

16-bits PIC/dsPIC[®] MCU
Microcontroller & MCC
PIC2001 v1.00



A Leading Provider of Smart, Connected and Secure Embedded Control Solutions



SMART | CONNECTED | SECURE

CAE Taiwan Team
Microchip Technology Inc.

November 10, 2022

Major Course (1/3)

- Microchip 16-bits Microcontroller Introduction
- MPLAB® X IDE, Compilers, MCC and Development Tools Introduction
- APP053 EVB Introduction
- Getting Started with First Project
 -  Lab0 – First Project
- GPIO Architecture
 -  Lab1 – Multi-GPIO Output
 -  Lab2 – GPIO Input
- Oscillator Architecture
 -  Lab3 – Clock PRIPLL
- Interrupt Architecture

Major Course ^(2/3)

- Timer Architecture

-  Lab4 – Timer1 Interrupt

- OLED Architecture(PPS&SPI)

-  Lab5 – OLED Display

- UART Architecture

-  Lab6 – UART Printf

- ADC Architecture

-  Lab7 – Multiple Channel Periodically

- External Interrupt Architecture

-  Lab8 – Multi-Ext Interrupt

- High Resolution PWM Architecture

-  Lab9 – PWM Output

Major Course (3/3)

- MCCP and SCCP Architecture



Lab10 – SCCP Compare/PWM Interrupt

- 2D Sketch Drawing Application

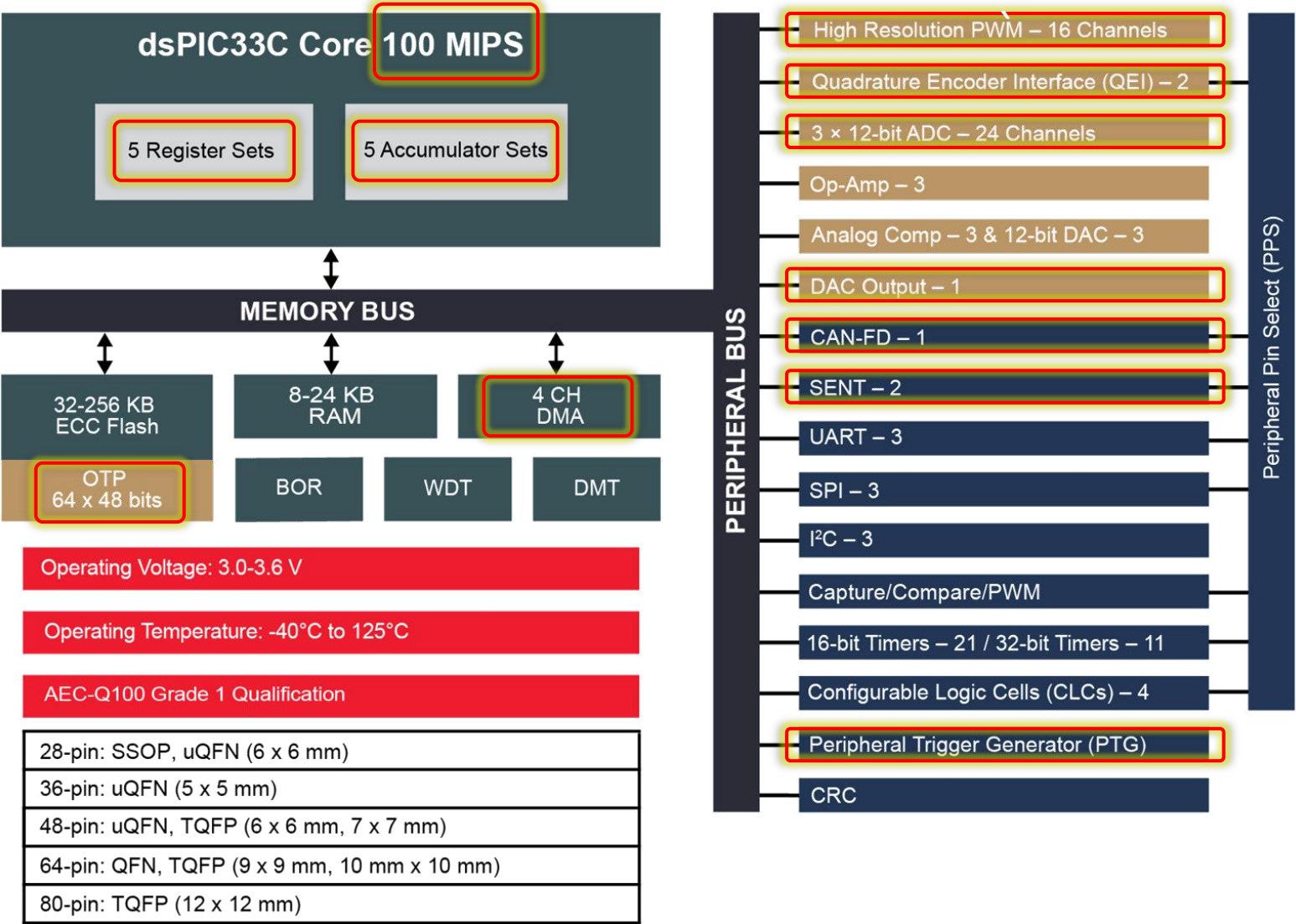
- G-code 3D Model and Slice

Microchip 16-bits Microcontroller Architecture

16-bits PIC/dsPIC® MCU series

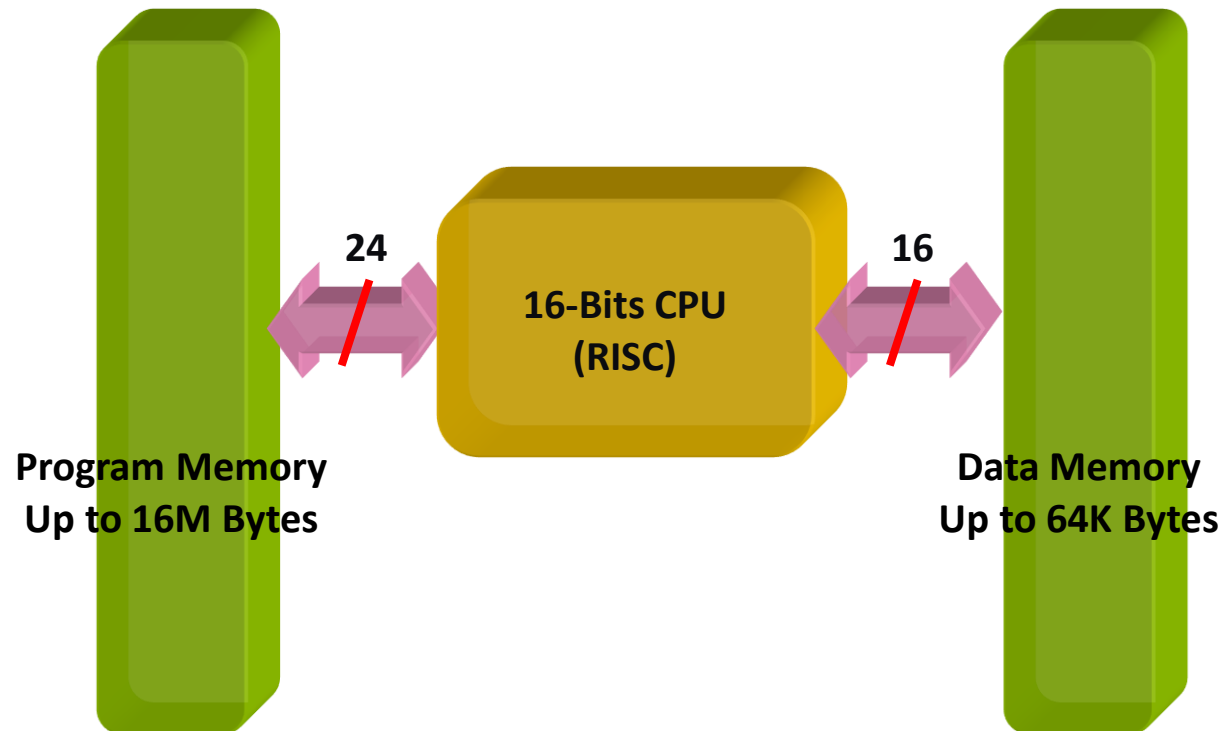
- **Microchip 16-bits MCU families include below,**
- PIC24FJ (16 MIPS)
 - Very high CP. Low price and high performance.
- PIC24HJ (40 MIPS)
 - Built-in DMA, upgrade performance to 40 MIPS.
- PIC24EP (70 MIPS)
 - Latest generation of PIC24 series, upgrade performance to 70 MIPS.
- dsPIC30F (30 MIPS)
 - Base on PIC24FJ, Built-in DSP.
- dsPIC33FJ (40 MIPS)
 - Base on PIC24HJ, Built-in DSP.
- dsPIC33EP (60 MIPS)
 - Built-in DSP & USB OTG, upgrade performance to 60 MIPS.
- **dsPIC33CK (100 MIPS)**
 - High Performance, 12-bits High Speed ADC & High-Resolution PWM.

