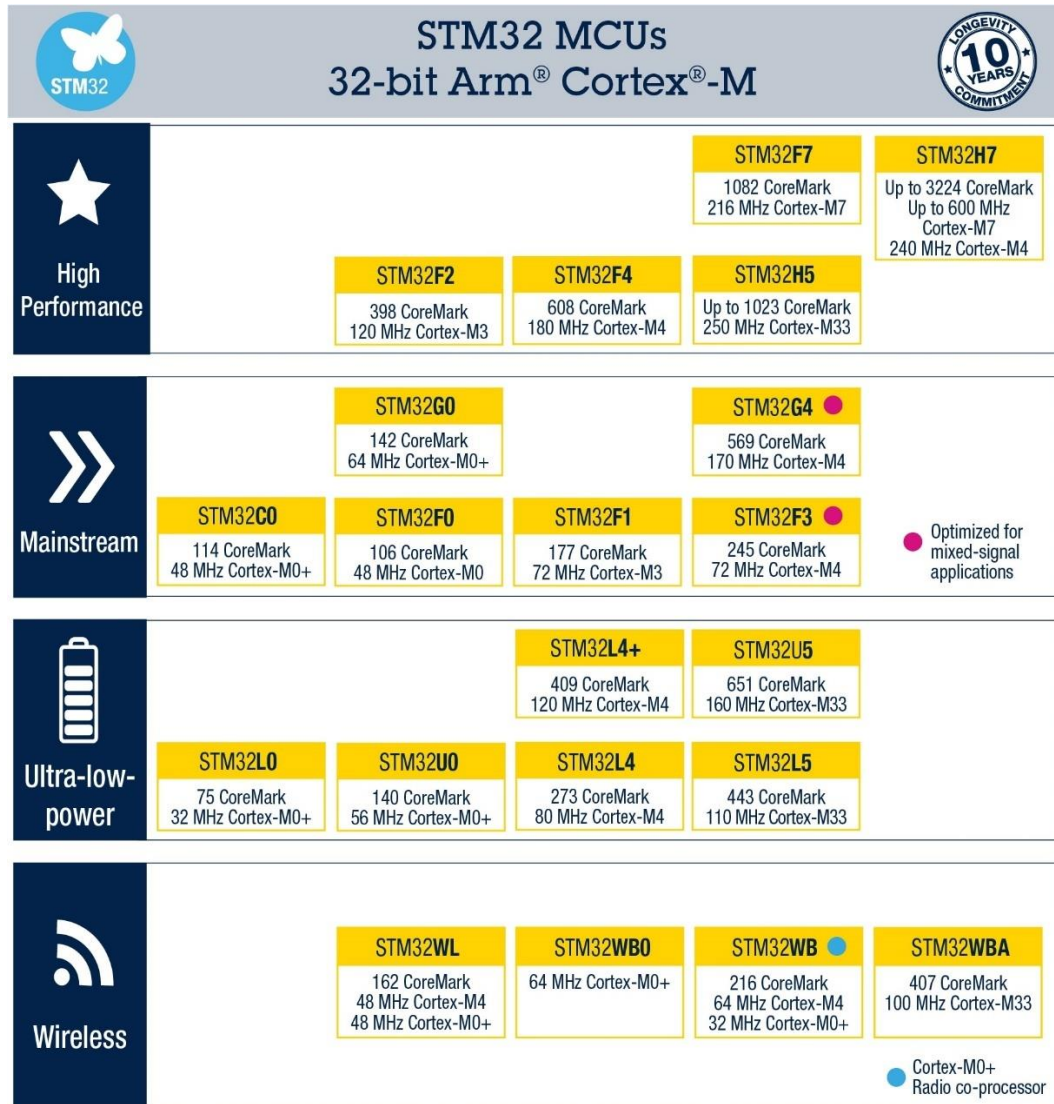
With over 230 billion ARM chips produced, as of 2022, ARM is the most widely used family of instruction set architectures.

3.5 Competitors

1. Nvidia
2. AMD
3. Rambus
4. MediaTek
5. Graphcore
6. Cavium Networks
7. Many more...

4 Family of STM32 MCU

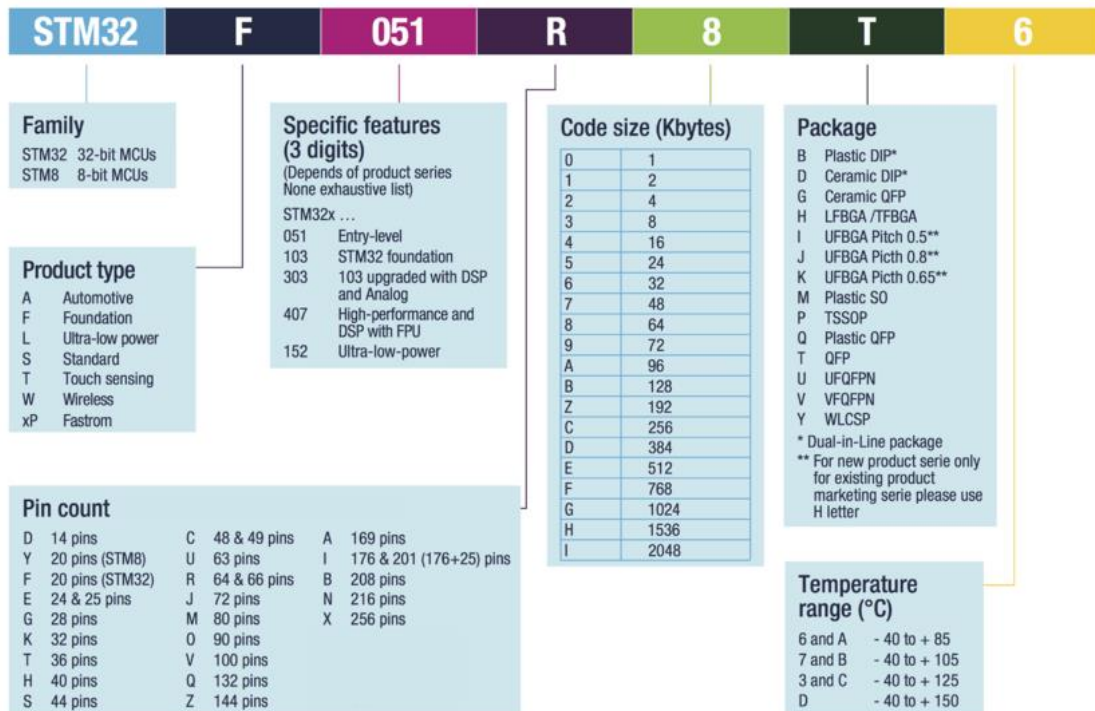
ST has a variety of products to meet many industrial applications. Microcontroller is a part in the product list. Visit ST website.



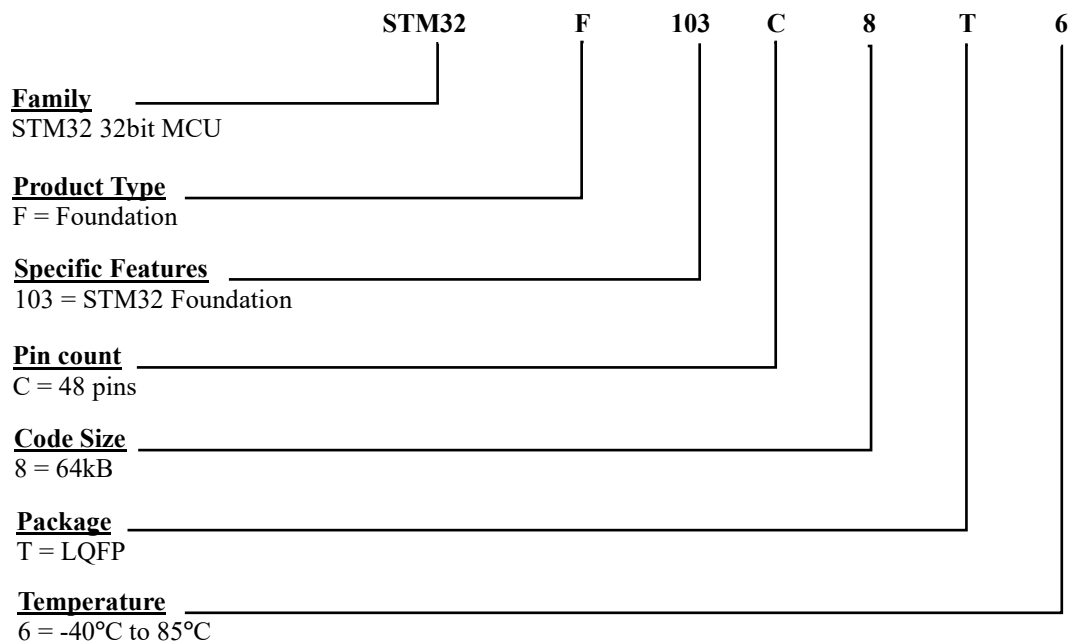
Source: <https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>

- STM32 is a 32-bit microcontroller developed by STCompany based on the ARM Cortex-M core (M0, M0+, M3, M4, M7, M33)
- STM32 is often used in embedded fields, such as smart cars, unmanned machines, robots, wireless communications, internet of things, industrial control, entertainment electronics, etc.
- STM32 is powerful, excellent performance, rich in resources, low power consumption, is a classic embedded microcontroller.

4.1 STM32 MCU Naming Convention



For example, STM32F103C8T6 (it is the MCU deployed in this training!)



4.2 ST Reference

Learners are advised to consult official documentation such as datasheets, instruction manuals, and notes provided by ST in PDF format. You may click the link and search for PDF:

<https://www.st.com/en/microcontrollers-microprocessors/stm32f103/documentation.html>

The MCU is STM32F103C8T6 (We recommended a datasheet and a reference manual.)

1. Datasheet
 - a. Medium-density performance line Arm
 - b. STM32F103x8, STM32F103xB
2. Reference Manual - RM0008, for STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs
3. Others.

More readings:

Comprehensive explanation –

<https://stm32-base.org/guides/getting-started.html>

https://stm32world.com/wiki/Blue_Pill

<https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html>

Highlighted information:

1. Based on ARM architecture.
2. Feature
3. Provided STM - https://www.st.com/content/st_com/en/stm32-mcu-developer-zone/boards-and-hardware-tools.html
4. Provided in market
5. Programming Language – C language

4.3 Competitors

1. Microchips (Atmel was acquired in 2016)
2. NXP Semiconductors
3. Infineon
4. Renesas Electronics
5. Texas Instruments
6. Many more...

5 Introduction To STM32 MCU Training

To begin, we need to ensure that we have the everything in place. This includes:

1. Hardware – MCU, programmer and peripheral devices – minimum system.
2. Software – Integrated Development Environment (IDE)
3. DC power source – 5VDC or 3.3VDC – we are tapping the source form a personnal computer.
4. Breadboard and jumper wires
5. A personal computer









5.1 Key Elements (To Highlight)

1. STM32 MCU and development board
 - a. Recommended here – STM32F103C8T6
 - b. **Do not confuse with STMF03C6T6**
2. Programmer
 - a. Recommended here – ST-Link V2
3. Software – Recommended here - STM32CubeIDE
 - a. Download for free
 - b. https://www.st.com/content/st_com/en/stm32cubeide.html
 - c. Install – See next.



5.2 List of Market Parts

No	Component	Qty	Images
1	MCU Training Board STM32F103C8T6, <i>Micro USB</i> a.k.a: Bluepill IF financially allowed, buy 2 units.	1	
2	ST-LINK V2	1	

3	micro USB to USB cable Programming cable (Optional)	1	
4	KY-009 RGB Full Color LED SMD Module for Arduino	1	
5	Resistor 0.25 W, any value from 220~470 Ohm (in pack of 10 pcs) – any value from this range	1 pack	
5	Resistor 0.25 W, any value from 2k ~ 10k Ohm (in pack of 10 pcs) – any value from this range	1 pack	
6	Momentary pushbutton, 4 pins. If financially allowed, buy a few units.	2	
7	breadboard, MB102	1	
8	Male to Male 20mm length, (normally in bundle of 40pcs or less) Dupont Jumper Wire	1 set	
9	Male to Female 20mm length, (normally in bundle of 40pcs or less) Dupont Jumper Wire	1 set	
10	USB to UART Converter CH340 or CP2102 (either one would work) If financially allowed, buy 2 units.	1	
12	OLED 0.96", 128x64 I2C If financially allowed, buy 2 units.	1	
13	A laptop OS - Window 10 or 11	1	

Note:

1. All self-purchased market parts belong to the participants.
2. These market components are currently available in the market.
3. The cost may vary depending on personal purchases.
4. If financially allowed, consider purchasing spare units, such as MCU, USB to UART Converter and OLED.
5. Not a must but it would be much convenient if you could equip your toolbox with a digital multimeter, wire cutter, screw drivers, tweezer, etc. All within your personal budget.

6 Knowing The Development Board

The development board is a minimum system with STM32F103C8T6, commonly called "BluePill" in the market.

STM32F103 microcontrollers use the Cortex-M3 core, with a maximum CPU speed of 72 MHz. The portfolio covers from 16 Kbytes to 1 Mbyte of Flash with motor control peripherals, USB full-speed interface and CAN.

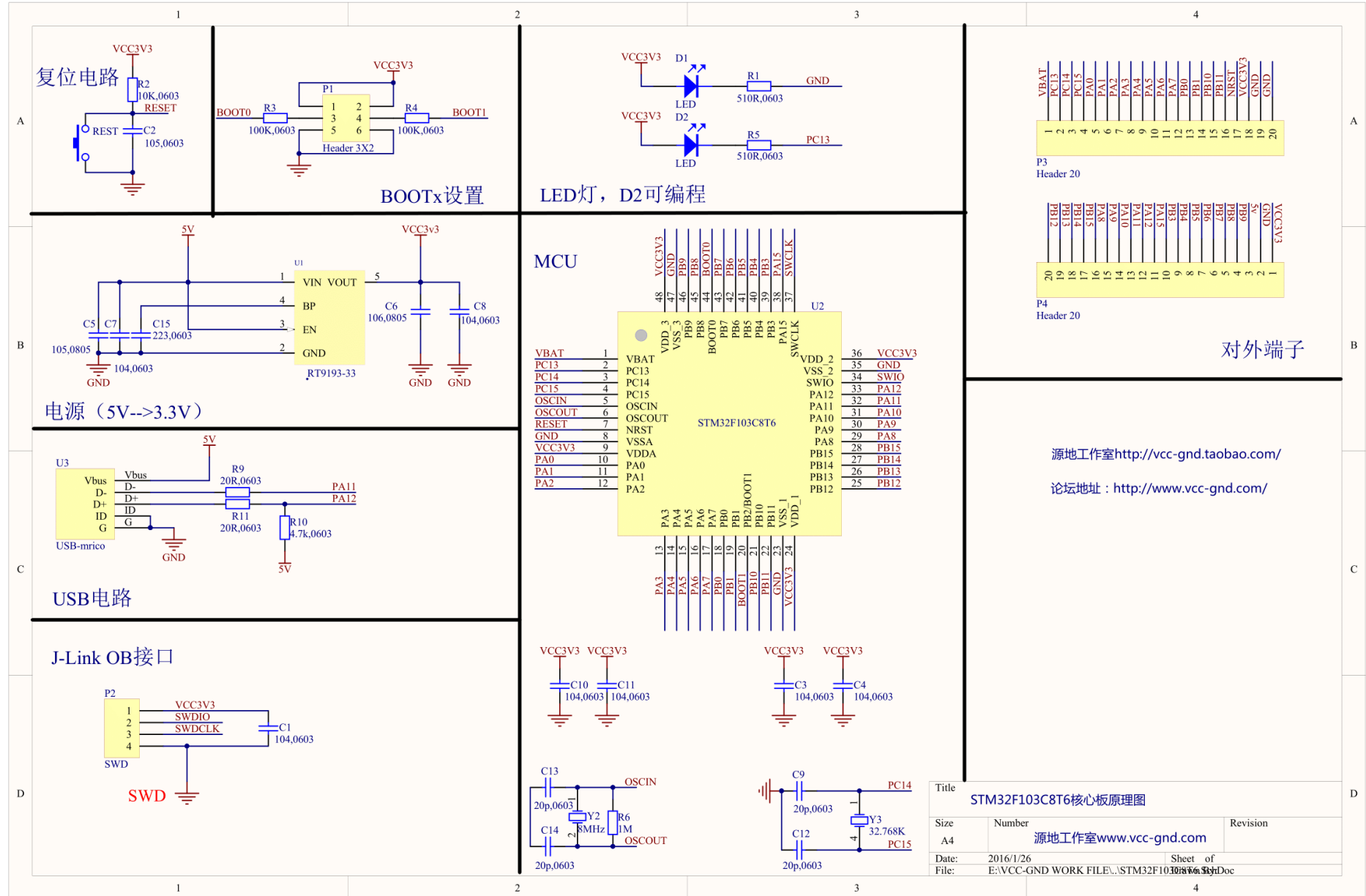


Source: <https://www.st.com/en/microcontrollers-microprocessors/stm32f103.html>

6.1 Schematic

See attachment.