dsPIC33CK256MP508 Family

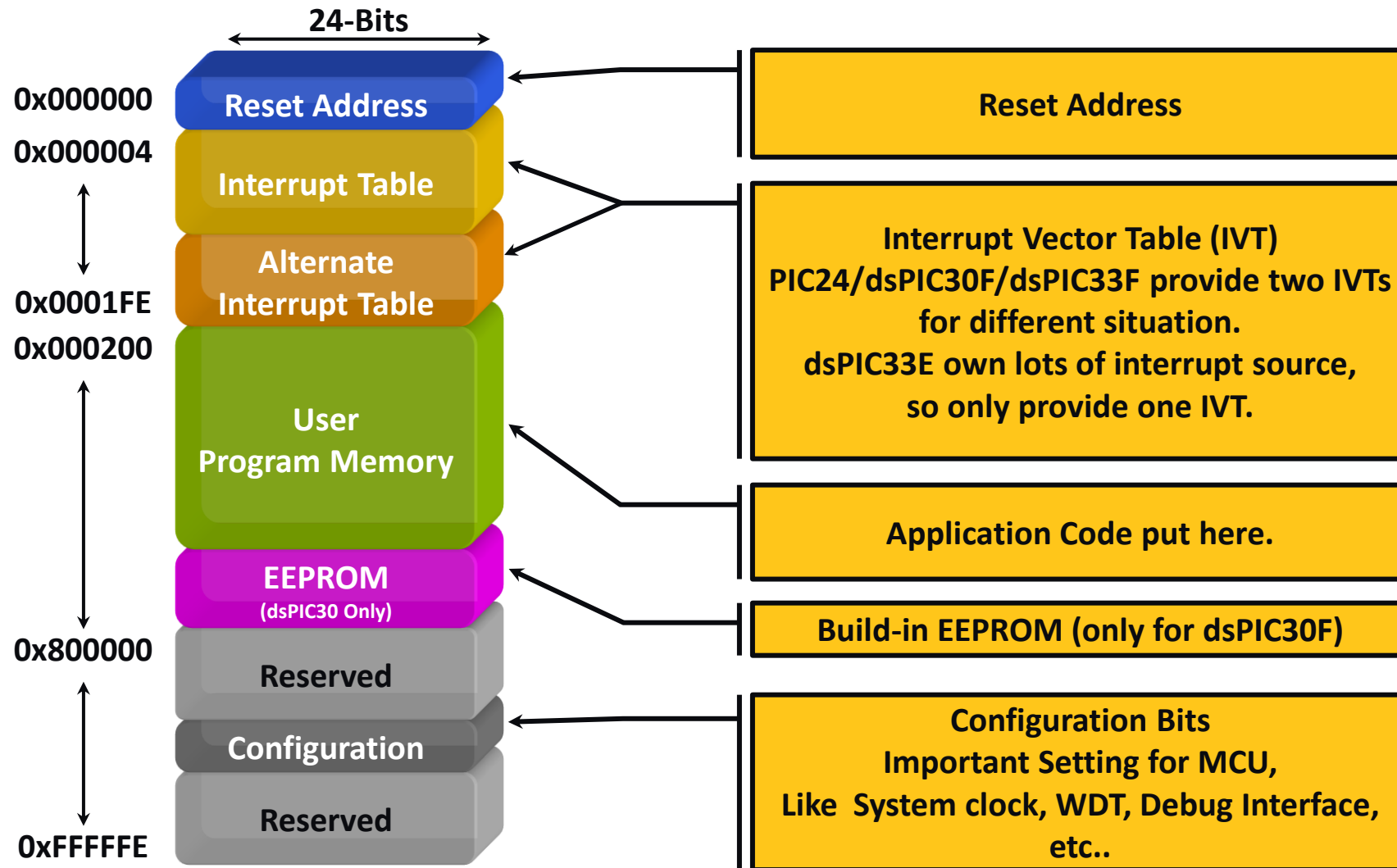


Core Architecture

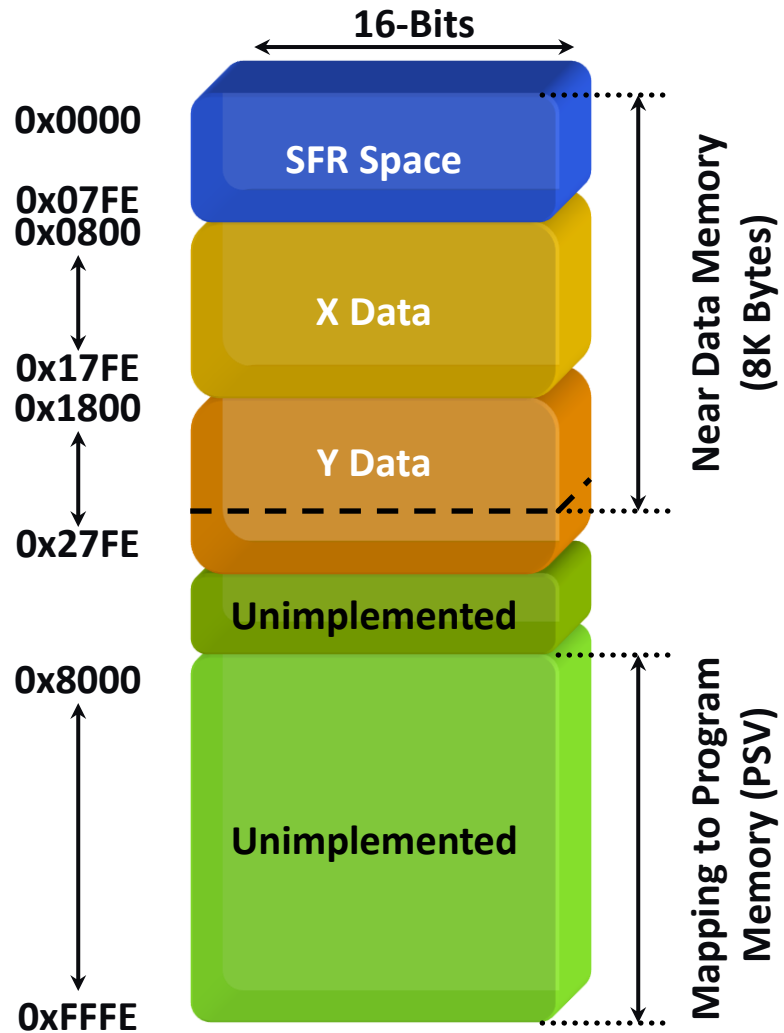
- Microchip 16-bits PIC/dsPIC® MCU is **Harvard architecture**. This means program and data memory bus are independently.
- Data Memory is **16-bits, Up to 64K Bytes**.
- Program Memory is **24-bits, Up to 16M Bytes**.



Program Memory Mapping



Data Memory Mapping



The 8-Kbyte area is referred as the near data space.

The 64-Kbyte area is referred as the far data space.

The first 2 Kbytes are primarily occupied with Special Function Registers (SFRs).

X Data, Y Data area only for DSP Engine
For Dual Access Operation.
All Data is X Data area for PIC24 series.

0x8000 ~ 0xFFFF is mapping area for PSV use.

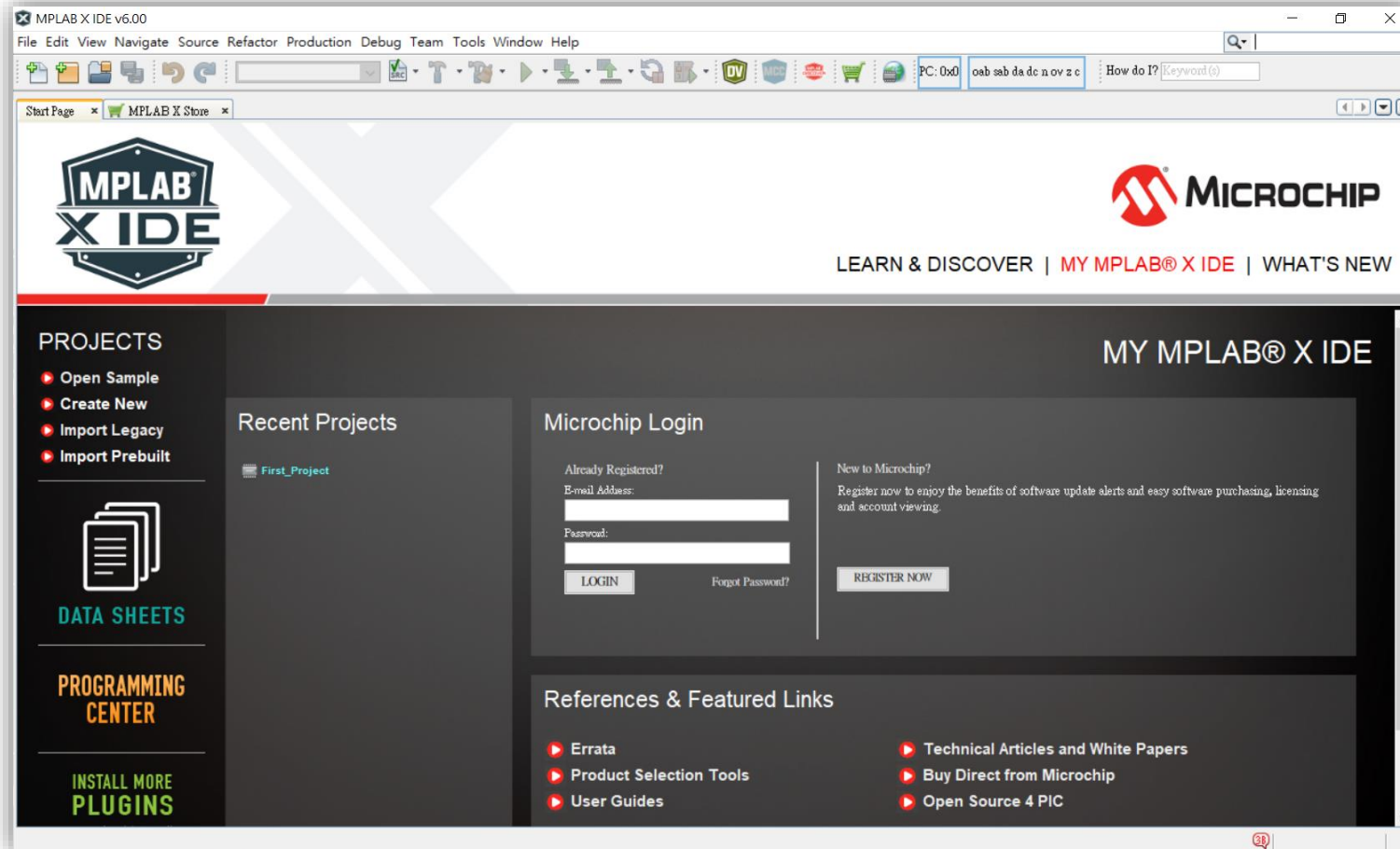
Machine & Instruction Cycle

- **T_{osc} (Machine Cycle) :**
 - A machine cycle consists of the steps that a computer's processor executes whenever it receives a machine language instruction. (techopedia)
- **T_{cy} (Instruction Cycle) :**
 - The basic operational process of a computer system.(wiki)
- **dsPIC30 Family**
 - $1 T_{cy} = 4 T_{osc} \cdot (F_{cy} = F_{osc} / 4)$
Ex:120 MHz => 30 MIPS, $T_{cy} = 33nS$.
- **PIC24F/PIC24H/dsPIC33 Families**
 - $1 T_{cy} = 2 T_{osc} \cdot (F_{cy} = F_{osc} / 2)$
Ex:32 MHz => 16 MIPS, $T_{cy} = 62.5nS$.

IDE, Compiler, MCC & Development Tools Introduction

All in One Development Tools

- New generation Integrated Development Environment, Support PIC, sic, AVR, CEC and SAM Series MCU/MPU, Provide Plug-in function to extend more advance feature. Java Based, Cross platform, latest version is v6.00.



C Compiler for 16-bits PIC/dsPIC® MCU

- XC16 is new generation C compiler, it's supported all 16-bits PIC/dsPIC® MCU (PIC24F/PIC24H/PIC24E, dsPIC).
- Base on GNU C, apply GPL License (GNU General Public License).
- Provide standard C libraries (printf, strlen, etc..) and Peripheral Libraries (old versions).
- All version you can download from Microchip website. It's provided free version.

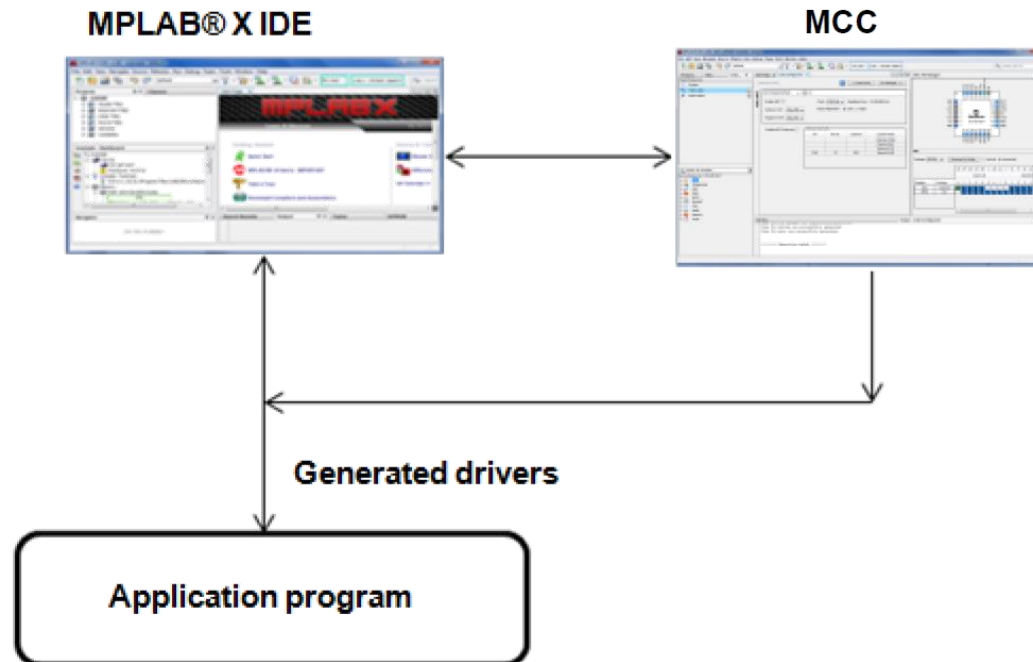
License Type	Installs On	# of Activations	# of Users	Wait Time Between Users	HPA
Workstation License	Workstation	3	1	None	Yes
Subscription License	Workstation	1	1	None	No
Site License	Network	1	Varies by Seat	None	Yes
Network Server License	Network	1	Unlimited	One Hour	Yes
Virtual Machine *	Network	1	N/A	N/A	No
Dongle License	Dongle	N/A	Unlimited	None	No

*This license must be used in addition to a network server or site license to enable the license to work in a virtual machine environment.



MPLAB® Code Configurator

- MPLAB® Code Configurator - MCC is a free, graphical programming environment that generates seamless, easy-to-understand C code to be inserted into your project.
- Supports 8-bits, 16-bits and limited 32-bits PIC® microcontrollers.



MCC's GUI Quick View

The screenshot displays the MPLAB X IDE v6.00 interface with the MCC (Microchip Configuration Center) GUI Quick View. The interface is divided into several panes:

- Project Resource Manager:** Located on the left, it shows the project structure with tabs for Projects, Files, Classes, Services, and Resource Manager. The Resource Manager tab is active, displaying a tree view of System and Device Resources. A yellow callout box labeled **Resources** points to this pane.
- System Module:** The central pane shows the configuration for the selected system module. It includes sections for Clock, FRC Postscaler, PLL Enable, and Auxiliary Clock. A yellow callout box labeled **Composer** points to this pane.
- Pin Manager:** Located on the right, it displays a pin diagram for the dsPIC33CK256MP506. A yellow callout box labeled **Pin Manager** points to this pane.

At the bottom, there is a table with columns for Category, Source, Type, and Description. The table is currently empty, with the text "No content in table" displayed below it.

MCC's GUI Quick View

MPLAB X IDE v6.00 - First_Project : default

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Search (Ctrl+I)

PC: 0x0 oab sab da dc n ov z c How do I? Keyword(s)

Projects Files Classes Services Resource Manager

Project Resources

- System
 - Interrupt Module
 - Pin Module
 - System Module

Device Resources

- Peripherals
 - ADC
 - CAN FD
 - CBG
 - CLC
 - CMP_DAC
 - CRC
 - DMA
 - Ext_Interrupt
 - High Resolution PWM

First_Project - Dashboard Core Versions [MCC]

MCC Core Versions

- MPLAB® Code Configurator (Plugin) v5.1.9
 - Core v5.4.4 [C:\Users\A62407\mcc\cores\5.4.4\co
 - Core v5.4.3 [Available for download.]
 - Core v5.4.2 [Available for download.]
 - Core v5.4.1 [Available for download.]
 - Core v5.4.0 [Available for download.]
 - Core v5.3.3 [Available for download.]
 - Core v5.3.0 [Available for download.]
 - Core v5.2.1 [Available for download.]
 - Core v5.1.0 [Available for download.]
 - Core v5.0.1 [Available for download.]

System Module

Easy Setup Registers

▼ Clock

8000000 Hz FRC Oscillator (8.0 MHz) Clock Source

▼ ☒ FRC Postscaler

8 MHz 1:1 Postscaler

☐ PLL Enable

8 MHz Fosc

4 MHz Fosc/2

▼ ☒ Auxiliary Clock

8000000 Hz FRC Clock Source

8 MHz ☐ PLL Enable

8 MHz 1:1 Prescaler

1200 MHz 1:150 Feedback

Pin Manager: Package View

dsPIC32CK256MP506

System clock Graphically.
Configuration Word settings
are clearer and simpler.

MCC's GUI Quick View

The screenshot shows the MPLAB X IDE v6.00 interface. The 'Pin Manager: Grid View' window is highlighted with a red border. It displays a grid for assigning pins to modules. The grid is organized by Package (TQFP64), Pin No., and Port (Port A and Port B). The modules listed are Clock, ICD, and Pin Module. The Pin Module is selected, and its GPIO pins are being assigned to the output direction. A purple arrow points from a yellow callout box to the grid.

**Pin assign Graphically.
PPS Code generate automatically.**

Package:	TQFP64	Pin No:	14	15	16	17	18	28	29	33	34	35	45	46	47	48	49	61	62	63	64	1	2	13	22	23	27	50	51	24
			Port A								Port B																			
Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6
Clock	CLKI	input						🔒																						
	CLKO	output							🔒																					
	OSCI	input						🔒																						
	OSCO	output							🔒																					
	REFI1	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ICD	PGCx	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	PGDx	input							🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒

MCC's GUI Quick View

The screenshot shows the MPLAB X IDE v6.00 interface. The 'Interrupt Module' window is open, displaying the 'Interrupt Manager' section. The 'Enable Global Interrupts' checkbox is checked. Below it is a table listing interrupt modules and their configurations. A purple arrow points from a yellow callout box to the table, indicating that interrupt assignment is done graphically and ISR code is generated automatically.

Interrupt Manager

☒ Enable Global Interrupts

Module	Interrupt	Description	IRQ Num...	Enabled	Priority	Context
Pin Module	CNAI	Change Notification A	2	☐	1	OFF
Pin Module	CNBI	Change Notification B	3	☐	1	OFF
Pin Module	CNCI	Change Notification C	19	☐	1	OFF
Pin Module	CNDI	Change Notification D	75	☐	1	OFF
DMT	DMTI	Dead Man Timer	45	☐	1	OFF

Interrupt assign Graphically. ISR related code generate automatically.

More ?

www.microchip.com/mcc

Programmer/Debugger

ICD4 (Part No. DV164045)

- The MPLAB® ICD 4 In-Circuit Debugger/Programmer is Microchip's fastest, cost-effective debugging and programming tool for PIC®, dsPIC®, AVR, SAM and CEC flash microcontrollers.
- **Features**
 - Full-Speed Real-Time Emulation
 - Target voltage of 1.20V to 5.5V.
 - Can supply up to 1A of power to the target
 - Overvoltage/ Over-current protection
 - Supports multiple breakpoints, stopwatch and source code file debugging
 - Selectable pull-up/pull-down option to the target interface



\$328.89

Programmer/Debugger

PICKit4 (Part No. PG164140)

- The MPLAB® PICKit™ 4 In-Circuit Debugger/Programmer allows fast and easy debugging and programming of all PIC®, dsPIC®, AVR, SAM and CEC flash microcontrollers.
- **Features**
 - Matches silicon clocking speed.
 - Target voltage of 1.20V to 5.5V.
 - Can supply up to 50mA of power to the target.
 - Minimal current consumption at <100µA from target.
 - Portable USB-powered and RoHS-compliant.
 - 8-pin single in-line header.
 - Backward compatible for demo boards, headers and target systems using 2-wire JTAG and ICSP.
 - Option to be self-powered from the target (2.7V to 5.5V).

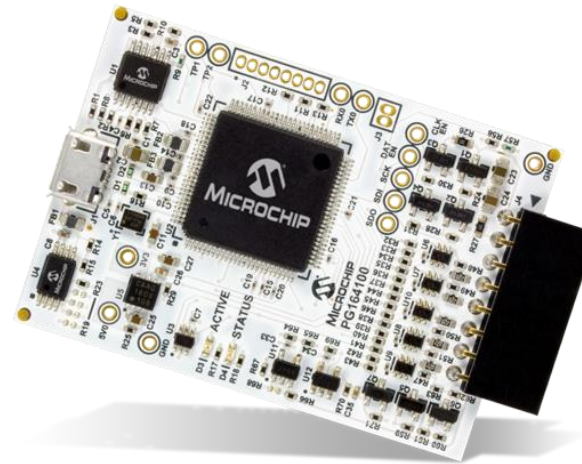


\$76.99

Programmer/Debugger

SNAP (Part No. PG164100)

- The MPLAB® SNAP In-Circuit Debugger/Programmer allows fast and easy debugging and programming of most PIC®, dsPIC®, AVR, SAM and CEC flash microcontrollers.
- **Features**
 - Affordable performance
 - Matches silicon clocking speed
 - Target voltage of 1.20V to 5.5V
 - Portable USB-powered
 - CE and RoHS-compliant
 - 8-pin single in-line header
 - Backward compatible for demo boards, headers and target systems using 2-wire JTAG and ICSP