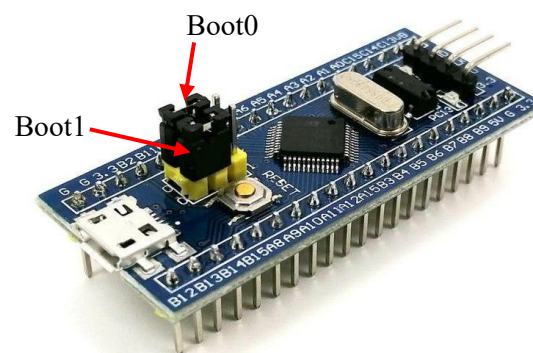
original-schematic-STM32F103C8T6-Blue_Pill.pdf



6.2 Boot Configuration

Table 9. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space



6.3 The MCU – STM32F103C8T6

1. Family: STM32F1
2. Core: Arm® 32-bit Cortex®-M3 CPU core
3. Frequency: Max. 72MHz
4. Memory: 20K (SRAM) , 64K (Flash)
5. Supply: 2.0~3.6V (Standard 3.3V)
6. Package: LQFP48

Refer to the official datasheet (see attachment)

STM32F103x8 / STM32F103xB - Medium-density performance line Arm®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. Interfaces

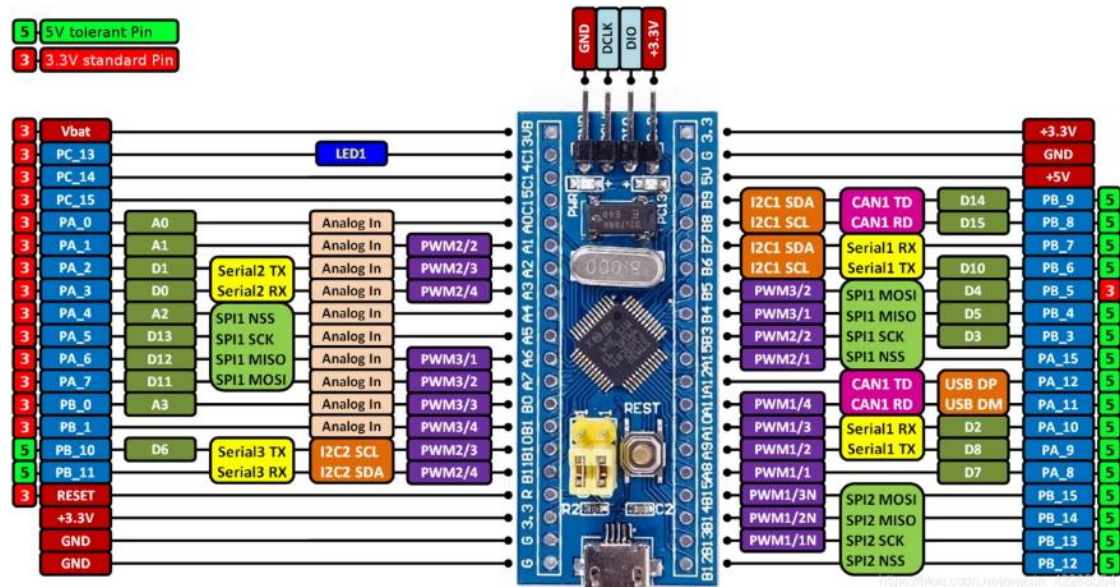
6.4 Pin Definitions

Alternatively re-arrange in excel spreadsheet. (See attachment)

[*STM32F103C8T6 -pin definition.xlsx*](#)

We shall exam this document often.

Alternatively, for quick review and assignment, we could refer to this pin assignment below.

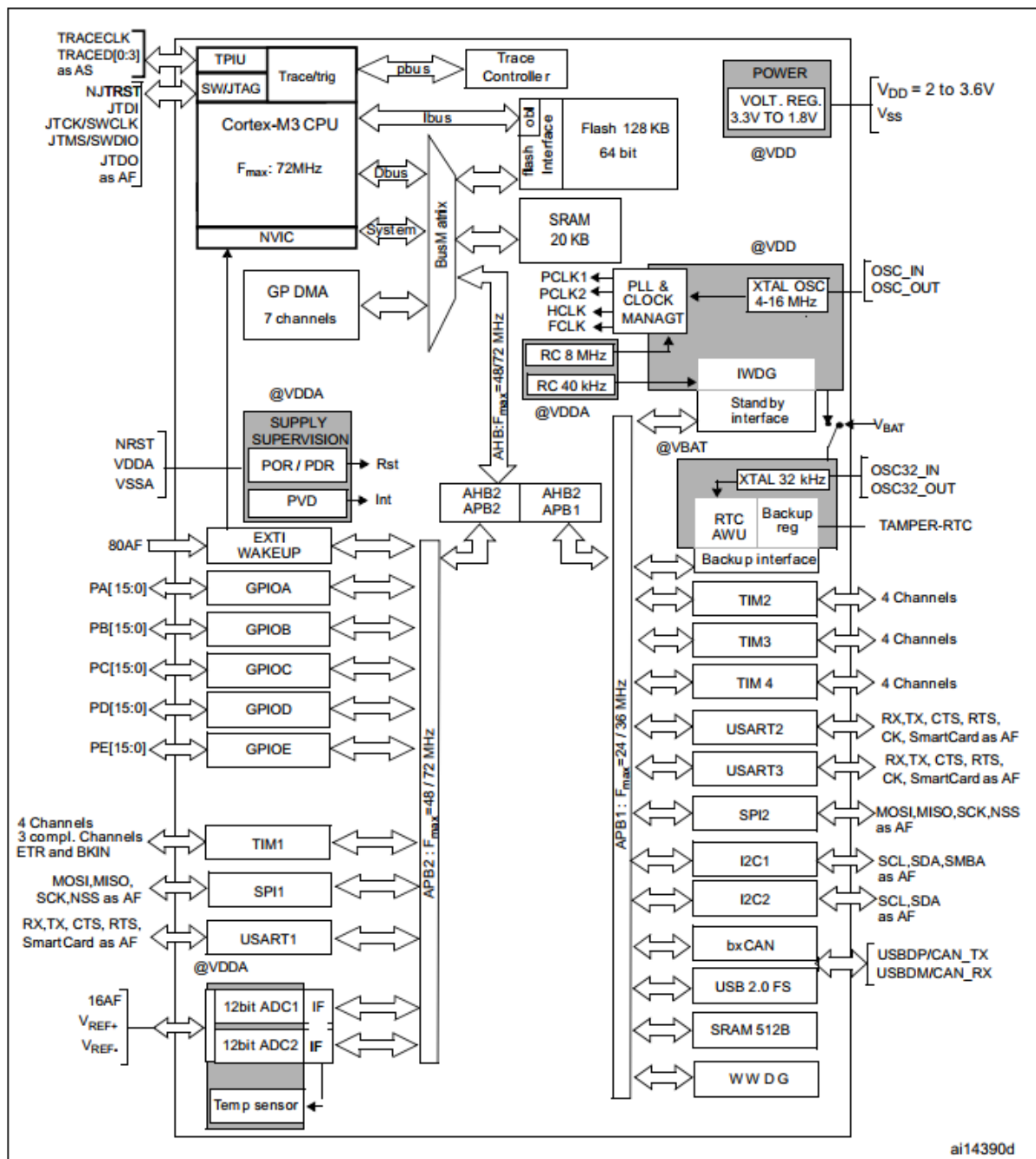


Source: <https://reversepcb.com/stm32f103c8t6/>

6.5 Device Features And Peripherals

Term	Description	Term	Description
NVIC	Nested Vectored Interrupt Controller	CAN	CAN Comm.
SysTick	the Cortex® System Timer	USB	USB Comm.
RCC	Reset and Clock Control	RTC	Real-time clock
GPIO	General-purpose I/O	CRC	Cyclic Redundancy Check
AFIO	Alternate-function I/O	PWR	Power Control
EXTI	External interrupt/event controller	BKP	Backup registers
TIM	Timer	IWDG	Independent watchdog
ADC	Analog-to-Digital Converter	WWDG	Window watchdog
DMA	Direct memory access	DAC	Digital-to-analog converter
USART	USART Comm.	SDIO	SD Interface
I2C	I2C Comm.	FSMC	Flexible static memory controller
SPI	SPI Comm.	USB OTG	USB

Refer to the manual for detail description. No worry if these terms appear to be scary.



(Source: Datasheet)

6.6 ST Reference

At ST official website, search for: STM32F103

<https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html#documentation>

1. Datasheet
2. Reference Manuals

7 Programming Software

There are a few Integrated Development Environment (IDEs) available in the market to code STM32 and its family. Some of them are paid services yet some are open-source. Among them:

1. **STM32CubeIDE** - free - explored in this training.
2. Keil MDK - paid service
3. IAR Embedded Workbench – paid service
4. Arduino IDE - free
5. PlatformIO - free
6. Matlab - Hardware support needed.
7. Etc.

7.1 STM32 – Hardware Abstract Layer (HAL)

To code an MCU, there are a few options:

1. Bare metal programming
 - a. Call the registers directly and manually
2. Standard peripheral library
 - a. Provided by ST
3. **Hardware Abstract Layer (HAL)**
 - a. **ST provides HAL for its MCU family.**
 - b. **This is implemented in this short course.**

The STM32 Hardware Abstraction Layer (HAL) provides APIs to connect with user applications, libraries, and stacks, making it easier to code embedded systems. HAL will make development faster because it's easy to use. This is especially true for first prototypes or examples where you do not need extensive tests but you need something to show quickly. HAL usually makes code easily portable within the same brand/manufacture.

Further readings:

1. <https://www.linkedin.com/pulse/bare-metal-vs-hal-unleashing-power-embedded-systems-daniel-oluwole/>
2. <https://embeddedthere.com/understanding-stm32-hal-library-fundamentals/>

7.2 C Language

STM32CubeIDE deploy C-language for HAL and coding.

Having a basic understanding of the C Language can be very useful.

Some C elements:

Variables, Data type, Operators, Loops, Struct, Pointer, Function, type cast, etc.

Note: C not equal to C++ however C could be implemented in C++.

8 Installing STM32CubeIDE

1. It is free and can be downloaded from (or google it!)
https://www.st.com/content/st_com/en/stm32-mcu-developer-zone/software-development-tools.html
2. Choose Windows version (I use Windows)
3. Follow the instruction. Register your name and email. Once registered, check your email. Click the link to download the software.
4. Unzip and then install STM32CubeIDE.
5. Official guide by STM32
 - a. Getting start video – by STM32 website
 - b. https://www.st.com/content/st_com/en/stm32cubeide.html

Friendly suggestion:

1. You could simply click ‘ok’ and accept all options along the installation.
2. Alternatively, when being prompted to enter the local folder for workspace, Suggestion: create a specific folder in your computer as the workspace for all projects.

Example:

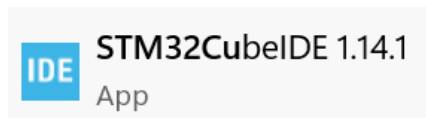
D:\STM32CubeIDE\workspace\

Because C-drive is app and os. D drive is my working drive.

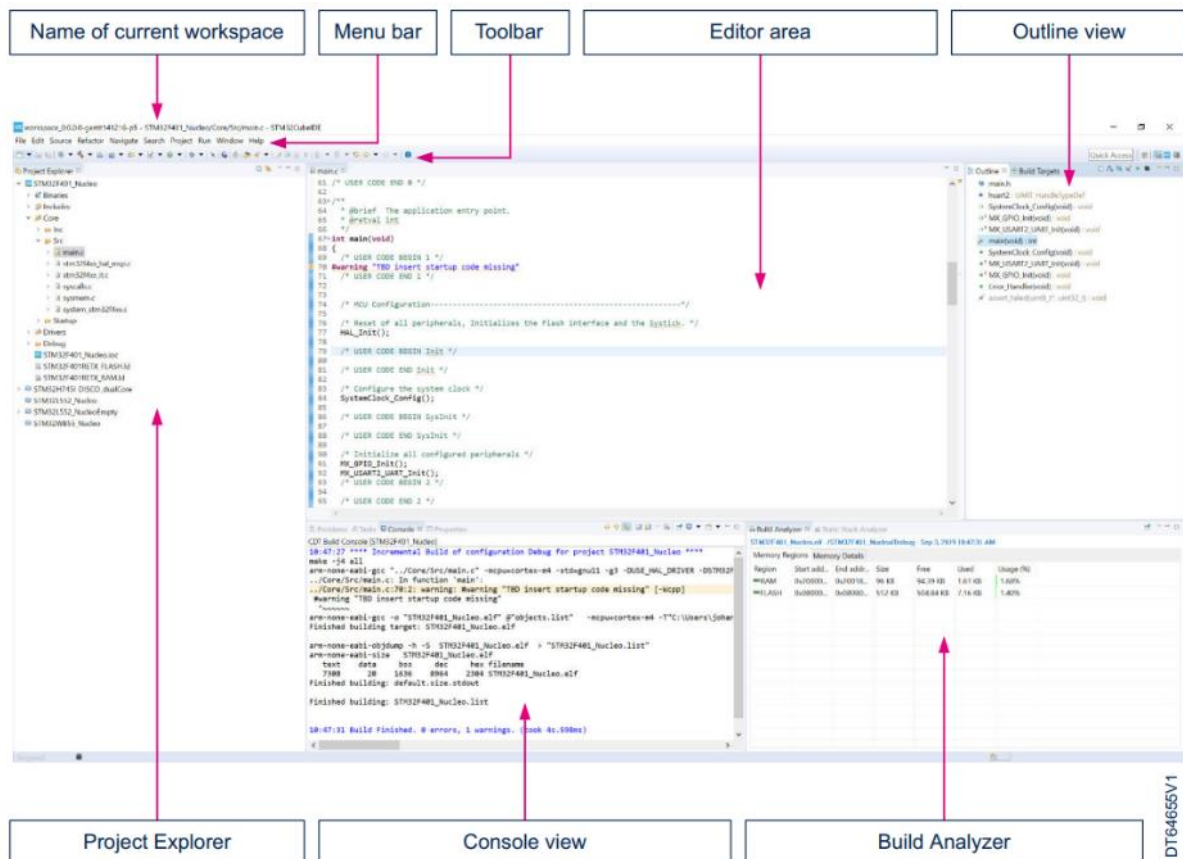
8.1 First Glance

We will explore this IDE in this training.

Once installed, search for “STM32CubeIDE 1.14.1”. (Your version might be different!). Click and run.



The IDE should run and similar to the image below. We shall go through the IDE as we practice.



Source: UM2609 – User Manual - STM32CubeIDE user guide

8.2 ST reference

<https://wiki.st.com/stm32mcu/wiki/Category:STM32CubeIDE>

No worry about it. We would know the operations as soon as we create a new project.

8.3 Important skills

Every often a program will do these:

1. Clone a project – either as the progression of coding or importing from a shared project.

See the link:

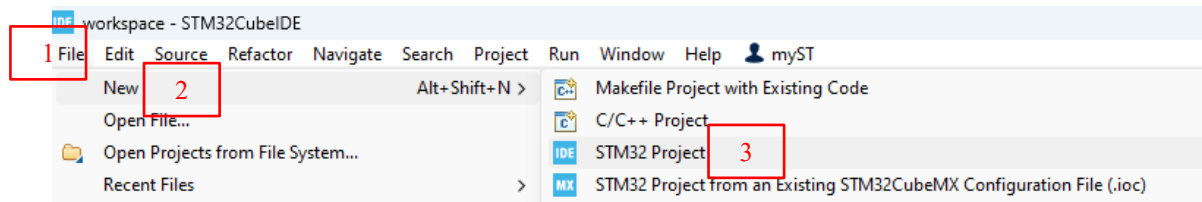
<https://community.st.com/t5/stm32-mcus/how-to-clone-a-dual-core-project-in-stm32cubeide/ta-p/619747>

9 First Project - Create A New Project

These are the general steps for creating a new project. We would repeatedly create a new project with the following practices.

9.1 Create A New Project

Step 1. Create a new project. File | New | STM32 Project



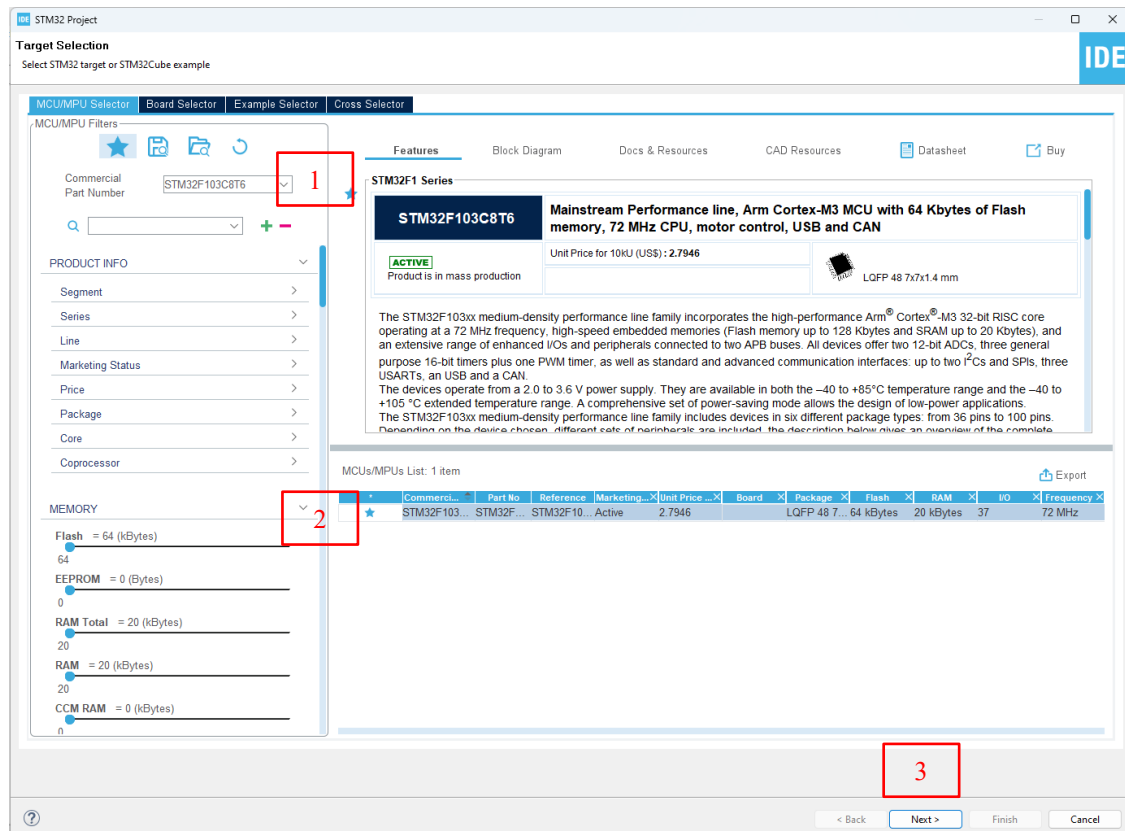
Step 2. Target Selection

[1] Type in Commercial Part number – STM32F103C8T6

[2] Select the part as listed. Click the star ☆. Save it as favorite. The star turn solid blue ★. The IDE would remember the favorite choice. If the same MCU is chosen for the following practice, just click the favorite.

[3] Click 'Next'

Note: For first-timer, spend sometime on the information shared.

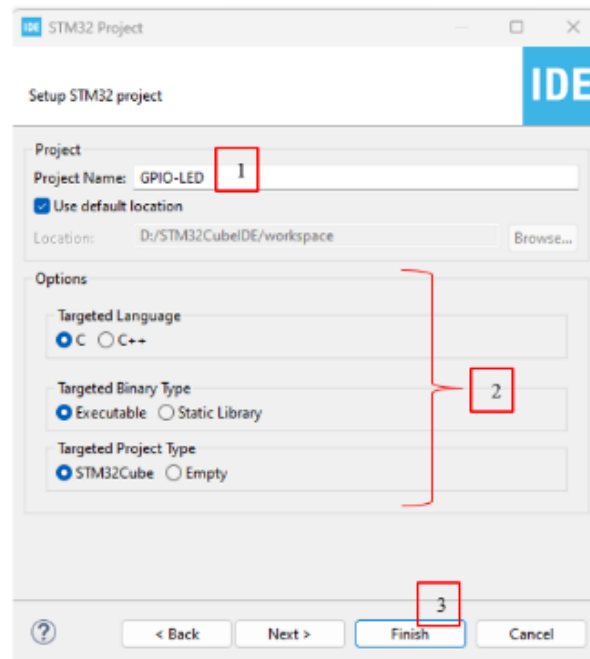


Step 3. Save to default location (workspace location)

[1] Type in Project Name – “GPIO-LED”

[2] Options. Do not change.

[3] Click Finish.




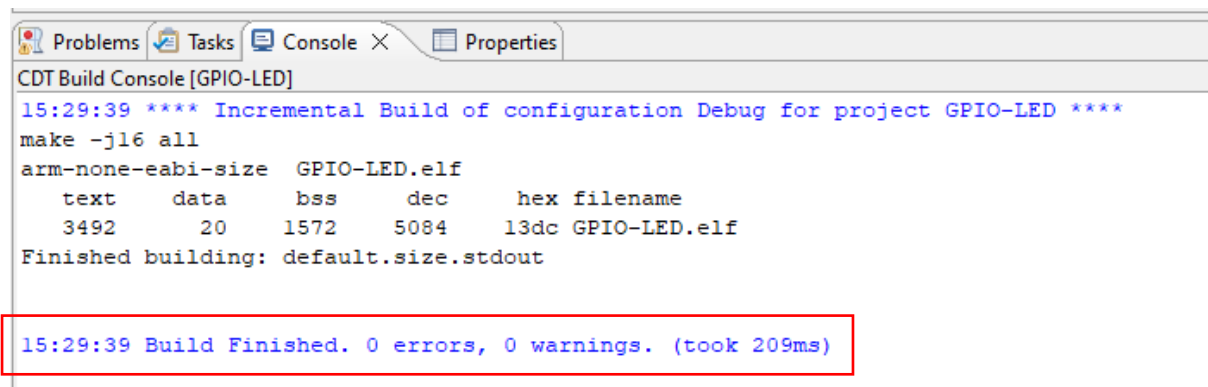
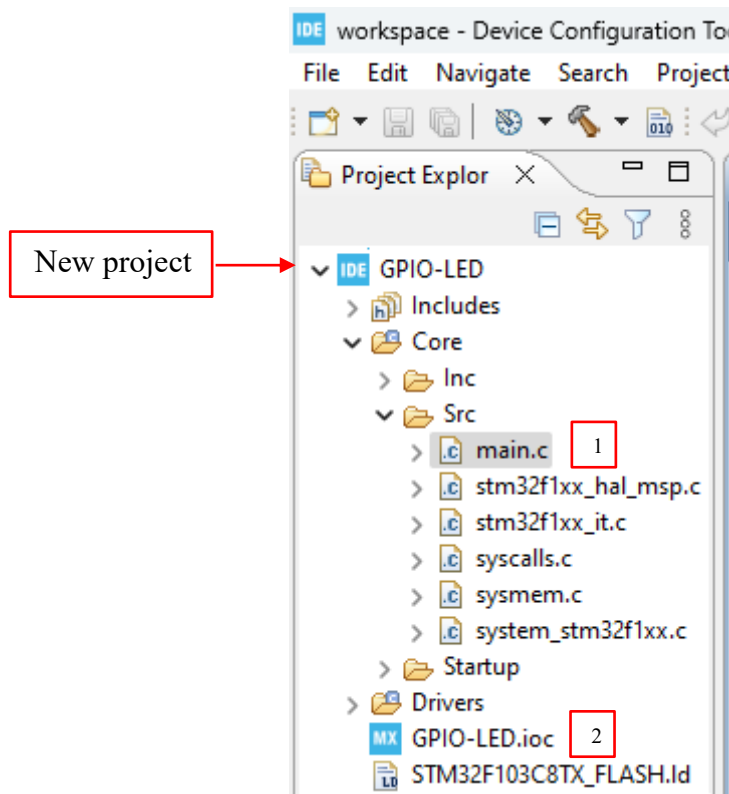
9.2 File Structure

The IDE would create a template for the selected MCU. Notice the listing in Project Explorer. A project is created.

1. A folder is created and saved in the default location (You specified it during installation, but you could still choose another location).
2. Open File Explorer and the save location. Notice the folder structure. No worry! We will explore and use the standard C and HAL libraries as we explore further.
3. At the moment, we are concerned about two files
 - [1] `./Core/Src/main.c`
 - [2] `./GPIO-LED.ioc`

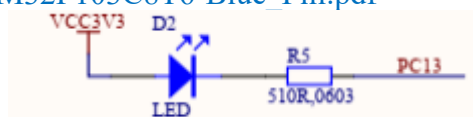
9.3 Build – main.c

1. Check the project and compiler
 - a. Do nothing on the new project.
 - b. Double click on `main.c`. It should pop up in the editor.
 - c. Build the code.
 - d. Click  or choose from the menu, Project | Build All
 - e. The compiled results should be printed on the console window.
`Example: Build Finished. 0 errors, 0 warnings. (took 209ms)`
 - f. The template is working. Ready for further work!

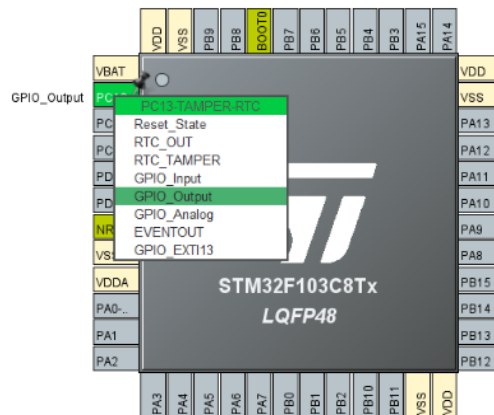


9.4 Pinout & Configuration

The development board has a built-in LED, at PC13, active LOW.
See: [original-schematic-STM32F103C8T6-Blue_Pill.pdf](#)



1. At the editor or Project Explorer, click Pinout & Configuration -- [GPIO-LED.ioc](#)
2. At the figure, click PC13 and set it as [GPIO_Ouput](#)




3. GPIO Mode and Configuration

[1] Click and select GPIO pin to configure

[2] GPIO output level – select **LOW**. (We expect LED is lit ON.)

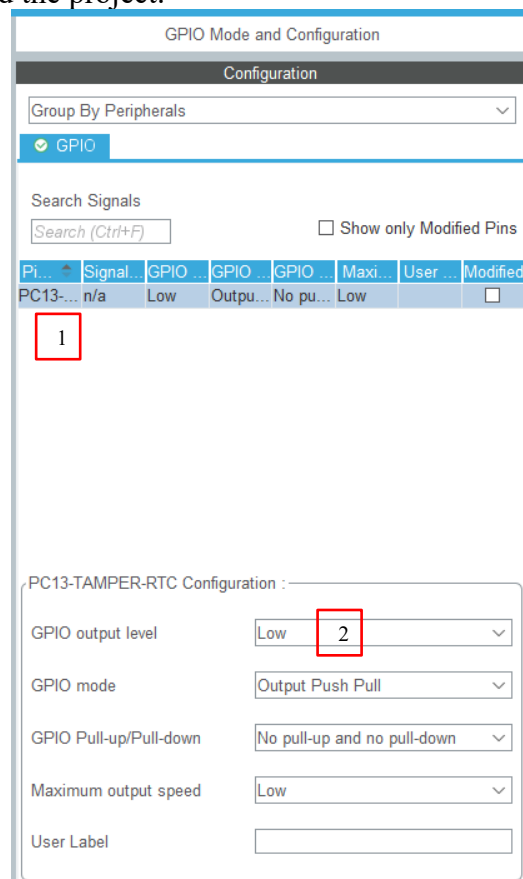
Keep the rest unchanged.

4. Code Generation


a. Click the icon  or from the menu, Project | Generate Code

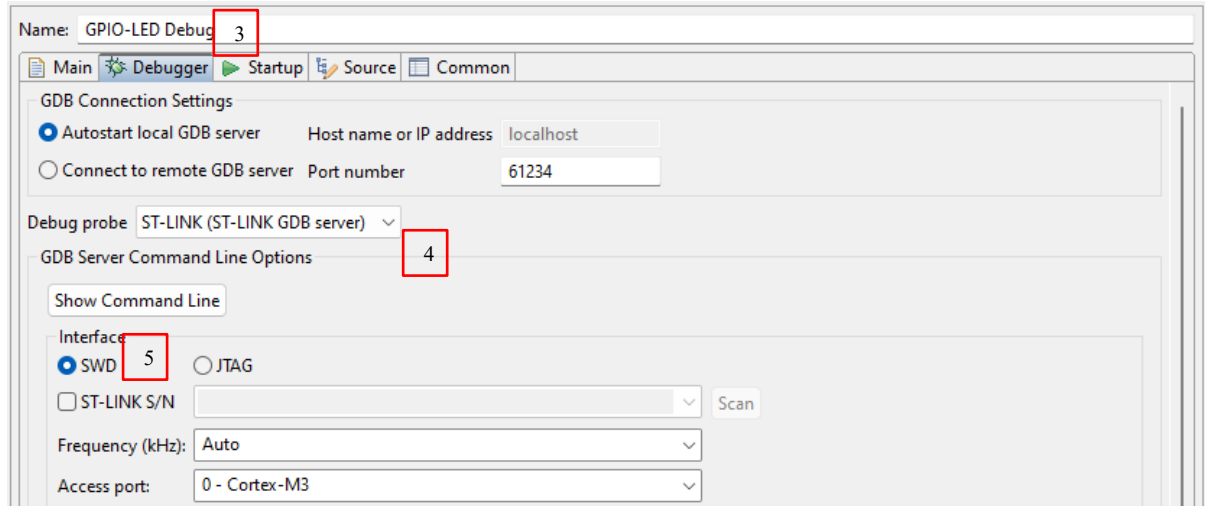
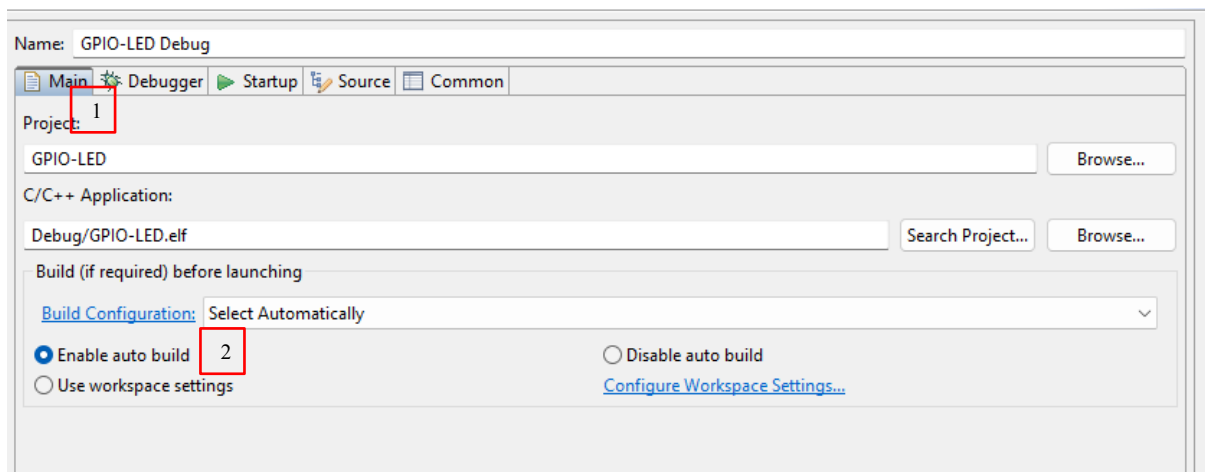
b. The IDE would convert the selected configuration into code (updated in main.c)

5. Click main.c, build the project.

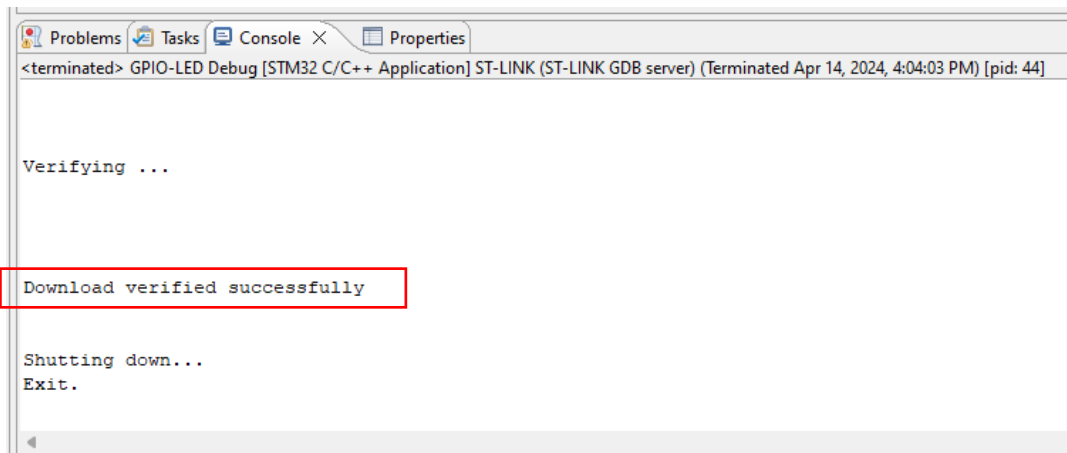


9.5 Run – Upload To MCU

1. Click the Run icon,  or click Run | Run Configurations... for the first time.
2. Run Configuration (only one time configuration)
 - [1] Main Tab
 - [2] Select [Enable auto build](#). This option would build the project before running the project.
 - [3] Debugger Tab
 - [4] At the dropdown menu, select [ST-Link](#)
 - [5] Select the interface as [SWD](#). (Namely Serial Wire Debug)



3. Make sure ST-Link V2 is plugged in and connected to the MCU.
4. Click [Run](#). Download the project to the MCU.
[Download verified successfully.](#)
 The build-in LED (PC13) is lit ON. (GPIO output level = LOW)



9.6 Re-configure Pinout

1. Repeat Pinout & Configuration, but this time, change the GPIO output level to **HIGH**
2. Compile and run.
3. The built-in LED (PC13) should be lit OFF. (GPIO output level = HIGH)

10 Practices

To create a new project, refer to First Project - Create A New Project.