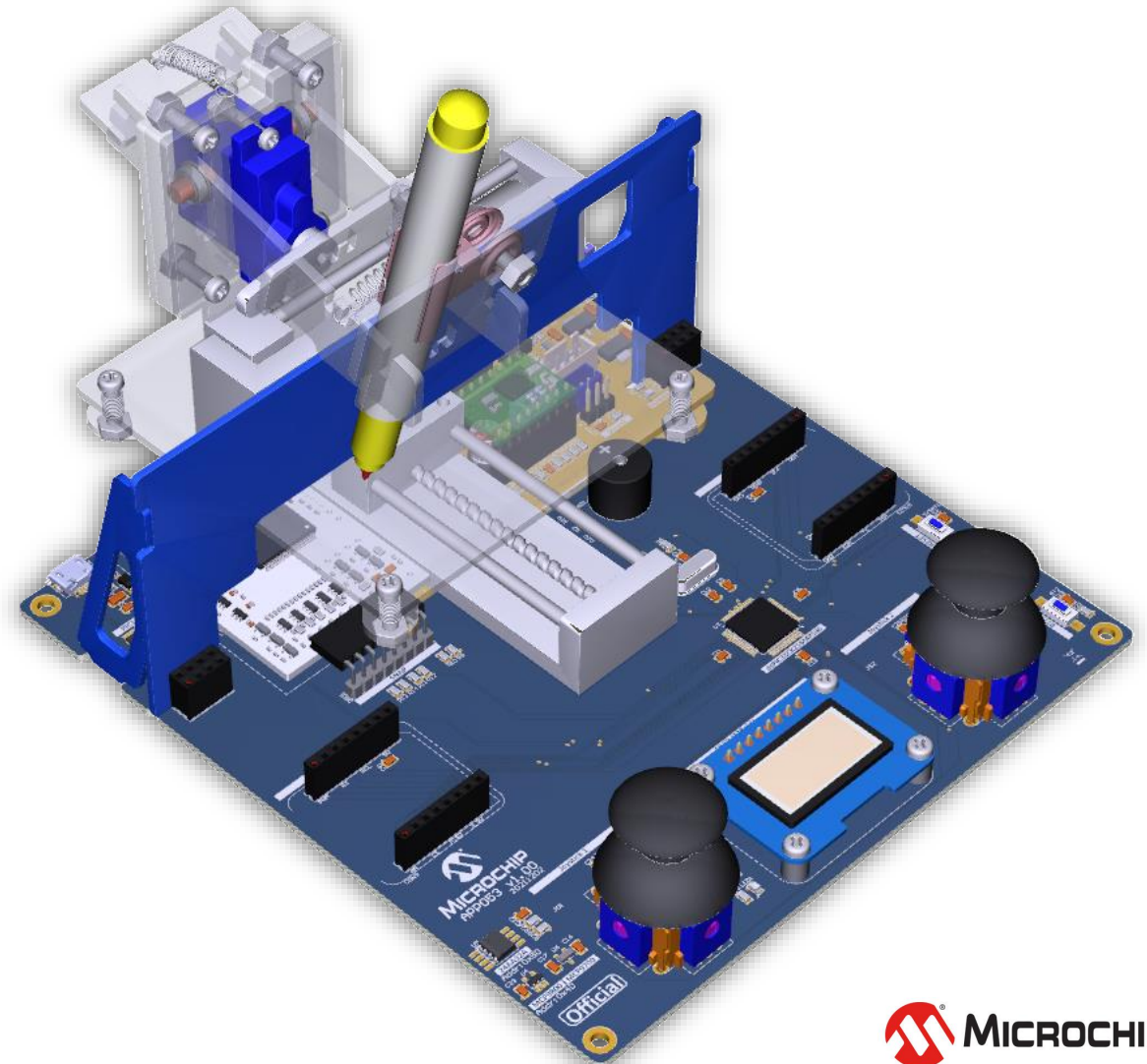
\$34.09

Free !Only For Elite

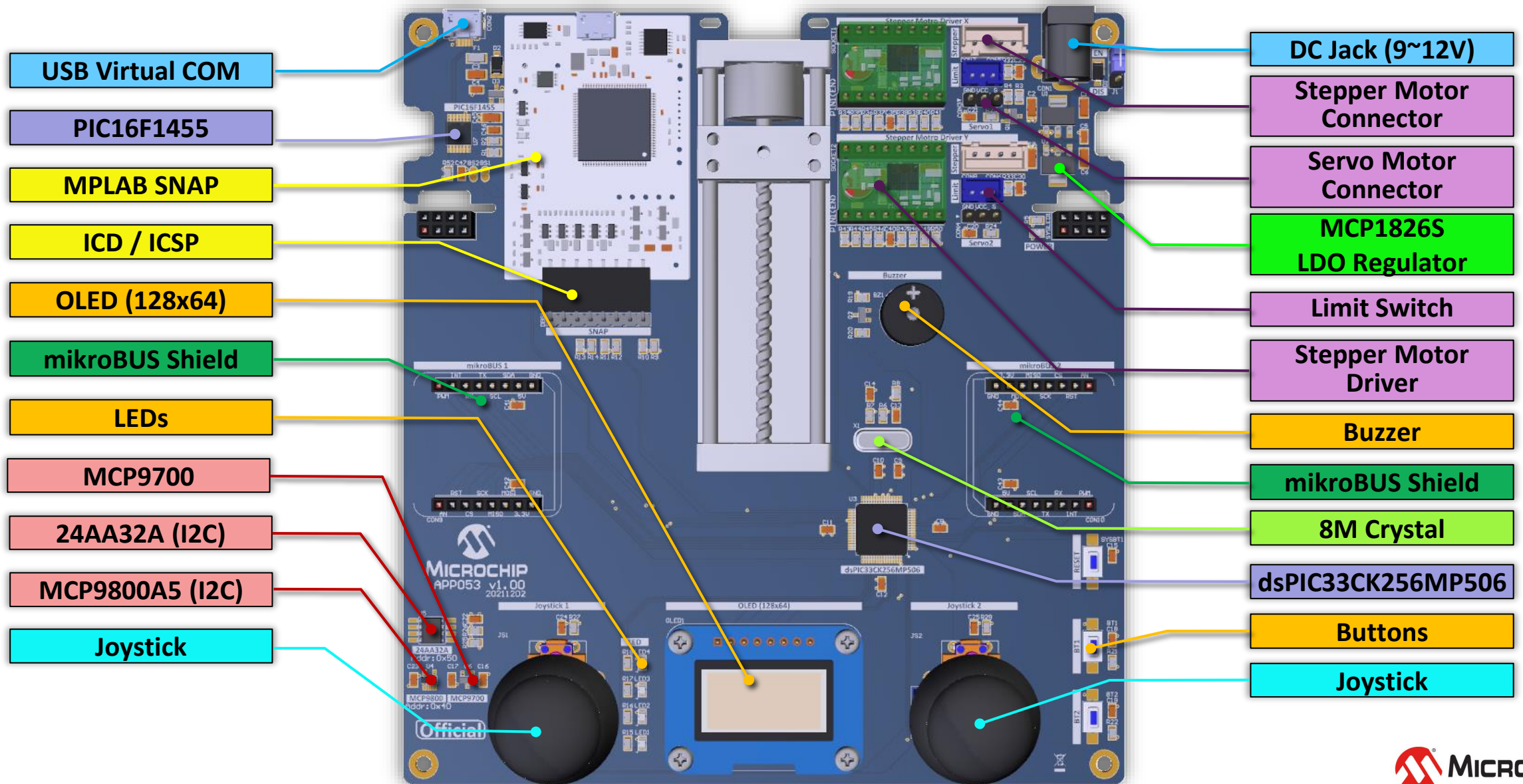
APP053 EVM Introduction

APP053 v1 Features

- On board dsPIC33CK256MP506 16Bit Microcontroller
- Support plugged MPLAB SNAP
- Onboard USB CDC Virtual COM Port
- Multiple Power Supply Source
- 2 x Buttons, 4 x LEDs, 1 x Buzzer
- 1 x Analog Temperature Sensor
- 1 x I2C Temperature Sensor
- 1 x I2C EEPROM with Unique ID
- 1 x SPI OLED (128 x 64, SSD1306 Compliant)
- 2 x mikroBUS™ Click Board Socket
- 2 x Joystick
- 2 x Stepper Motor Driver
- 2 x Servo Motor Connector
- **APP053+ Added Mechanical Assembly Kit.**

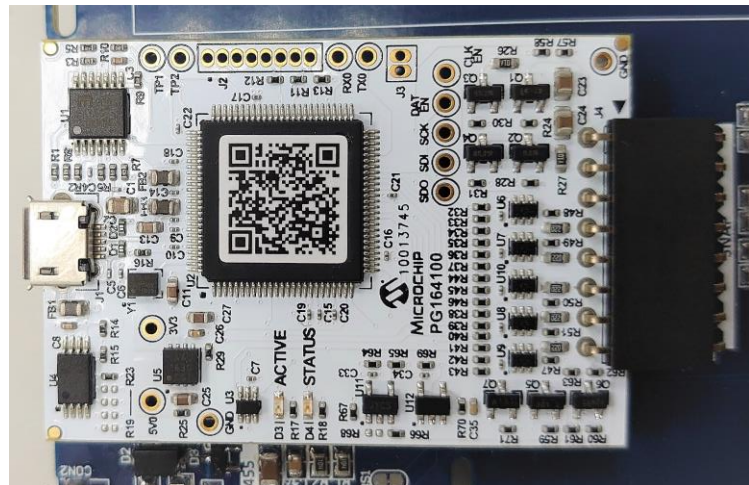
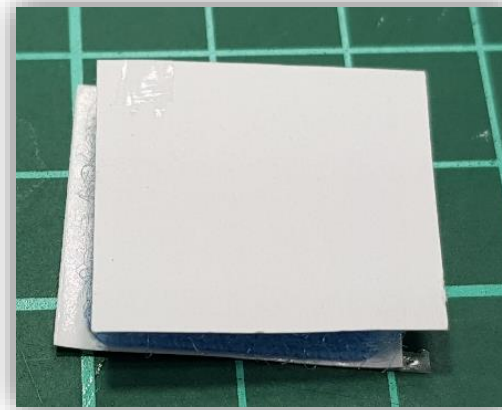
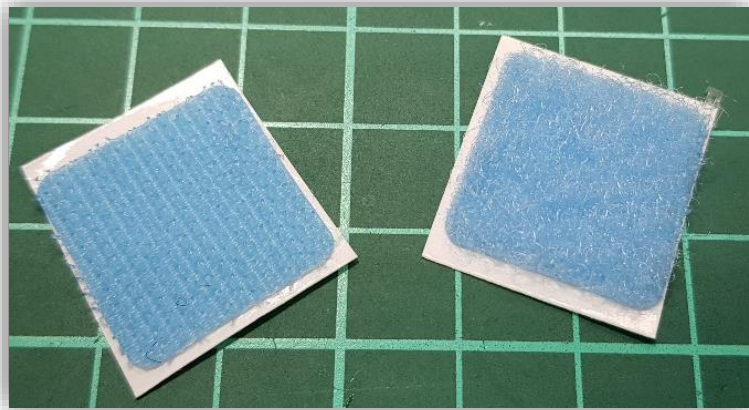


APP053 v1 Overview



MPLAB® SNAP Setting

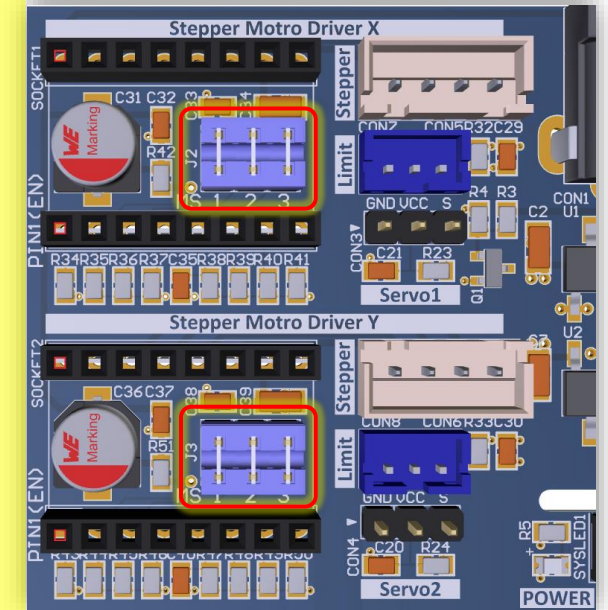
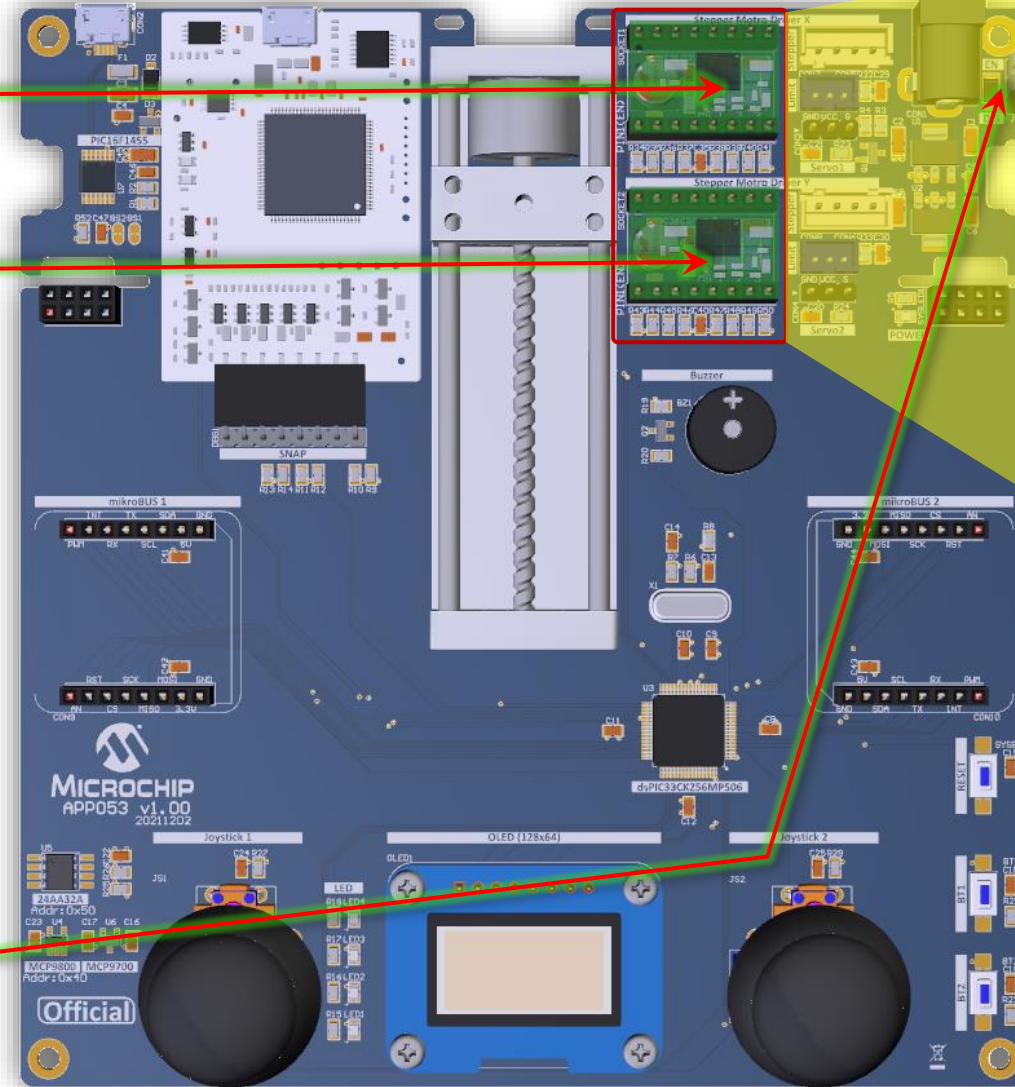
- MPLAB® SNAP can be assembled on APP053 v1.00.
- Suggest to use Velcro for stability.



Caution !