List of practices:

1. GPIO-LED
2. GPIO-LED Blink

10.1 GPIO- LED

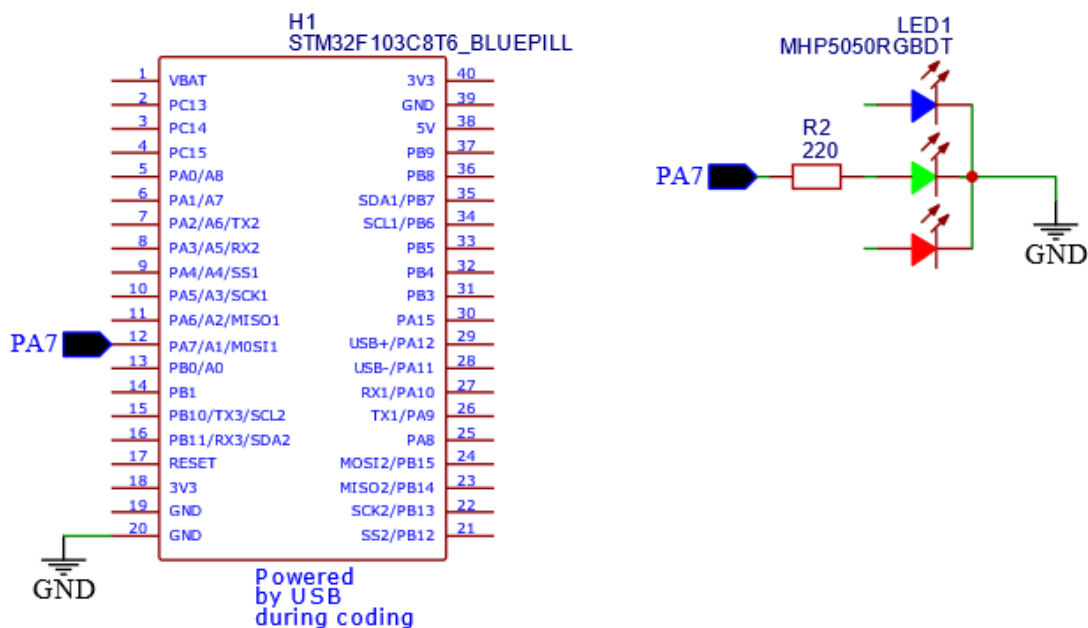
Requirement:

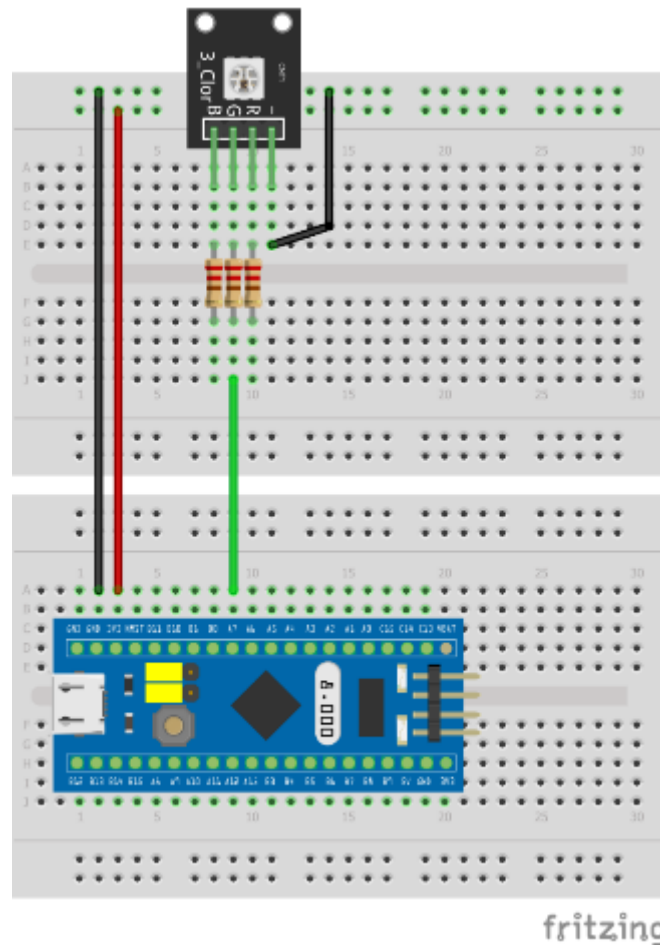
1. LED_GREEN blinks on specified time intervals.

10.1.1 Diagram

Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull , No pull-up and no pull-down	LED_GREEN

- Pay attention of the direction of current flow. This determine GPIO output level, either active LOW or active HIGH.





Current limiting
resistors
220 ~ 470 Ω

10.1.2 Practices

1. Explore GPIO – try different configurations.
 - a. The initial GPIO is configured as [GPIO_Output, **Low**, **Output Push Pull**, No pull-up and no pull-down]. [Build and run](#)
 - b. Update GPIO to [GPIO_Output, **High**, **Output Push Pull**, No pull-up and no pull-down]. [Build and run](#)
 - c. Update GPIO to [GPIO_Output, **Low**, **Output Open Drain**, No pull-up and no pull-down]. [Build and run](#).
 - d. Update GPIO to [GPIO_Output, **High**, **Output Open Drain**, No pull-up and no pull-down]. [Build and run](#).
2. Exploring main.c (update GPIO as [GPIO_Output, **Low**, **Output Push Pull**, No pull-up and no pull-down])
 - a. Scroll by mouse or click outline in Outline View
 - b. For the first time, observe the template (main.c)
 - i. header
 - ii. Functions declaration in C
 - iii. Comments – you could try to comment / uncomment. Shortcut: CTRL+/
iv. `int main(void)`
 1. Initialization – default and user
 2. `while` (1)
 - c. `void MX_GPIO_Init(void)` - Generated template by STM32CubeIDE
 1. struct and HAL

3. Exploring STM32CubeIDE. A quick search of the function used.
 - a. Within the same file. Practice: At main.c, hover the mouse cursor at `HAL_Init()`, right-click the mouse, and a window will pop up. Click [Open Declaration \(F3\)](#).
 - b. Source files and header files of STM32 HAL. Practice: At main.c, hover the mouse cursor at `HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, GPIO_PIN_SET);`, right-click the mouse; a window will pop up. Click [Open Declaration \(F3\)](#).
 - c. Spend some moments glaring at the source files. Observe HAL and its declaration.
4. Exploring STM32CubeIDE. Add user codes (Important! If you want to update configuration repetitively.)

```
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
LED_GREEN_Pin, GPIO_PIN_SET);
HAL_Delay(1000);
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
LED_GREEN_Pin, GPIO_PIN_RESET);
HAL_Delay(1000);
```

Identify the coding area as commented in the template.

Example:

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

- a. Add code within comment / before or after comment respectively in two attempts.
- b. Do update Pinout & Configuration and observe the difference.

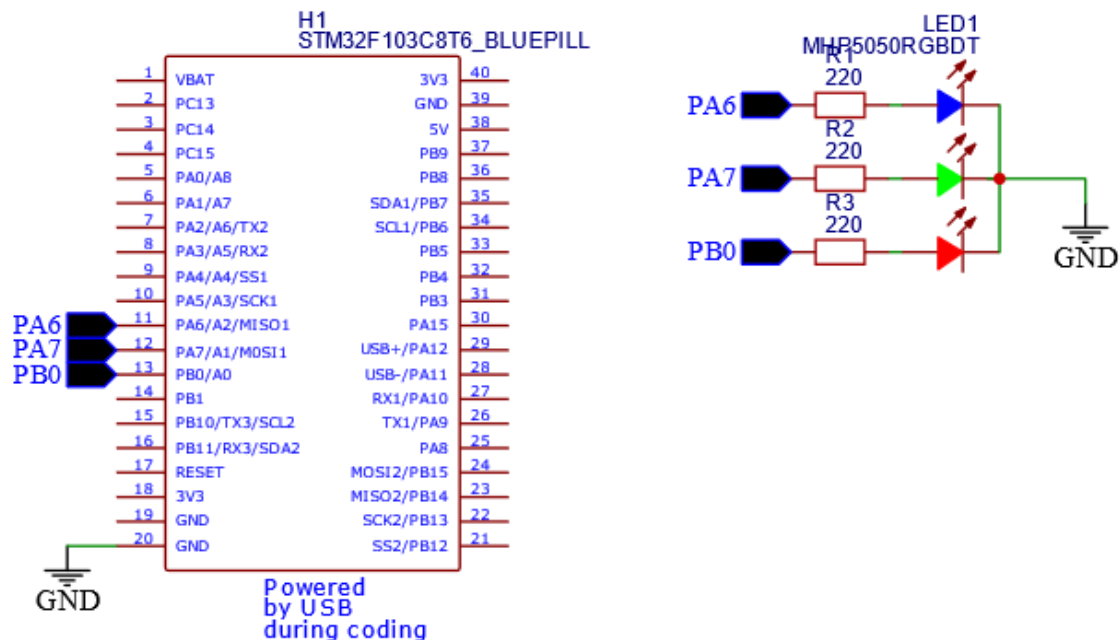
10.2 GPIO- LEDs Blink

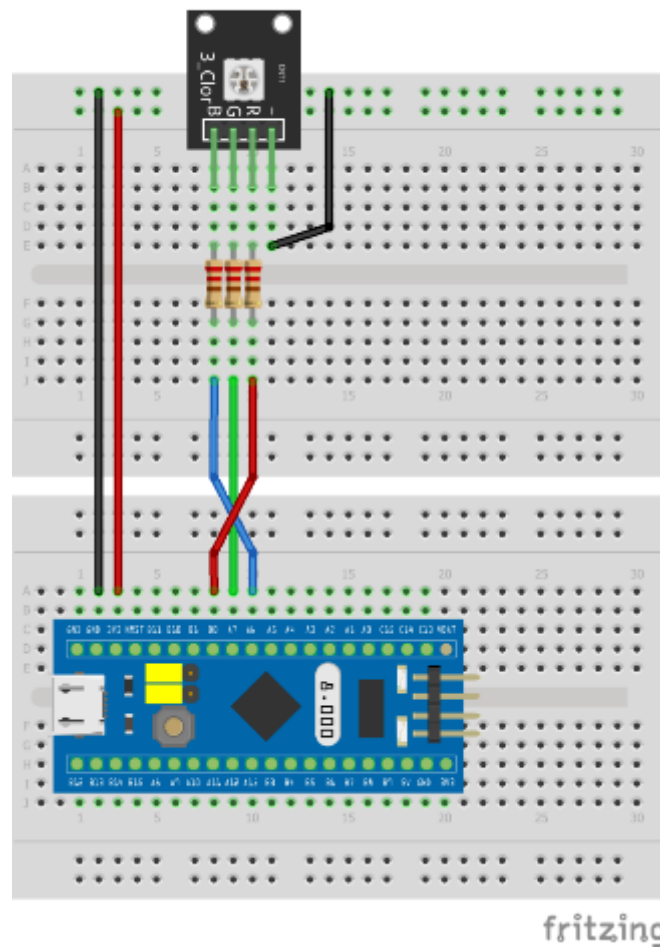
Requirement:

1. LED_BLUE, LED_GREEN, and LED_RED would be lit ON/OFF according to a pre-defined sequence.

10.2.1 Diagram

Pin	Mode	User Label
PA6	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_BLUE
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED





Current limiting
resistors
220 ~ 470 Ω

10.2.2 Practices

Let's say we want to mix the color repetitively using these sequences to update RGB led.

	State0	State1	State2	State3	State4	State5
LED_BLUE	0	0	1	1	1	1
LED_GREEN	0	1	1	1	0	0
LED_RED	1	1	1	1	0	1

- 1 – mean light ON
- 0 – mean light OFF

For each state count, it would take 1 second (Anyway, you could change it).

1. Update main.c, add in between the comment (after **while** (1))


```

/* USER CODE BEGIN 2 */
int state = 0;
/* USER CODE END 2 */
/* USER CODE BEGIN WHILE */
while (1)
{
    // red state
    if(state <=3 || state == 5){
        HAL_GPIO_WritePin(LED_RED_GPIO_Port,
        LED_RED_Pin,GPIO_PIN_SET);
    }else{
        HAL_GPIO_WritePin(LED_RED_GPIO_Port,
        LED_RED_Pin,GPIO_PIN_RESET);
    }
    // green state
    if(state >=1 && state <= 3){
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
        LED_GREEN_Pin,GPIO_PIN_SET);

        }else{
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
        LED_GREEN_Pin,GPIO_PIN_RESET);
        }

    // blue state
    if(state >=2&& state <= 5){
        HAL_GPIO_WritePin(LED_BLUE_GPIO_Port,
        LED_BLUE_Pin,GPIO_PIN_SET);
    }else{
        HAL_GPIO_WritePin(LED_BLUE_GPIO_Port,
        LED_BLUE_Pin,GPIO_PIN_RESET);
    }

    HAL_Delay(1000);

    state++;
    if(state>5){
        state = 0;
    }
}

```

2. Change `HAL_Delay(1000);` to shorter time. Say: 0.1 sec
3. Design your own sequence. Then update the code.

10.3 GPIO- LED-Buttons

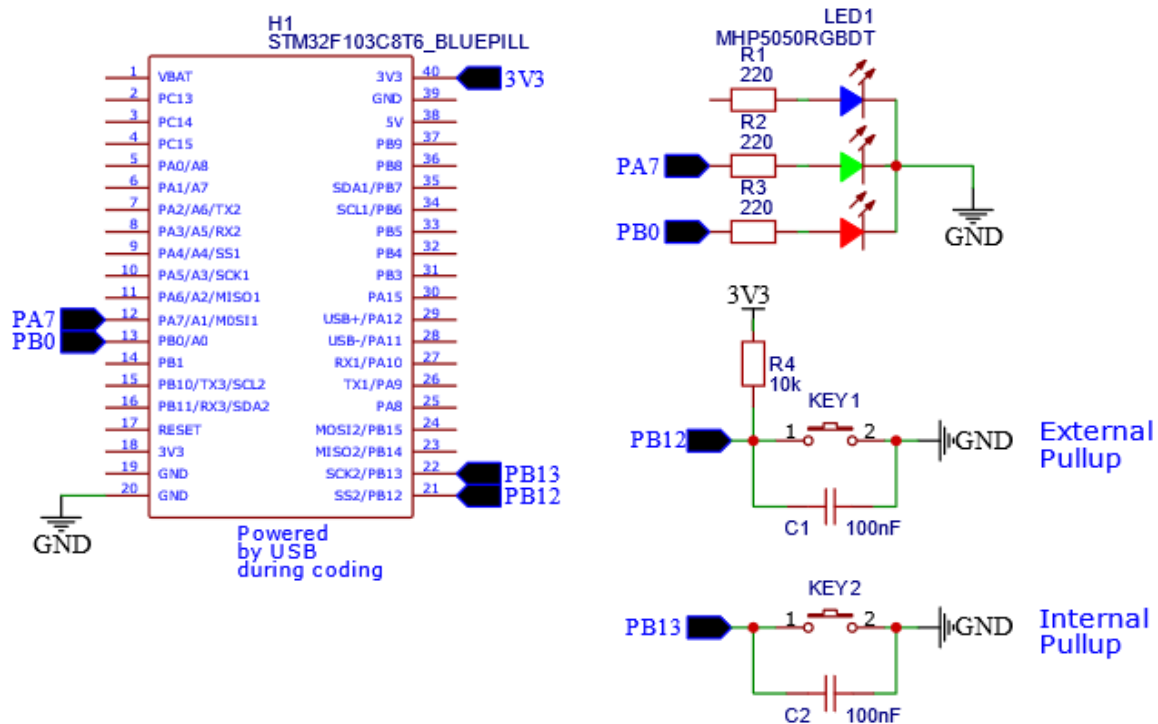
Requirement

1. We shall explore GPIO input and mechanical button bouncing issues.

10.3.1 Diagram

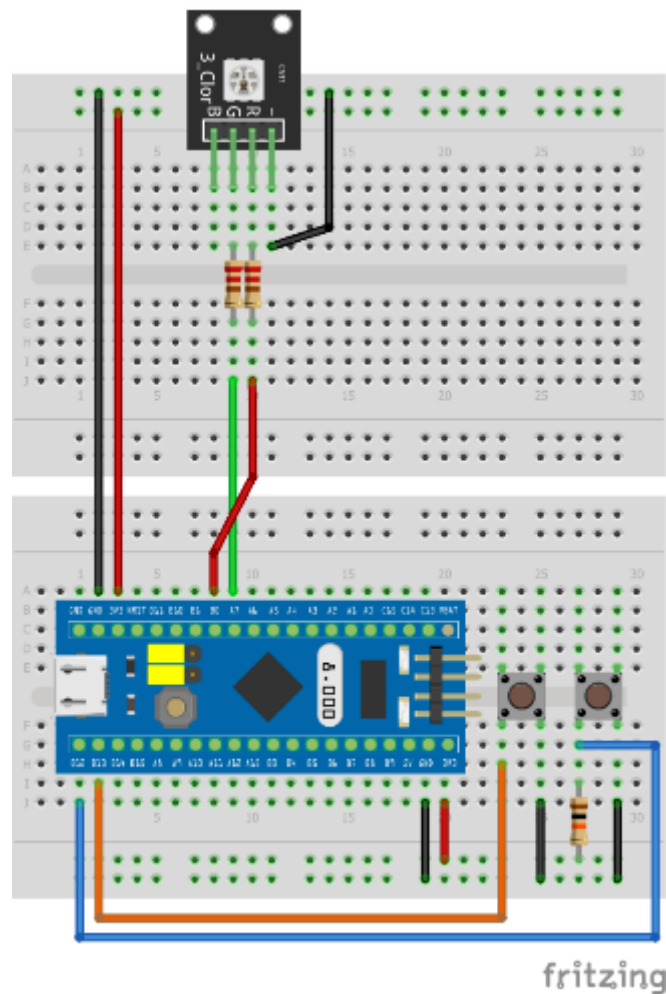
Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PB12	GPIO Input, No pull-up and no pull-down	KEY1
PB13	GPIO Input, Pull-up	KEY2

- KEY1 and KEY2 are mechanical momentary type.



*Adding capacitor 100nF to some extent could resolve the bouncing issue.

** Ignore the capacitor connection if not available.



10.3.2 Practices

1. KEY1 – not self holding.

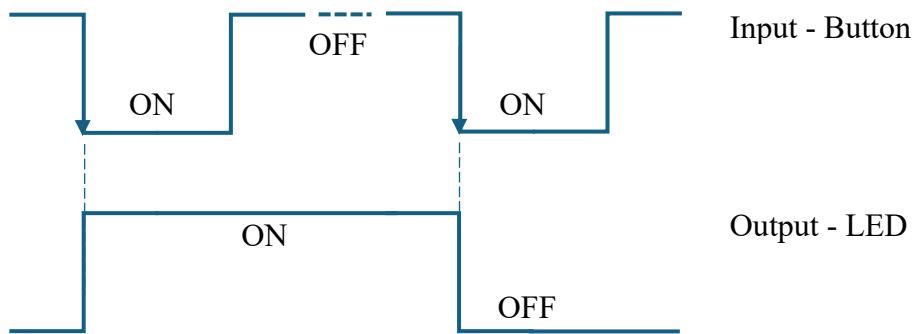
a. Add in these codes (after **while** (1)).

```
if(HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin)==GPIO_PIN_RESET ){
    HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
        LED_GREEN_Pin, GPIO_PIN_SET);
}else{
    HAL_GPIO_WritePin(LED_GREEN_GPIO_Port,
        LED_GREEN_Pin, GPIO_PIN_RESET);
}
```

b. Build and run

c. Press KEY1. Observe **GREEN** led.

2. KEY2 – self holding. We expect the button to operate this way, i.e., toggle the LED alternatively.



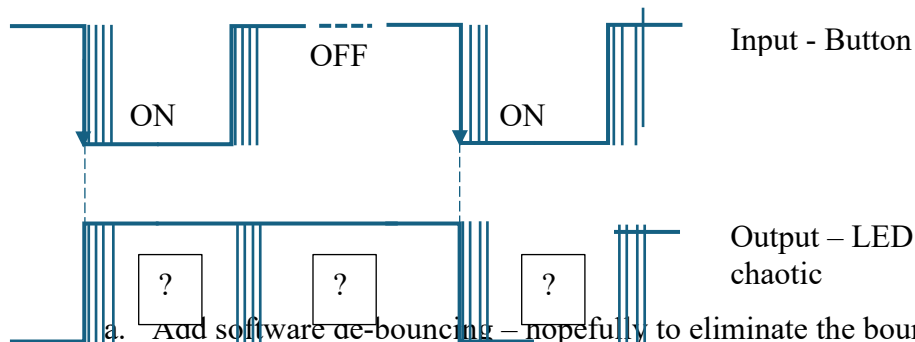
a. Add these codes within the while loop.

```
if(HAL_GPIO_ReadPin(KEY2_GPIO_Port, KEY2_Pin)==GPIO_PIN_RESET ){

    if(HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin)==GPIO_PIN_RESET ){
        HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
    }
}
```

c. Press KEY2. Observe **RED** led.

3. A momentary mechanical pushbutton displays a phenomenon called a bouncing effect. Practically, the button would exhibit a short moment of state changes as its switch contact displays the spring effect. This uncertainty could disturb the expected outcome.



a. Add software de-bouncing – hopefully to eliminate the bouncing effect.

```
if(HAL_GPIO_ReadPin(KEY2_GPIO_Port, KEY2_Pin)==GPIO_PIN_RESET ){
    HAL_Delay(10);
    if(HAL_GPIO_ReadPin(KEY2_GPIO_Port, KEY2_Pin)==GPIO_PIN_RESET ){
        HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
        while(HAL_GPIO_ReadPin(KEY2_GPIO_Port,
                                KEY2_Pin)==GPIO_PIN_RESET){};
    }
}
```

b. Build and run

c. Press KEY2. Observe **RED** led.

Think about this:

1. Bouncing effects are found during:
 - a. High \rightarrow Low
 - b. Low \rightarrow High
2. Is software de-bouncing the ultimate solution?

10.4 GPIO- EXTI

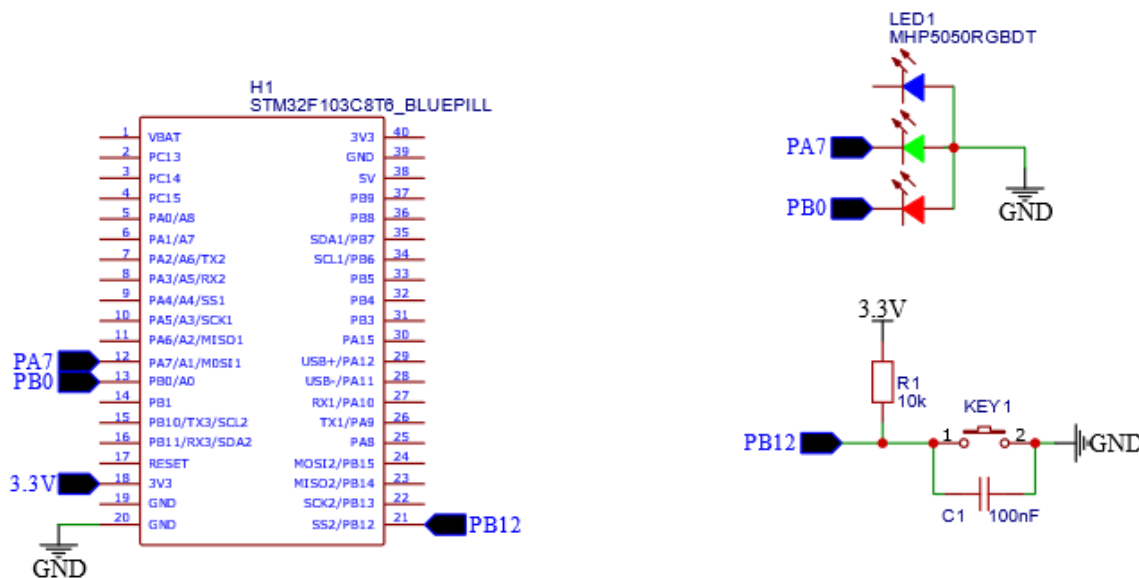
Requirement:

1. LED_RED is blinking at intervals of 2 seconds repetitively.
2. Pressing KEY1 would toggle LED_GREEN at any time.

10.4.1 Diagram

Pin	Mode	User Label
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PB12	GPIO_Input, No pull-up, and no pull-down	KEY1

- KEY1 is a mechanical momentary type.



10.4.2 Practices

1. KEY1 – as GPIO_Input – code within main.c - **while** (1).
 - a. Add in these codes.


```

HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, GPIO_PIN_SET);
HAL_Delay(1000);
HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, GPIO_PIN_RESET);
HAL_Delay(1000);

// without interrupt
if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET) {
    HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
    while (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET);
}

```

- b. Build and run.
 - c. Press KEY1. Observe RGB LED.
2. KEY1 – as GPIO interrupt
 - a. Re-configure PB12 as [GPIO_EXTI12, No pull-up and no pull-down]. Name as KEY1. Generate the codes.
 - b. In main.c – remove `// without interrupt` and if-statement
 - c. In stm32f1xx_it.c, scroll to the end and find:



```
void EXTI15_10_IRQHandler(void)
{
    /* USER CODE BEGIN EXTI15_10_IRQn 0 */
    // de-bouncing by delay
    HAL_Delay(10);
    if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET) {
        HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
    }

    /* USER CODE END EXTI15_10_IRQn 0 */
    HAL_GPIO_EXTI_IRQHandler(KEY1_Pin);
    /* USER CODE BEGIN EXTI15_10_IRQn 1 */

    /* USER CODE END EXTI15_10_IRQn 1 */
}
```

**Note: Adding `HAL_Delay(10)` is not a good practice whenever deploying EXTI*

- d. Build and run. (Well! It won't work as expected!)
- e. Open [Pinout & Configuration](#) – update [Preemption Priority](#)
 - i. Time base: System tick timer > EXTI line[15:10] interrupts

Configuration		
NVIC		
Code generation		
NVIC interrupt table	Enabled	Preemption P
Non maskable interrupt	<input checked="" type="checkbox"/>	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0
Memory management fault	<input checked="" type="checkbox"/>	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0
Debug monitor	<input checked="" type="checkbox"/>	0
Pendable request for system service	<input checked="" type="checkbox"/>	0
Time base: System tick timer	<input checked="" type="checkbox"/>	14
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0
Flash global interrupt	<input type="checkbox"/>	0
RCC global interrupt	<input type="checkbox"/>	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	15

- f. Build and run. (Well! This time it should work as expected!)
- g. Press KEY1. Observe RGB LED.

Explanation:

1. The higher the priority, the lower the assigned number.
2. `HAL_Delay(10)` is dependant on `Systick()`.

10.5 UART in Polling Mode

UART STM32

- Simple UART communication in polling mode
- UART with Interrupt
- UART with DMA

This example features UART in polling mode.

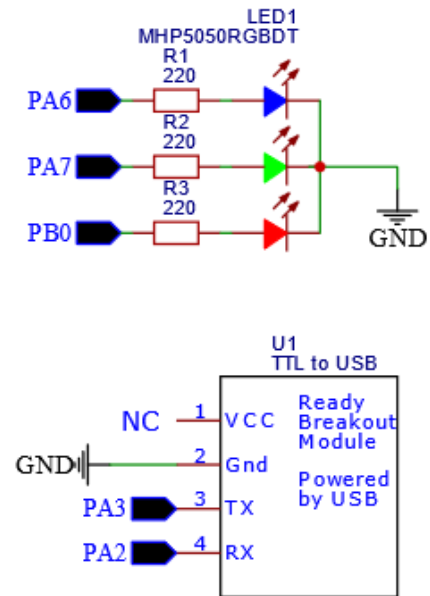
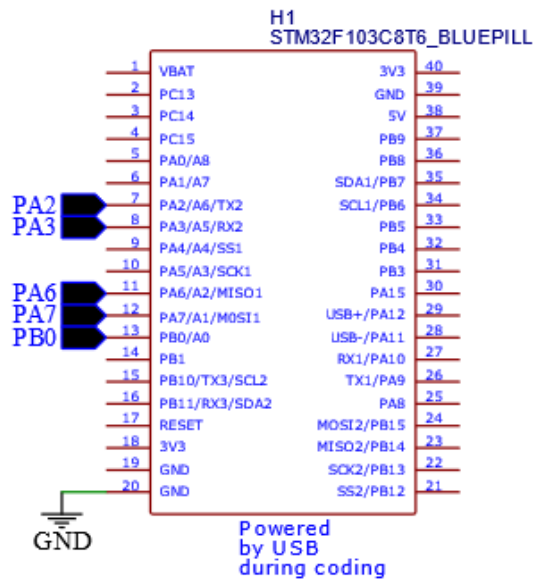
Requirement:

1. Each LED is lit up on the text command via the serial terminal app.
Press enter to send. Observe the change.

Command	Color	ON/OFF
R0	LED_RED	OFF
R1	LED_RED	ON
G0	LED_GREEN	OFF
G1	LED_GREEN	ON
B0	LED_BLUE	OFF
B1	LED_BLUE	ON

10.5.1 Diagram

Pin	Mode	User Label
PA6	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_BLUE
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PA2	USART2_TX GPIO settings completed by Pin & Configuration	
PA3	USART2_RX GPIO settings completed by Pin & Configuration	