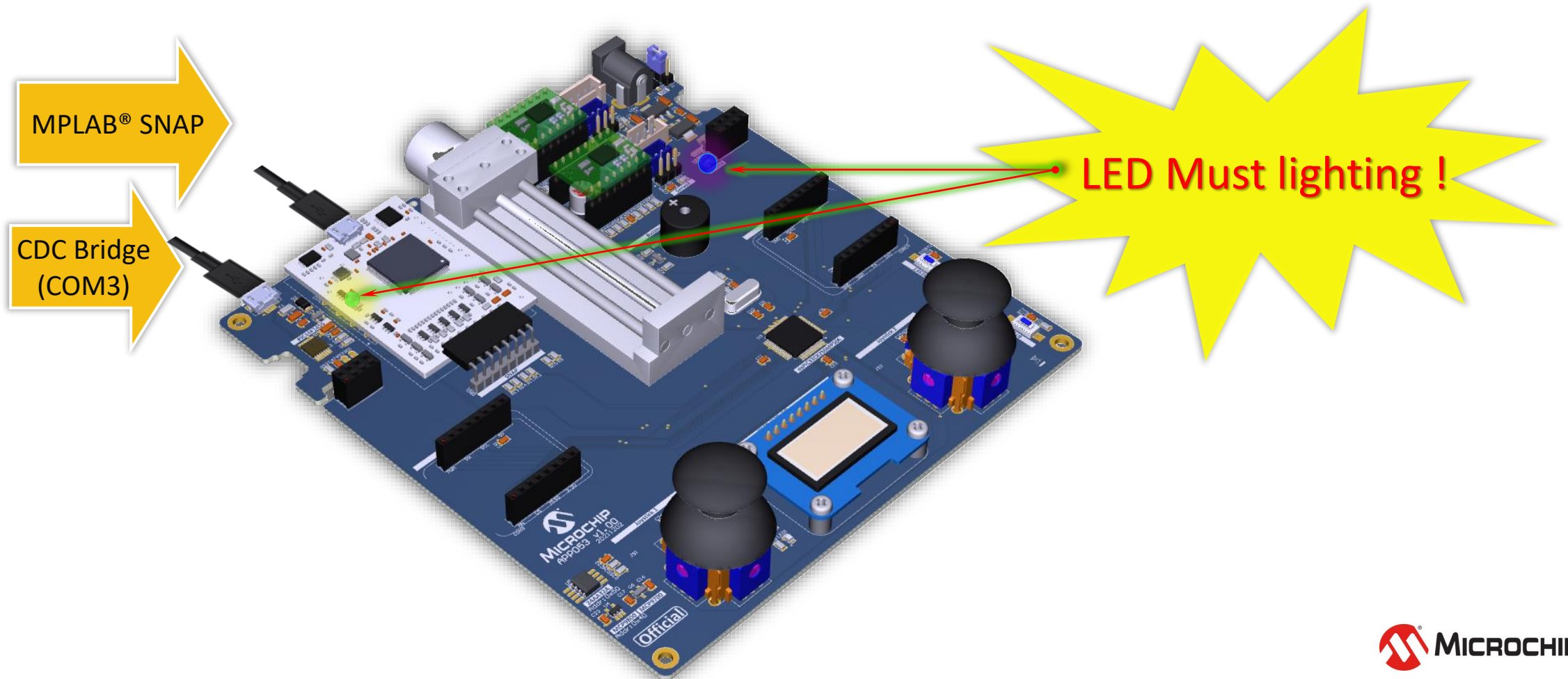
- You must check all jumpers position before power on. Wrong setting will damage silicon and your power source.

J2P	All closed (Full Step)
J3P	
J4P	
J5P	All closed (Full Step)
J6P	
J7P	
J1P	1-2 closed (EN)



Power On & Getting started

- Just plug-in micro-USB cable to COM3 and MPLAB® SNAP then confirm power LED (SYSLED1) and ACTIVE LED on MPLAB® SNAP are light or not.
- OLED/LCM will show preloaded demo firmware.



First MCC Project

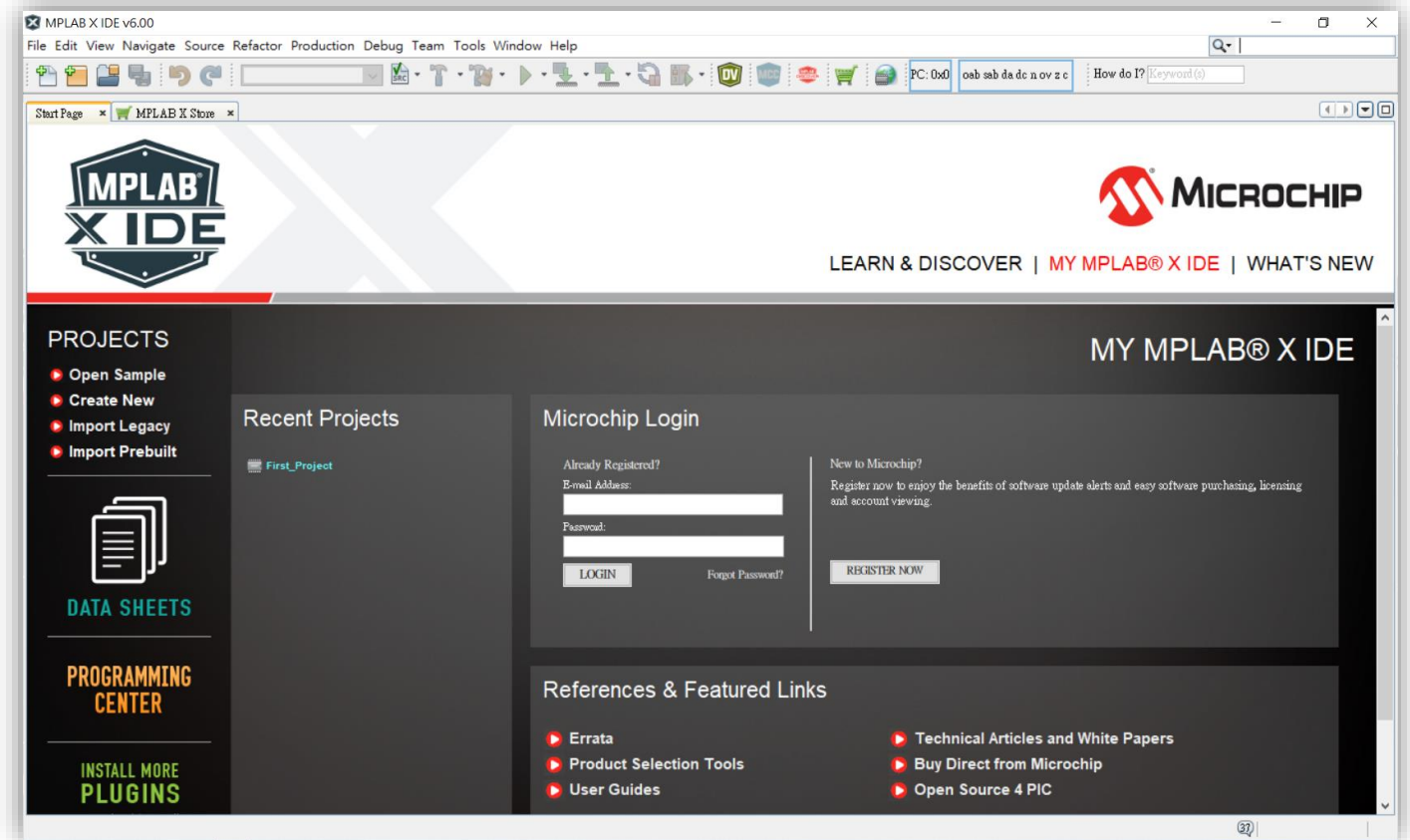
Lab Preparation

Add MCC Libraries
Check Compiler & MCC version

Lab Preparation

Step 1

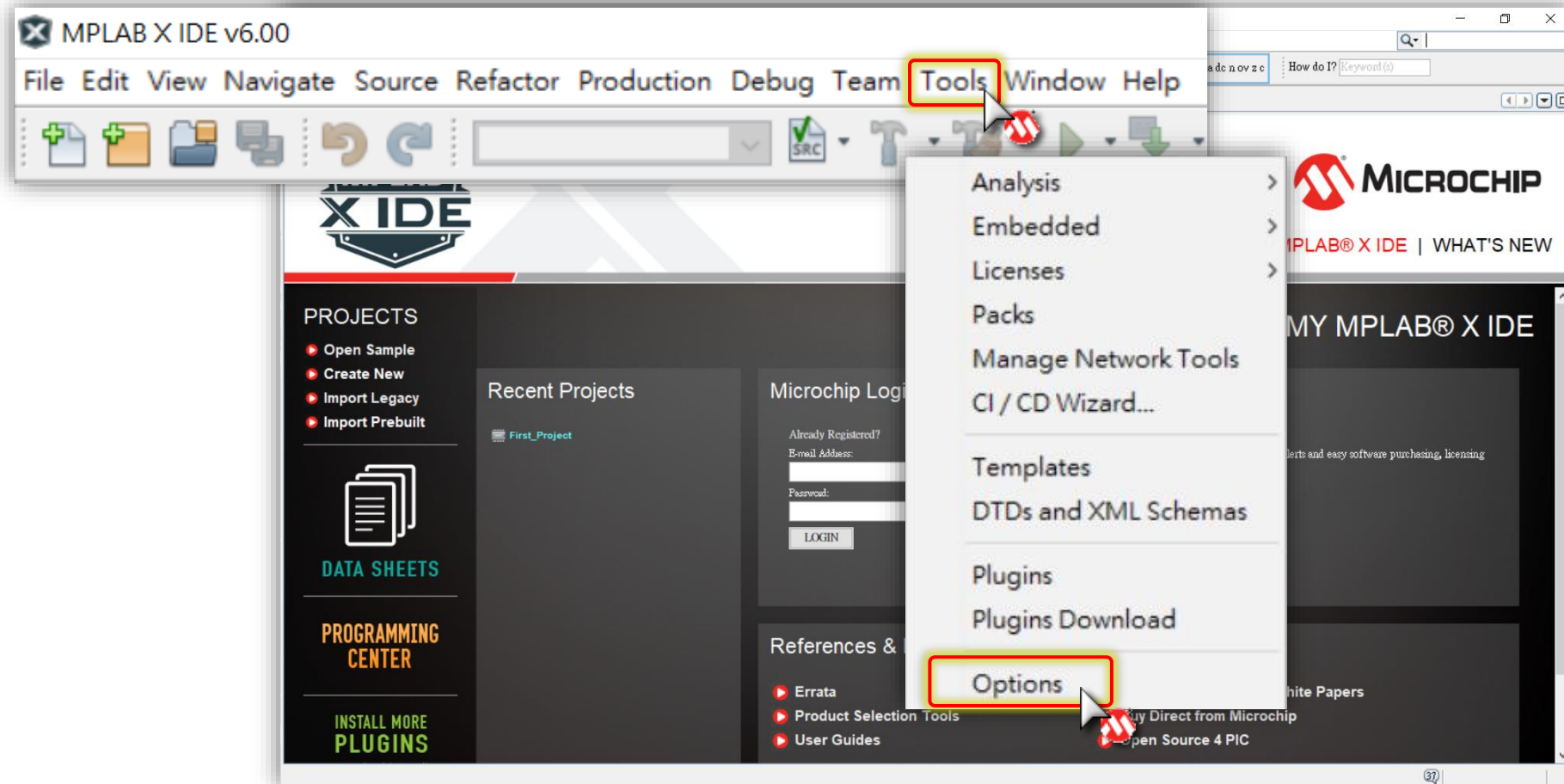
- Open **MPLAB X IDE v6.00** firstly.