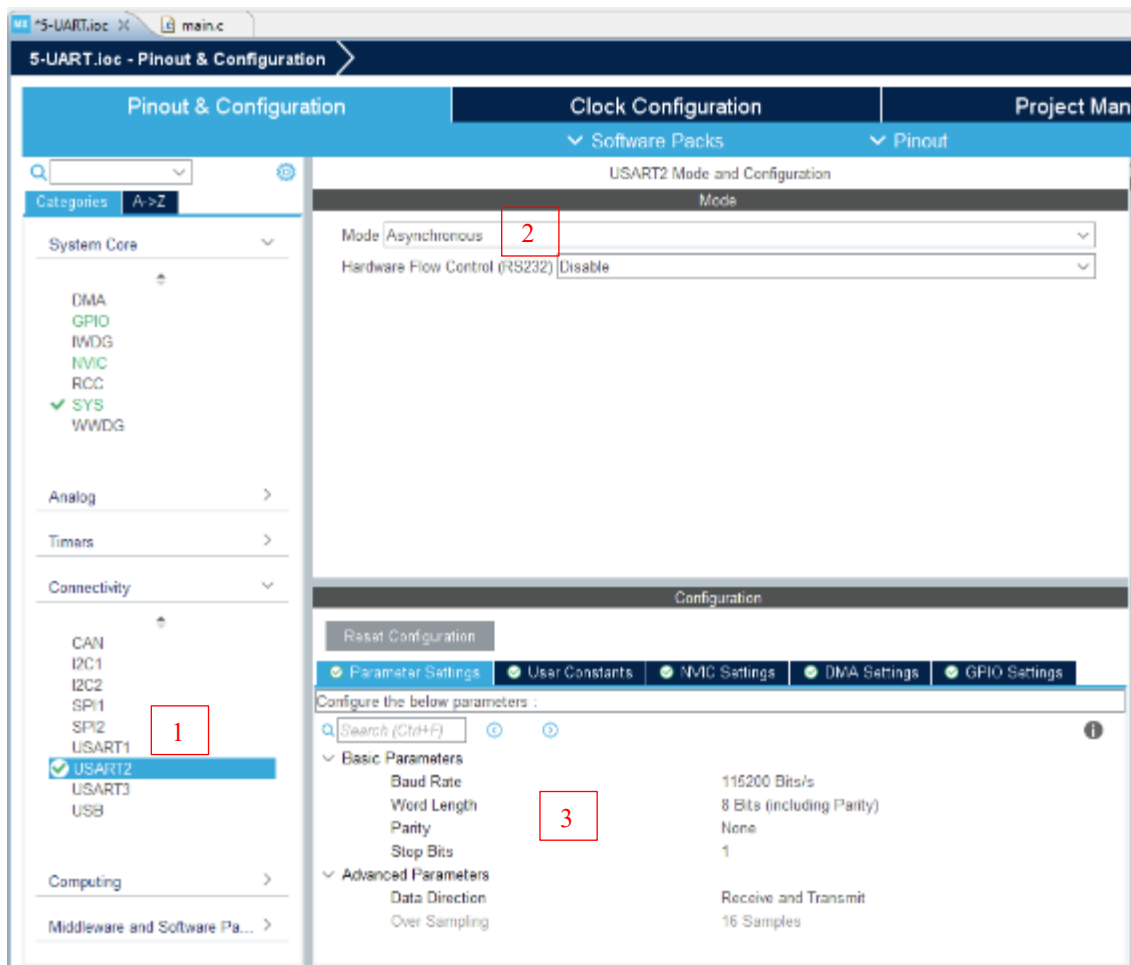
Note:

The connection of TX-RX between an MCU and a UART-to-USB converter must be criss-crossed.

USART2 – Configuration (refer to the figure below.)

- 1 Under Connectivity, select USART2
- 2 Mode: Asynchronous
- 3 Basic parameters: 115200 Bits/s, 8N1

Note: The same configuration must be set up for the serial terminal app.




10.5.2 Practices

1. At main.c, observe `static void MX_USART2_UART_Init(void)`. Check if USART2 is configured as planned.
2. Update the code.
 - a. Add in these codes at main.c

```

uint8_t receiveData[2]; // buffer

while (1) 
{
    HAL_UART_Receive(&huart2, receiveData, 2, HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart2, receiveData, 2, 100);
    GPIO_PinState state = GPIO_PIN_SET;

    if (receiveData[1]=='0'){
        state = GPIO_PIN_RESET;
    }else if (receiveData[1]=='1') {
        state = GPIO_PIN_SET;
    }

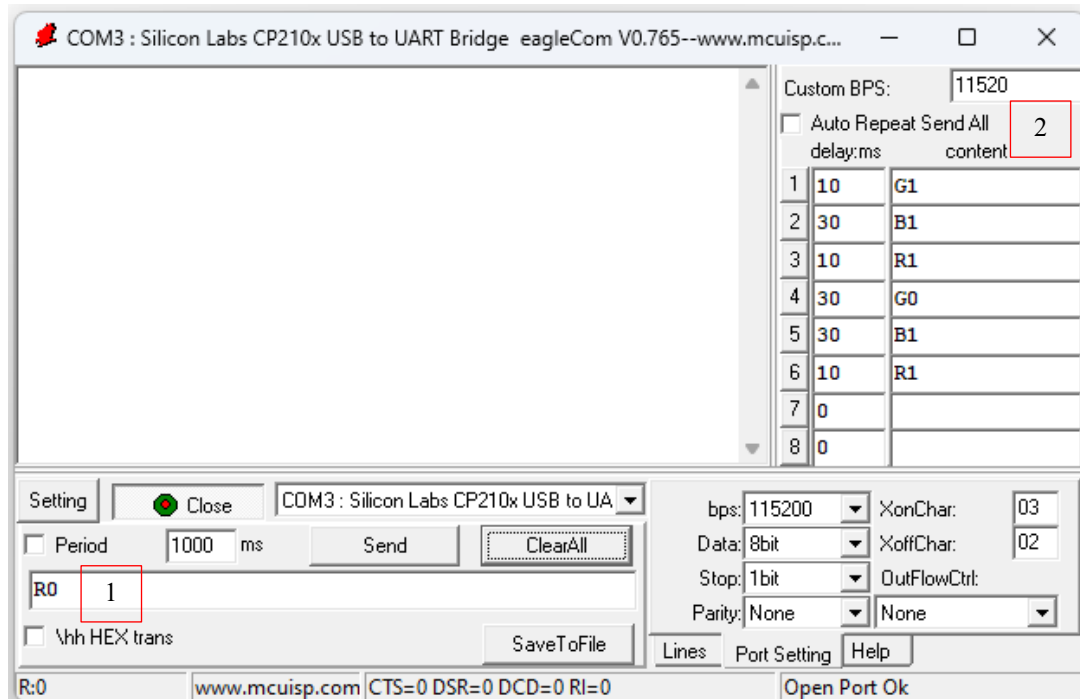
    if (receiveData[0]=='R'){
        HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, state);
    }else if (receiveData[0]=='G') {
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, state);
    }else if (receiveData[0]=='B') {
        HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin, state);
    }
}

```

- b. Build and run
- c. Open serial terminal app. Configure port setting (115200, 8N1). Click Open to establish serial communication with the MCU. At serial terminal app. This example deploys – eagleCom downloaded from www.mcuisp.com
- d. At serial terminal app. Refer to the figure below and enter the text command:

At 1, try to enter these commands. The commands are listed in **UART in Polling Mode**.

Refer to the figure below, enter text command: At 2, try to enter a series of commands and time intervals. Click “Auto Repeat Send All”. Observe the change.



3. What's wrong?

a. Received buffer

`HAL_UART_Receive(&huart2, receiveData, 2, HAL_MAX_DELAY);`
 configured to read 2 bytes of data

- At transmit side, enter more than two characters. Ex: R00
- The 'expected' sequence is not in a row! Only 2 characters is read, and the rest is left to the following read.

b. Missing UART-Rx

- Notice that in main.c, Uart receive is coded within **while** (1) loop. What if there are codes that take longer processing time? Missing data?
- Try add in `HAL_Delay(2000);` within **while** (1) loop. Build and run. Examine the LEDs.

10.6 UART With Interrupt

This is another mode of UART.
UART with Interrupt.

Requirement:

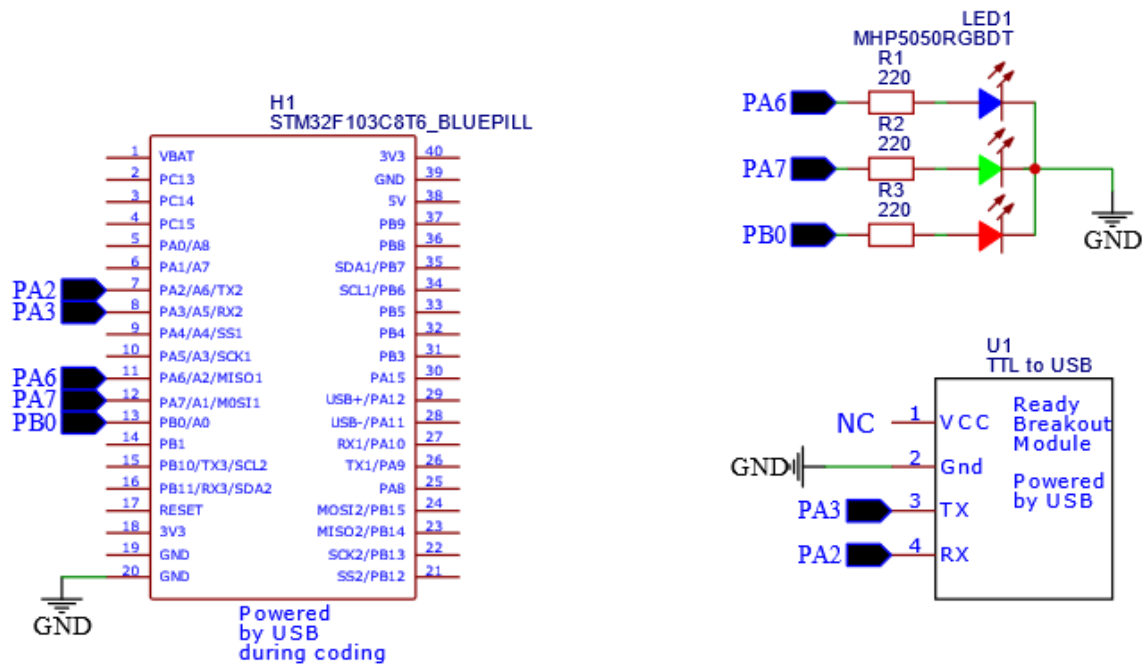
1. Each LED is lit up on text command via serial terminal app in a callback function.
Press enter to send. Observe the change.

Command	Colour	ON/OFF
R0	LED_RED	OFF
R1	LED_RED	ON
G0	LED_GREEN	OFF
G1	LED_GREEN	ON
B0	LED_BLUE	OFF
B1	LED_BLUE	ON

10.6.1 Diagram

Hardware wiring and Pin & Configuration are the same as before; add in the configuration of NVIC Settings.

Pin	Mode	User Label
PA6	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_BLUE
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PA2	USART2_TX GPIO settings completed by Pin & Configuration	
PA3	USART2_RX GPIO settings completed by Pin & Configuration	



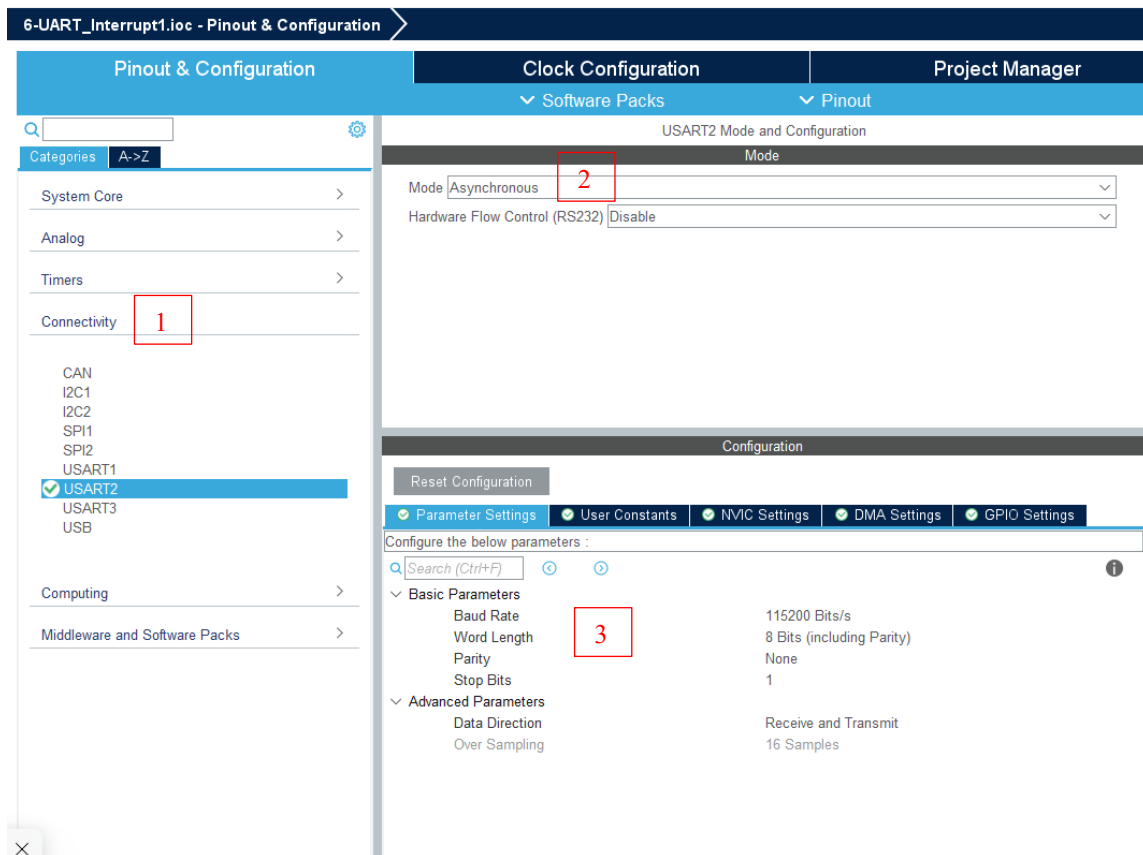
Note:

The connection of TX-RX between an MCU and a UART-to-USB converter must be criss-crossed.

USART2 – Configuration (refer to the figure below.) Under Parameter Settings

- 1 Under Connectivity, select USART2.
- 2 Mode: Asynchronous.
- 3 Basic parameters: 115200 Bits/s, 8N1.

Note: The same configuration must be set up for the serial terminal app.



USART2 – Configuration (refer to the figure below.) Under NVIC Settings

- 4 Select tab | NVIC Settings
- 5 Click Enabled – USART2 global interrupt

6-UART_Interrupt1.ioc - Pinout & Configuration

Pinout & Configuration

Categories A-Z

System Core

- DMA
- GPIO
- IWDG
- NVIC
- RCC
- ☒ SYS
- WWDG

Analog

Timers

Connectivity

- CAN
- I2C1
- I2C2
- SPI1
- SPI2
- USART1
- ☒ USART2
- USART3
- USB

Computing

Middleware and Software Packs

Clock Configuration

Pinout

USART2 Mode and Configuration

Mode

- Mode Asynchronous
- Hardware Flow Control (RS232) Disable

Configuration

Reset Configuration

NVIC Settings
DMA Settings
GPIO Settings

Parameter Settings
User Constants

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
USART2 global interrupt	<input checked="" type="checkbox"/>	0	0

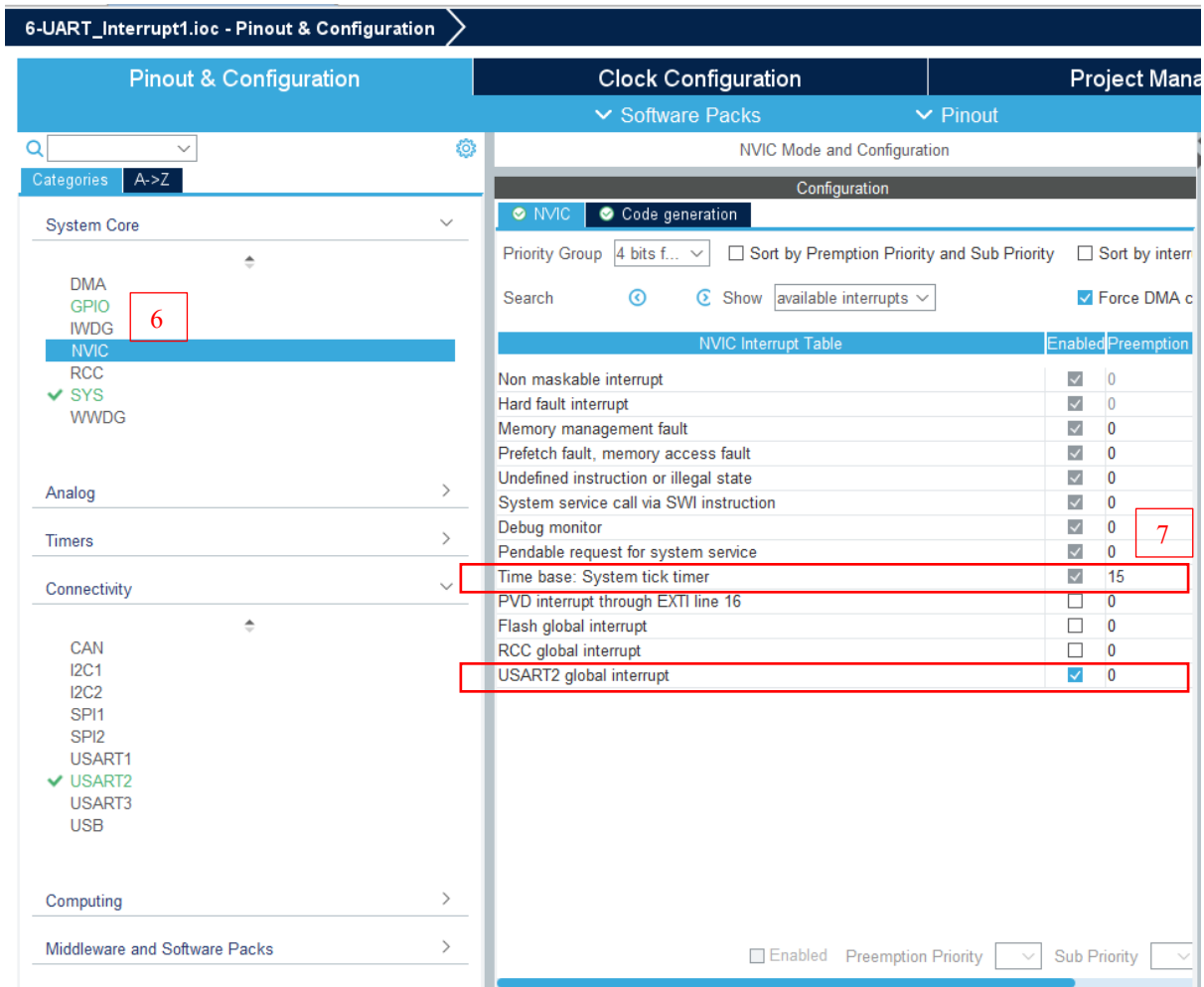
NVIC – Preemption Priority

6

Under NVIC, select USART2

7

Update preemption priority as shown. *[note: Lower number has higher priority.]*



10.6.2 Practices

1. Check the self-generated, **void USART2_IRQHandler(void)** at source file - stm32f1xx_it.c. Do nothing.
2. At main.c, observe **static void MX_USART2_UART_Init(void)**. Check if USART2 is configured as planned. Do nothing.
3. At main.c, update the code.
 - a. Add in a callback function at main.c
 - b. Build and run.
 - c. Open serial terminal app. Configure port setting (115200, 8N1). Click Open to establish serial communication with the MCU. At serial terminal app. This example deploys – eagleCom downloaded from www.mcuisp.com
 - d. Examine the LEDs. See the command text in the beginning of **UART With Interrupt**.

Refer to 10.5 UART in Polling Mode.

Compared to the previous example on UART-Polling Mode, communication commands are removed in **while** (1)


```

/* Private user code -----
-----*/
/* USER CODE BEGIN 0 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    HAL_UART_Transmit_IT(&huart2, receiveData, 2);
    GPIO_PinState state = GPIO_PIN_SET;

    if (receiveData[1]=='0'){
        state = GPIO_PIN_RESET;
    }else if (receiveData[1]=='1') {
        state = GPIO_PIN_SET;
    }

    if (receiveData[0]=='R'){
        HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, state);

    }else if (receiveData[0]=='G') {
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, state);
    }else if (receiveData[0]=='B') {
        HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin, state);
    }
    HAL_UART_Receive_IT(&huart2, receiveData, 2);
}
/* 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();

    /* 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 */

    HAL_UART_Receive_IT(&huart2, receiveData, 2); 
    /* USER CODE END 2 */

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

        /* USER CODE END WHILE */ 

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

```

4. At main.c, let's pretend to do some work in **while** (1)
 - a. Add a delay - `HAL_Delay(2000);`
 - b. Build and run.
 - c. Examine the LEDs.
 - d. Is the LEDS being delayed if `HAL_Delay(2000)` is prolonged? Try.

10.7 UART With DMA

This is another mode of UART.
UART2 is configured as an event.

Requirement:

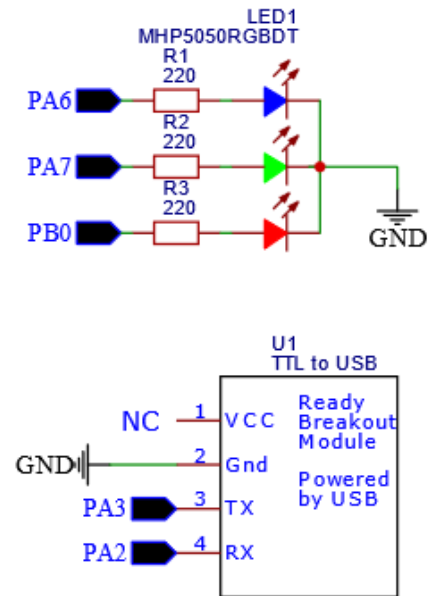
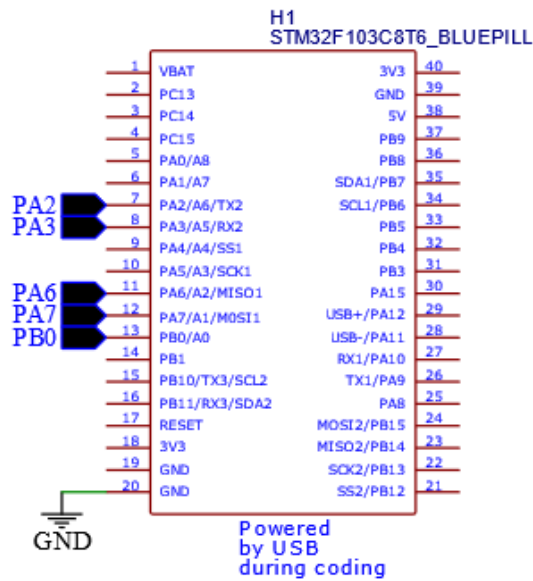
1. Each LED is lit up on text command via serial terminal app in a callback function.
Press enter to send. Observe the change.

Command	Colour	ON/OFF
R0	LED_RED	OFF
R1	LED_RED	ON
G0	LED_GREEN	OFF
G1	LED_GREEN	ON
B0	LED_BLUE	OFF
B1	LED_BLUE	ON

10.7.1 Diagram

Hardware wiring and Pin & Configuration are the same as before; add in the configuration of NVIC Settings.

Pin	Mode	User Label
PA6	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_BLUE
PA7	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_GREEN
PB0	GPIO_Output, Low, Output Push Pull, No pull-up and no pull-down	LED_RED
PA2	USART2_TX GPIO settings completed by Pin & Configuration	
PA3	USART2_RX GPIO settings completed by Pin & Configuration	



Note:

The connection of TX-RX between an MCU and a UART-to-USB converter must be criss-crossed.

Note:

Repeat the same configuration for UART With Interrupt; add in the selection DMA Settings.

Note:

Alternatively, you could clone a project.

NVIC – Preemption Priority

1

Under NVIC, select USART2

2

Update preemption priority as shown. [note: Lower number has higher priority.]

7-USART_Interrupt_DMA.ioc - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project M

Software Packs | Pinout

NVIC Mode and Configuration

Configuration

✓ NVIC | ✓ Code generation

Priority Group: 4 bits for pre-... | ☐ Sort by Preemption Priority and Sub Priority | ☐ Sort by interrupts names

Search: | Show: available interrupts | ☒ Force DMA channels Interrupts

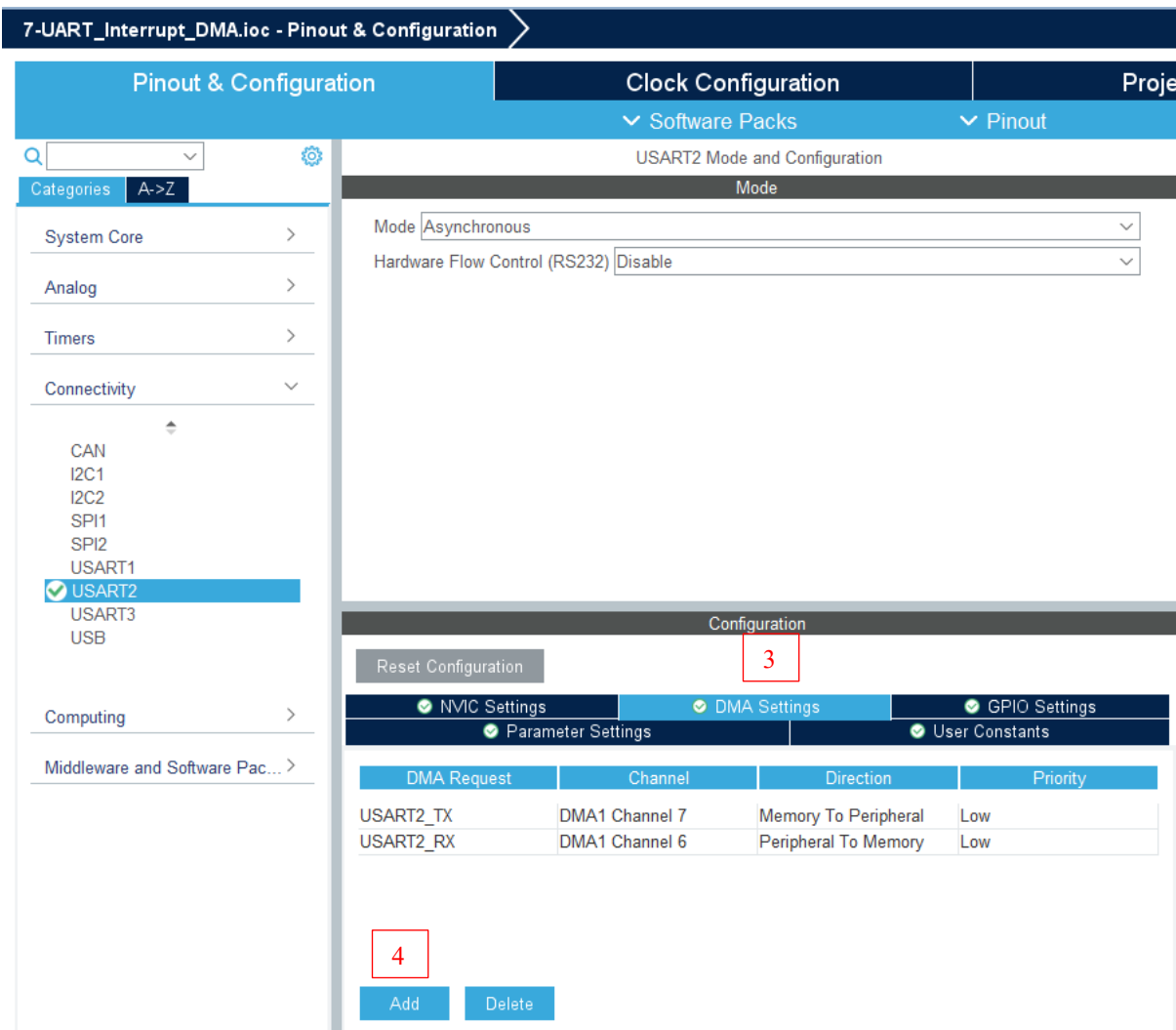
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	15	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
DMA1 channel6 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel7 global interrupt	<input checked="" type="checkbox"/>	0	0
USART2 global interrupt	<input checked="" type="checkbox"/>	0	0

☐ Enabled | Preemption Priority: | Sub Priority:

USART2 – DMA Settings

- 3 Under USART2, select DAM Settings
- 4 Add TX and RX. Leave the setting alone.

Note: We configure a channel between UART2 and DMA. The CPU would not be called for each interrupt.



10.7.2 Practices

1. Check the self-generated, **void USART2_IRQHandler(void)** at source file - stm32f1xx_it.c. Do nothing.
2. At main.c, observe **static void MX_USART2_UART_Init(void)**. Check if USART2 is configured as planned. Do nothing.
3. At main.c, observe **static void MX_DMA_Init(void)**. Check if DMA is configured accordingly. Do nothing.
4. At main.c, update the code.
 - a. Add in a callback function at main.c
 - b. Build and run.
 - c. Open serial terminal app. Configure port setting (115200, 8N1). Click Open to establish serial communication with the MCU. At serial terminal app. This example deploys – eagleCom downloaded from www.mcuisp.com
 - d. Examine the LEDs.

```

/* Private user code -----
-----*/
/* USER CODE BEGIN 0 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
    HAL_UART_Transmit_DMA(&huart2, receiveData, 2);
    GPIO_PinState state = GPIO_PIN_SET;

    if (receiveData[1]=='0'){
        state = GPIO_PIN_RESET;
    }else if (receiveData[1]=='1') {
        state = GPIO_PIN_SET;
    }

    if (receiveData[0]=='R'){
        HAL_GPIO_WritePin(LED_RED_GPIO_Port, LED_RED_Pin, state);

    }else if (receiveData[0]=='G') {
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, state);
    }else if (receiveData[0]=='B') {
        HAL_GPIO_WritePin(LED_BLUE_GPIO_Port, LED_BLUE_Pin, state);
    }
    HAL_UART_Receive_DMA(&huart2, receiveData, 2);

}

void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t
Size){
    if(huart==&huart2){
        HAL_UART_Transmit_DMA(huart, receiveData, Size);

        HAL_UARTEx_ReceiveToIdle_DMA(&huart2, receiveData,
sizeof(receiveData));
        __HAL_DMA_DISABLE_IT(&hdma_usart2_rx, DMA_IT_HT);
    }
}

/* 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();

    /* 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_DMA_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */


    // HAL_UART_Receive_DMA(&huart2, receiveData, 2);
    HAL_UARTEx_ReceiveToIdle_DMA(&huart2, receiveData,
    sizeof(receiveData));
    __HAL_DMA_DISABLE_IT(&hdma_usart2_rx, DMA_IT_HT);
    /* USER CODE END 2 */

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

        /* USER CODE END WHILE */

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

```



5. At main.c, let's pretend to do some work in **while** (1)
 - e. Add a delay - `HAL_Delay(2000);`;
 - f. Build and run.
 - g. Examine the LEDs.
 - h. Is the LEDs being delayed if `HAL_Delay(2000)` is prolonged? Try.

6. Study the difference of UART in Polling Mode, UART With Interrupt, UART With DMA.

10.7.3 References on UART

https://wiki.st.com/stm32mcu/wiki/Getting_started_with_UART

11 STM32CubeIDE – Shortcut Keys

Coding becomes much efficiency if some frequent commands are repeated. A few of them are:

CTRL+SHIFT+L	List keyboard shortcuts List all defined keyboard shortcuts
CTRL+Left Mouse or F3	Open Declaration
CTRL+/ CTRL+SPACE	comment or uncomment // Code completion Code completion/parameter hints depending on context Parameter hints

Do search in STM32CubeIDE for a complete list of shortcut keys. Or simply google it.

12 Tools

A few useful tools are introduced.

1. Serial terminal

A number of free apps are available online. **CoolTerm** and **eaglecom** are introduced; however, you could learn and use any serial terminal app for the application in reading/writing serial data. Used together to read/write serial data via USB port at the computer. For example: UART-to-USB, RS485-to-USB, RS232-to-USB, etc.

Note: We will explore other serial terminal apps throughout the practices. Some may appear to be convenient and intuitive than others.

2. In System Programming (ISP), include In Circuit Programming and In Application Programming.

3. ST-Link Utility

STM32 ST-LINK Utility (STSW-LINK004) is a full-featured software interface for programming STM32 microcontrollers.