Lab Preparation

Step 2

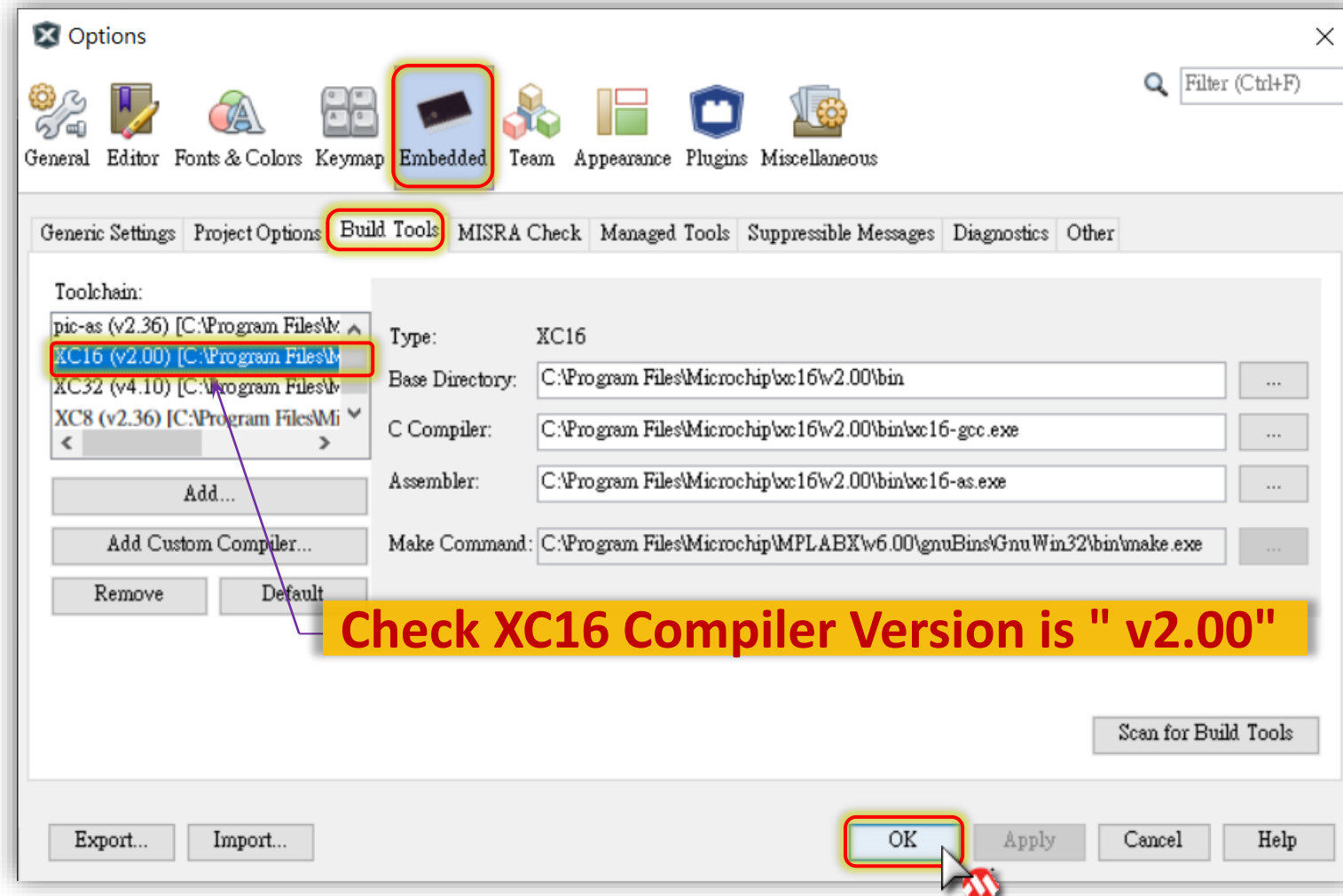
- Select : **Tools > Options**



Lab Preparation

Step 3

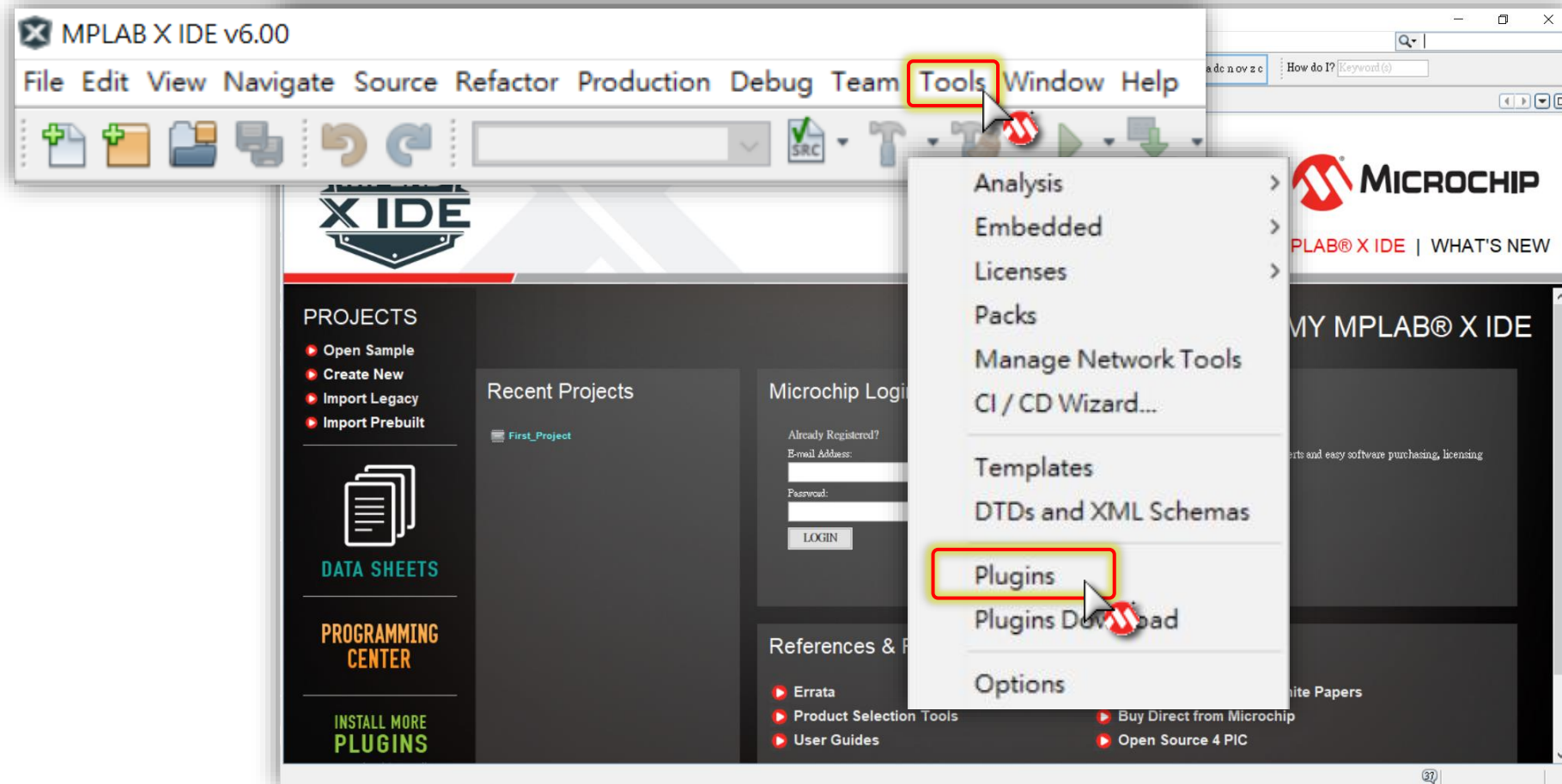
- Select : **Embedded > Build Tools**



Lab Preparation

Step 4

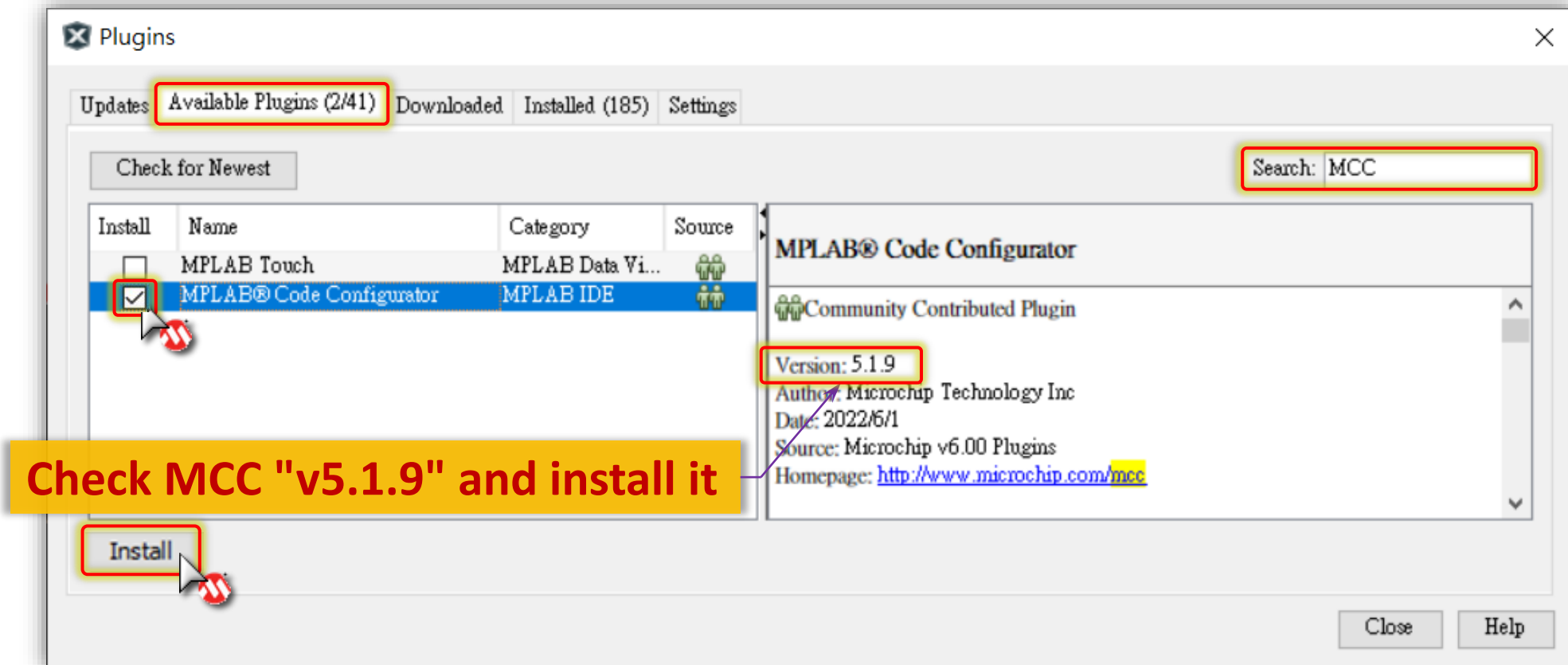
- Select : **Tools > Plugins**



Lab Preparation

Step 5 (MCC not installed yet)

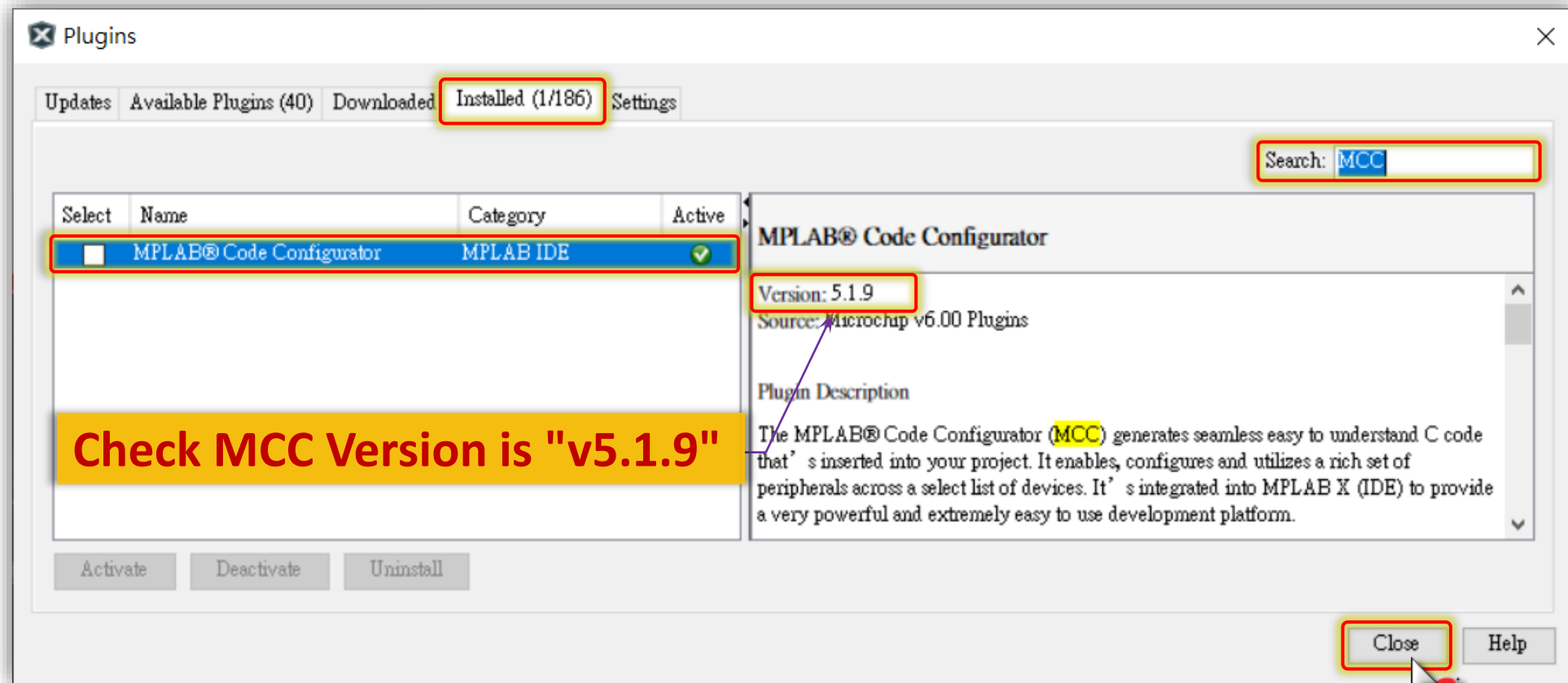
- Select : **Available Plugins**
 - Search : "**MCC**"



Lab Preparation

Step 5 (MCC has installed)

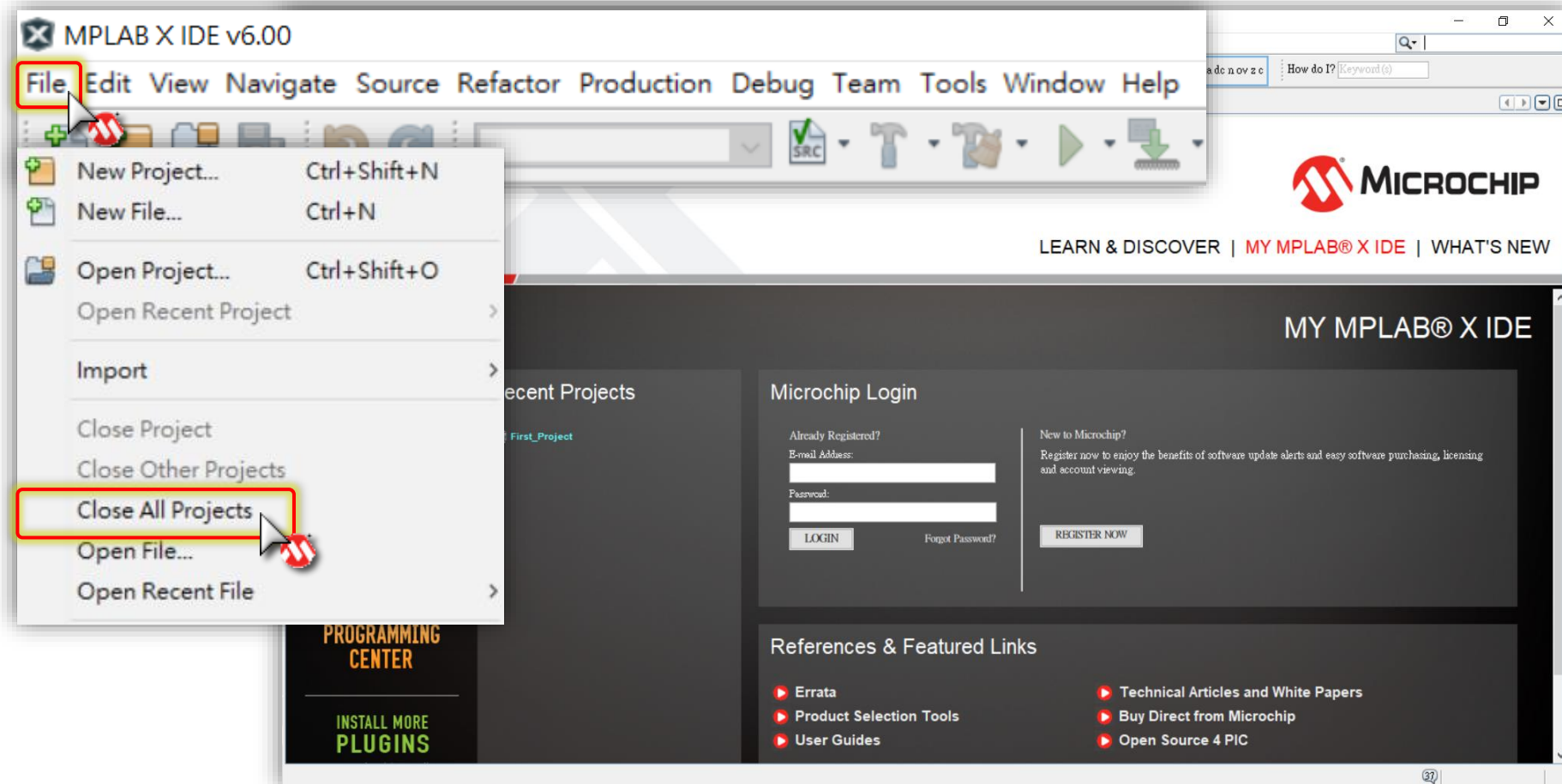
- Select : **Installed**
 - Search : "**MCC**"



Lab Preparation

Step 6

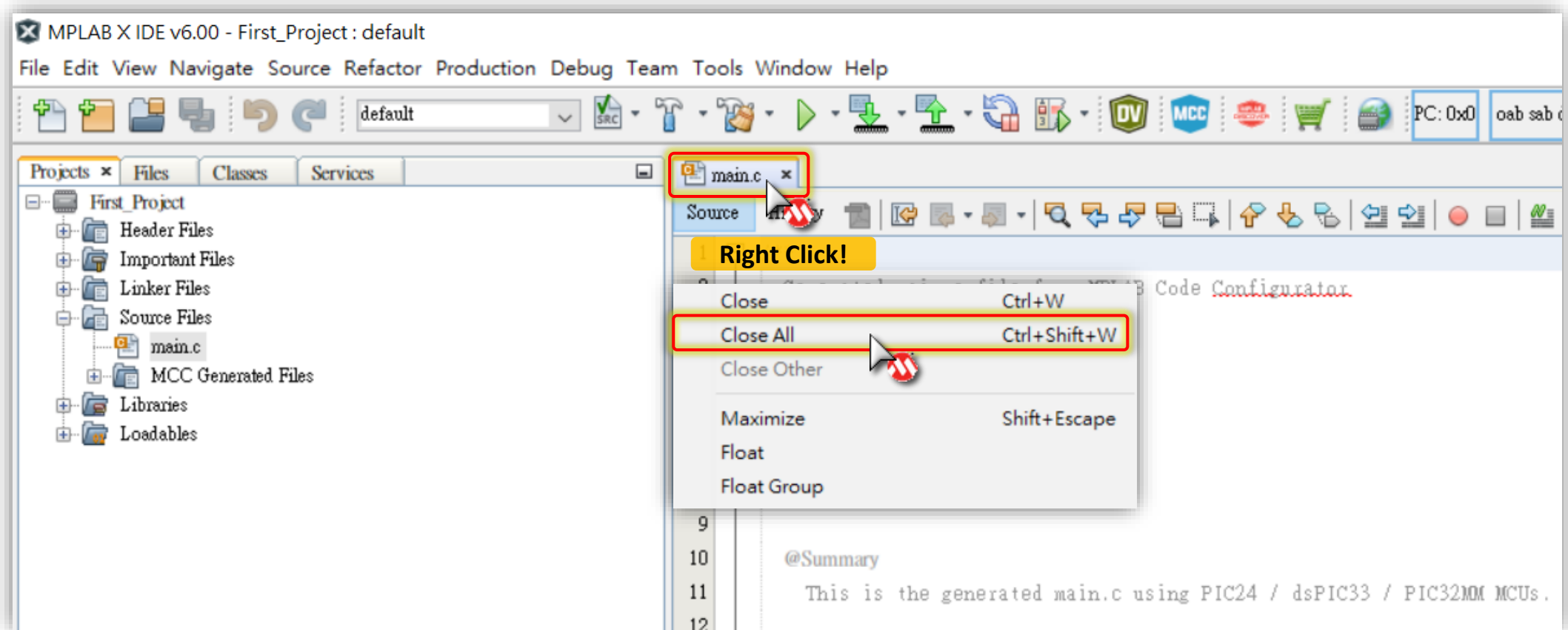
- To prevent opened projects to confuse our main project.
- Please select **File > Close All Projects**.



Lab Preparation

Step 7

- To prevent opened files to confuse our implementation.
- Please "**Right Click**" on opened file and choose "**Close All**"



Lab0 First Project

Lab0 First Project

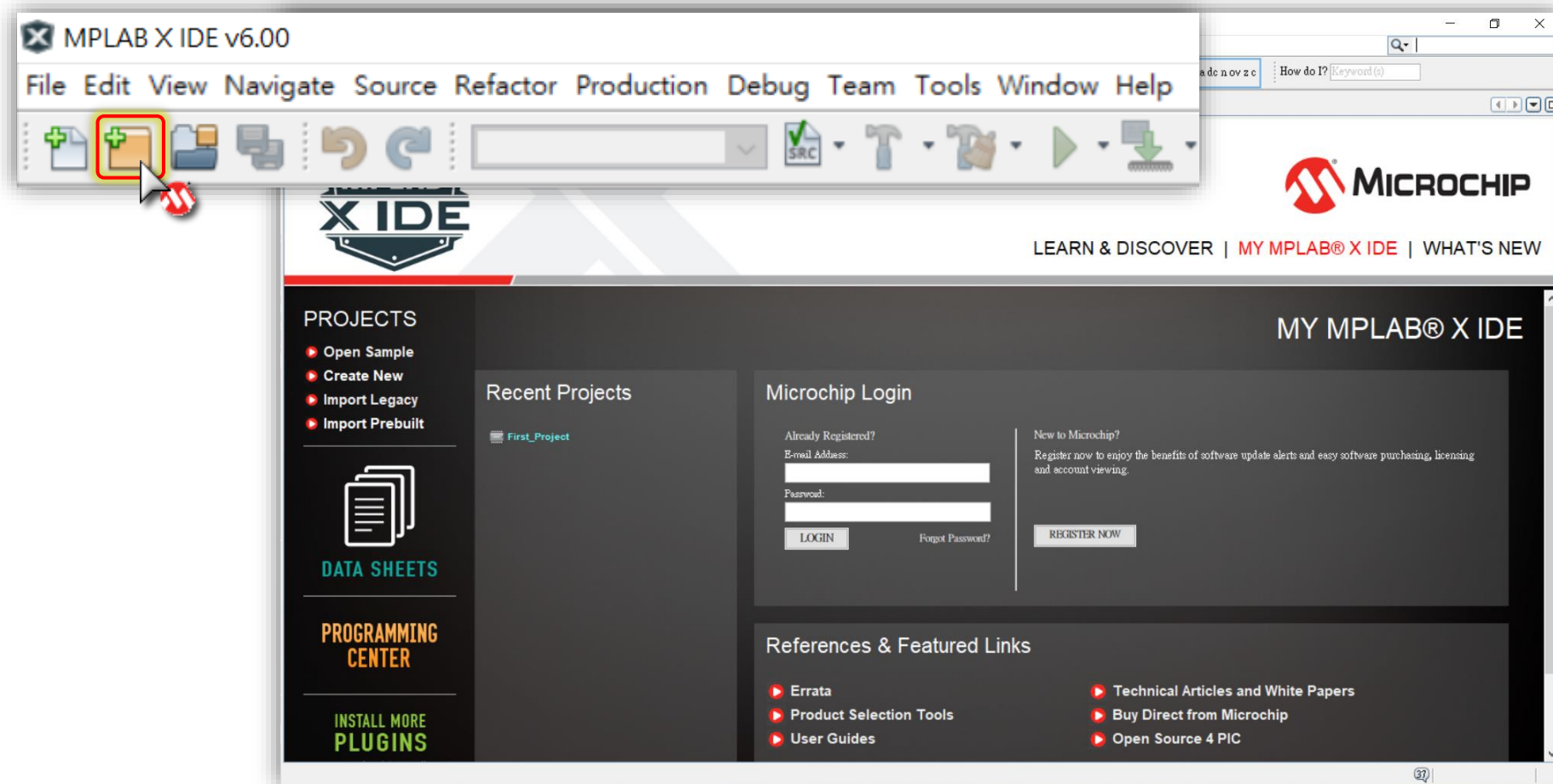
- Try to create your first MPLAB X IDE Project.
- Try to create your first MCC style Project.
- To understanding MPLAB X IDE basic operation.
- Learn how to use edit, build, project manager and program functions.

How to start ?

Lab0 First Project

Step 1

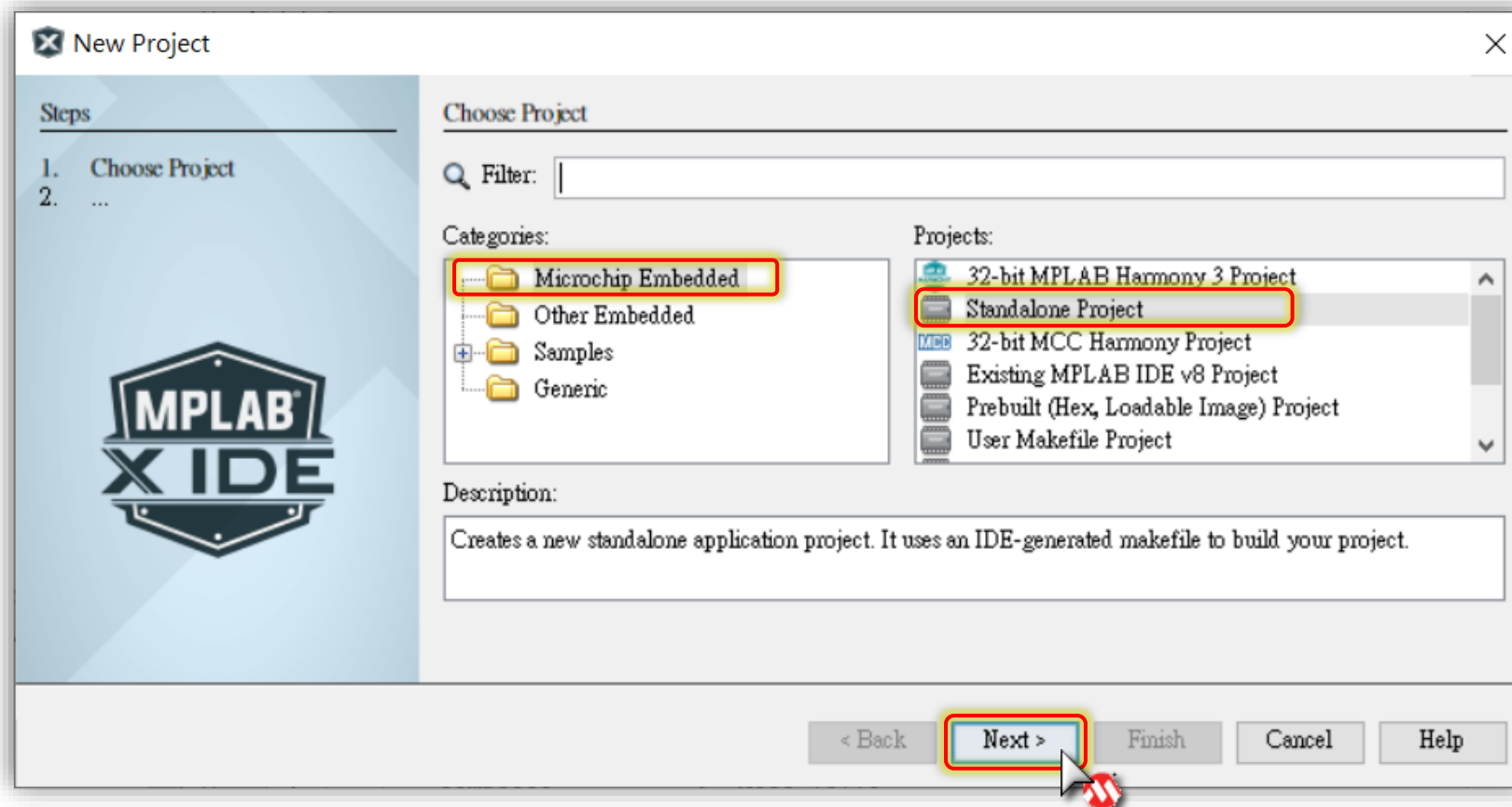
- Execute Project Wizard
- Select : **File > New Project** or Click icon



Lab0 First Project

Step 2

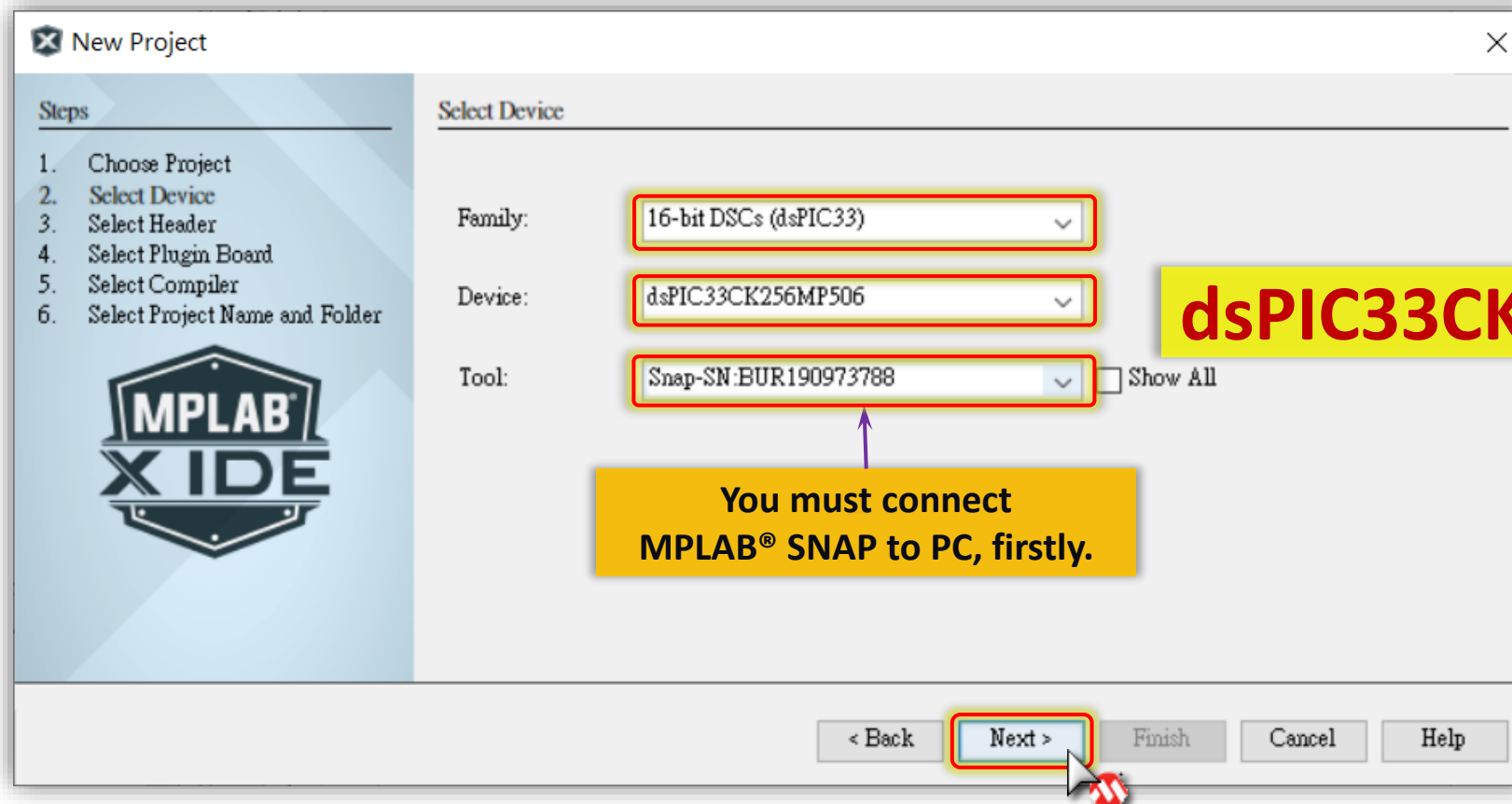
- Chooses Project :
 - Categories : **Microchip Embedded** , Projects : **Standalone Project**



Lab0 First Project

Step 3

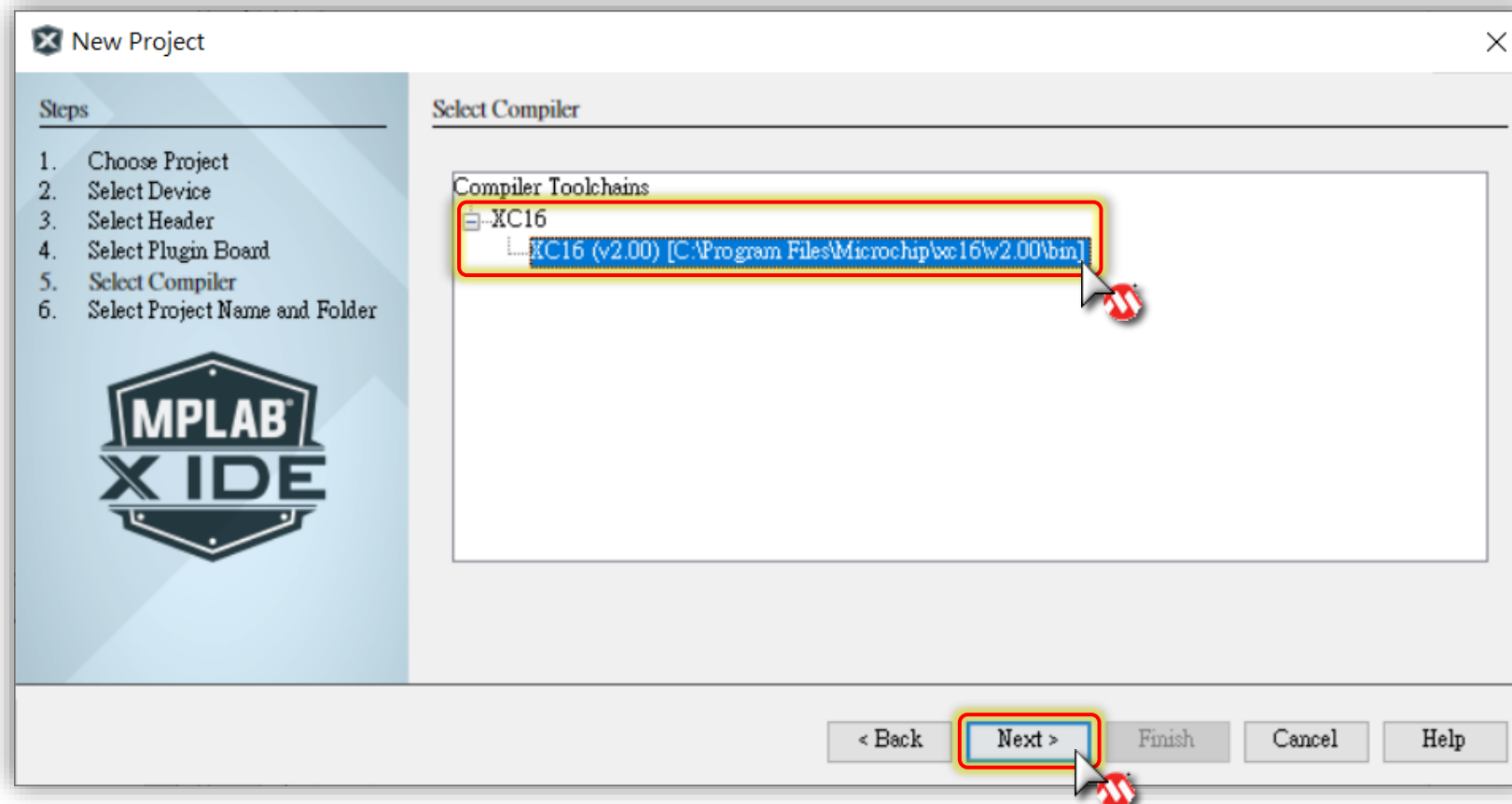
- Select Target Device :
 - Family: **16-bit DSCs (dsPIC33)** , Device: **dsPIC33CK256MP506** , Tool : **Snap-SN : BURXXXXXXXXXX**



Lab0 First Project

Step 4