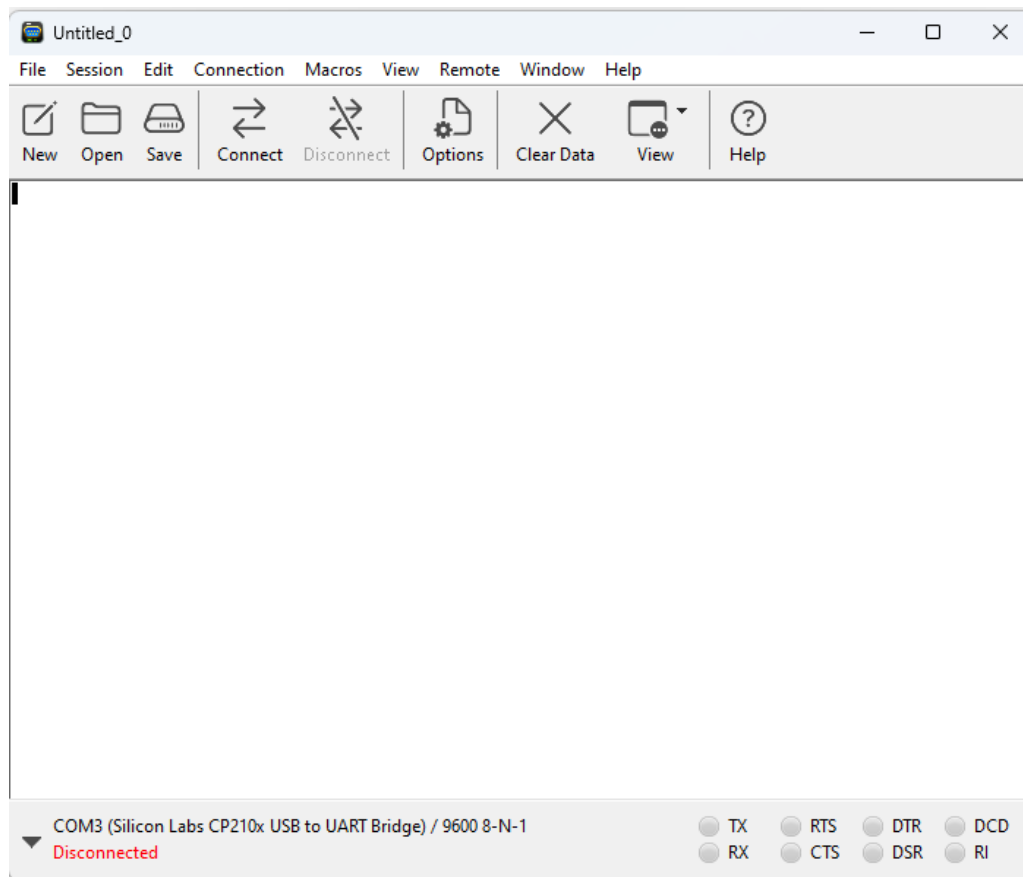
12.1 Serial App – CoolTerm

Serial terminal app. Free to download.

<https://coolterm.en.lo4d.com/windows>

Tutorial on CoolTerm

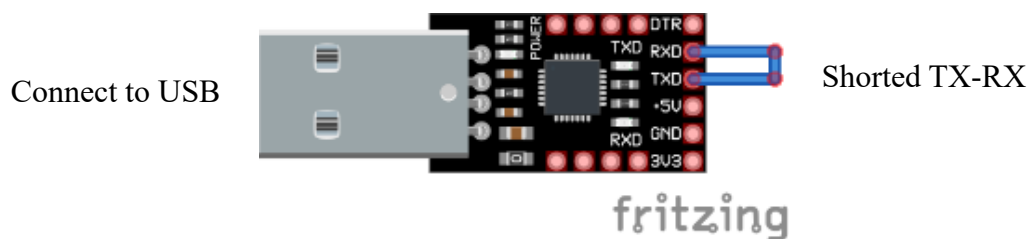
1. <https://learn.adafruit.com/getting-started-with-binho-nova/quickstart-with-coolterm>
2. Brief Steps:
 - Step 1. Plug in the UART-to-USB converter in the computer. Open **Device Manager** check COM port availability.
 - Step 2. Open CoolTerm – click **Options** – set **Serial Ports** - COM port, baud rate, data bits, parity, stop bits
 - Step 3. Click **Connect**. You are ready to connect
 - Step 4. Click **Connection** | Send String. Select Plain or Hex.



12.1.1 Self-test With UART-to-USB

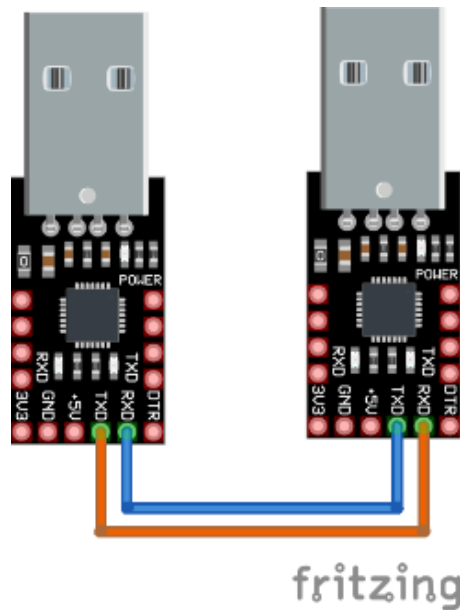
Connect TX to RX. Plug in the converter in the computer then open serial terminal app. Do send anything, if the converter is working and the connection is correct, you should receive the return data.

- Note: The converter must be detected and listed in Device Manager. Use the same configuration settings as listed in Device Manager. Example, 9600, 8N1.



12.1.2 Self-test With Two Units of UART-to-USB

Imagine that two serial devices are communicating. Connect RX to TX and TX to RX. Open two serial terminal apps and select a COM port for each converter. Simulate Read/Write data.



12.2 Flashing Using UART Port

There are a number of options to flash an MCU. One way is to flash via UART1 port.

Note: This differs from using ST-Link V2 using the SWD (Serial Wire Debug) interface or the full JTAG interface.

Download “Free STM32 ISP software” from

<http://www.mcuisp.com/English%20mcuisp%20web/ruanjianxiazai-english.htm>

This method deploys the bootloader located the system memory. Refer to the BOOT modes.

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

(Source: RM0008 Reference manual)

Criss-crossed connection of [TX, RX] between UART-to-USB and UART1 as shown in the figure below.

(MCU must be powered. Not Shown in the diagram)

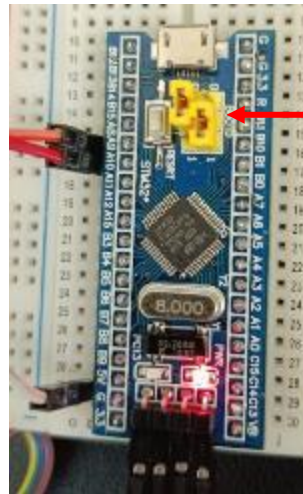


- Step 1. Identify connection and COM port name – check Device Manager
- Step 2. At MCUISP, choose the COM port. Leave the settings as it is.
- Step 3. At Codes Files For Online ISP – choose the hex file.
- Step 4. Hardware. Switch jumper **BOOT0** → **1**. Press reset button. MCU would switch to System memory - Bootloader.

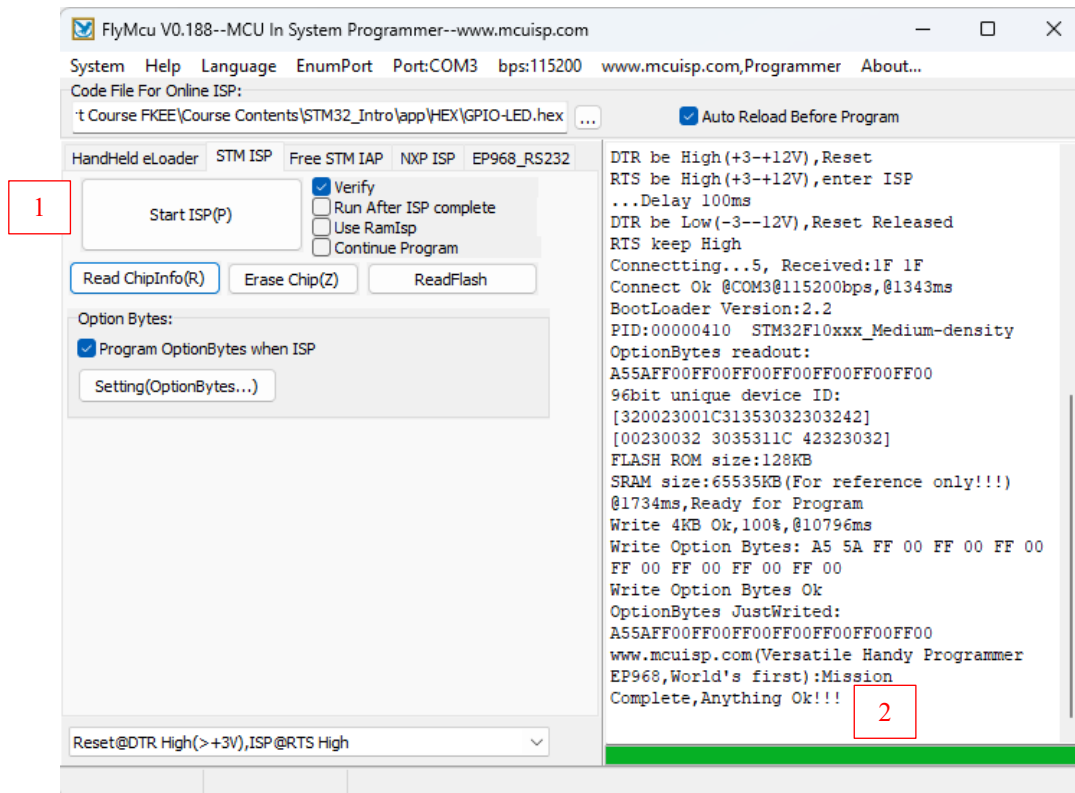
- Step 5. At MCUISP, click the button 1 – Start ISP(P). Wait till the operation is completed 2.
- Step 6. Hardware. Switch jumper BOOT0 to 0. Press reset button. MCU would switch to Main Flash memory.

Info:

Switch BOOT0 from 0→1→0 is done manually for this training board (Blue Pill – STM32F103C8T6) because no flow control circuitry is designed. Other development with flow control (DTR, RTS) circuitry, switching between Main Flash memory and System memory is completed electronically.



BOOT0 → 1
Press Reset



12.2.2 Procedures to Read/Read ChipInfo

The same procedures as Write.

For Read, save Bin file in location.

12.3 ST Tools

Some tools provided by ST.

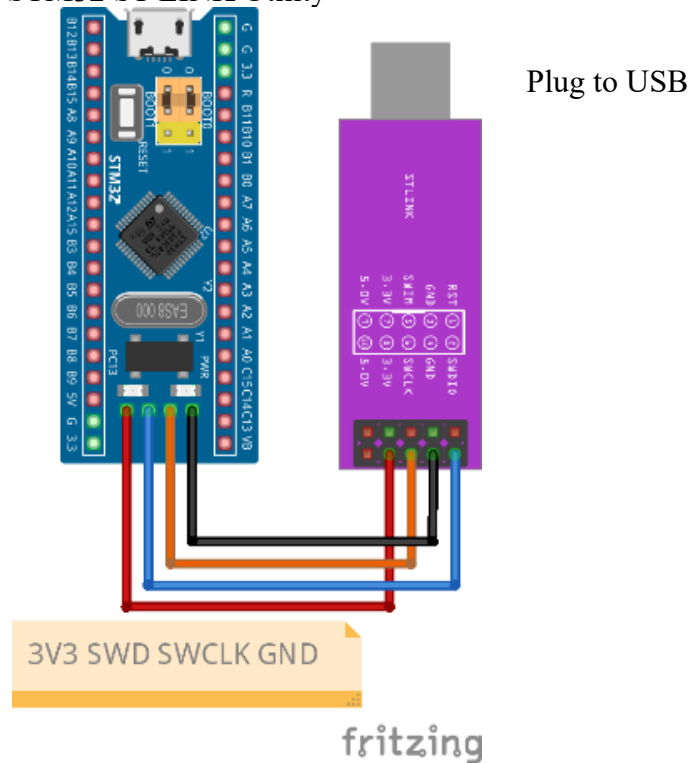
1. STM32 Flash loader demonstrator
<https://www.st.com/en/development-tools/flasher-stm32.html#get-software>
2. STM32CubeProgrammer software for all STM32
<https://www.st.com/en/development-tools/stm32cubeprog.html#get-software>

13 STM32 ST-LINK Utility

Alternatively, we could use **STM32 ST-LINK Utility** (STSW-LINK004) via the SWD interface. It is a full-featured software interface for programming STM32 microcontrollers. Download the software and install it on the computer (Windows).

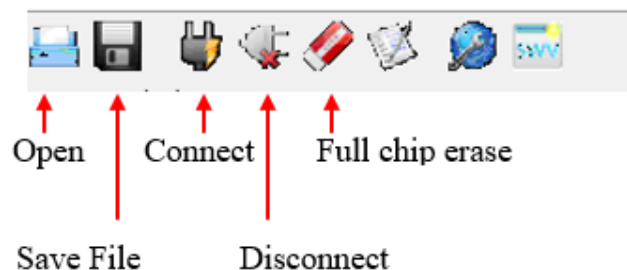
<https://www.st.com/en/development-tools/stsw-link004.html>

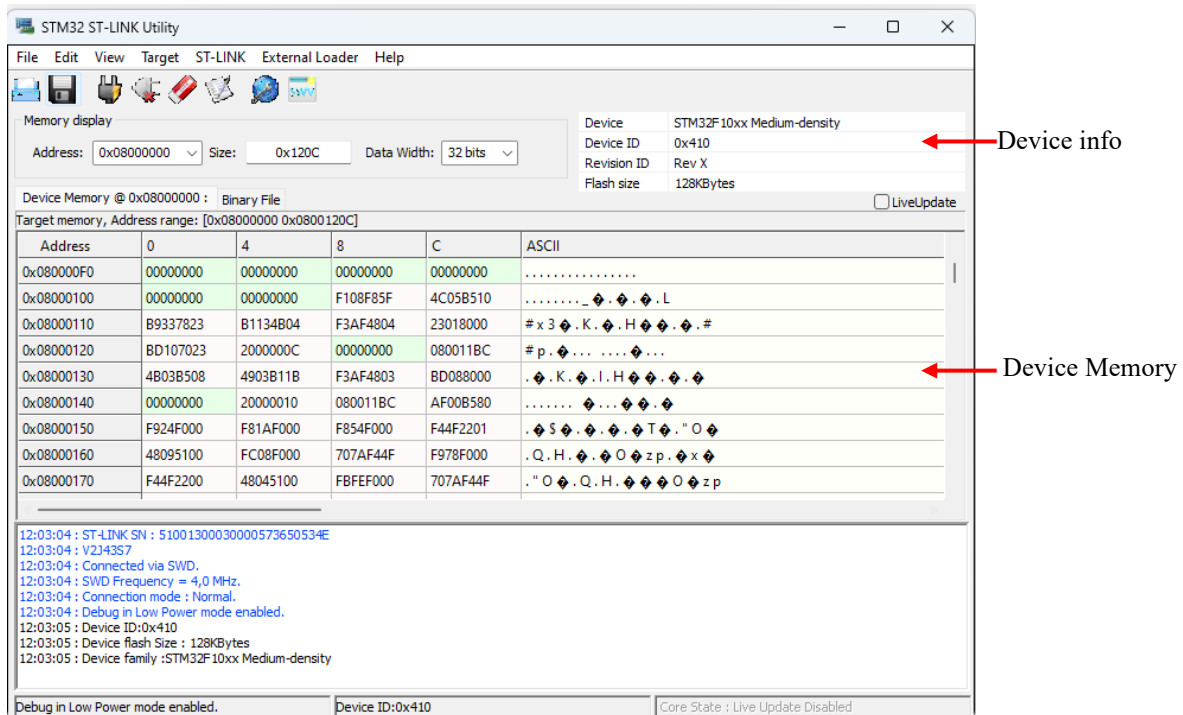
1. Connect ST-Link V2 as shown. [3V3, SWD, SWCLK, GND]
2. Plug in ST-Link V2 to the USB port. At Device Manager, under Universal Serial Bus devices | STM32 STLink should be detected. The MCU is powered up.
3. Open the software – STM32 ST-LINK Utility



13.1 Write HEX / Bin File to MCU

Step 1. At the app, click connect.





- Step 2. Open and choose a Hex file.
- Step 3. At menu, choose Target | Program & Verify. At pop-up, choose Start.
- Step 4. Observe the MCU.

13.2 Read HEX / BIN File and Erase Full Chip

The same procedures.

Save read into HEX or BIN file in location.

For full chip erase, the device memory would be set to 0xFF.

13.3 Why not STMCubeProgrammer?

ST-Link used in practices is a third-party clone version; Latest STMCubeProgrammer would not detect the serial number of this debugger/programmer.



Source: [stlink-debugging-and-programming-tools-overview.pdf](#)

STLINK portfolio

Debugging & programming

STLINK-V3MINIE



STLINK-V3PWR



ST-LINK/V2



STLINK-V3SET



STLINK-V3MODS



..and expansion boards!

Comments: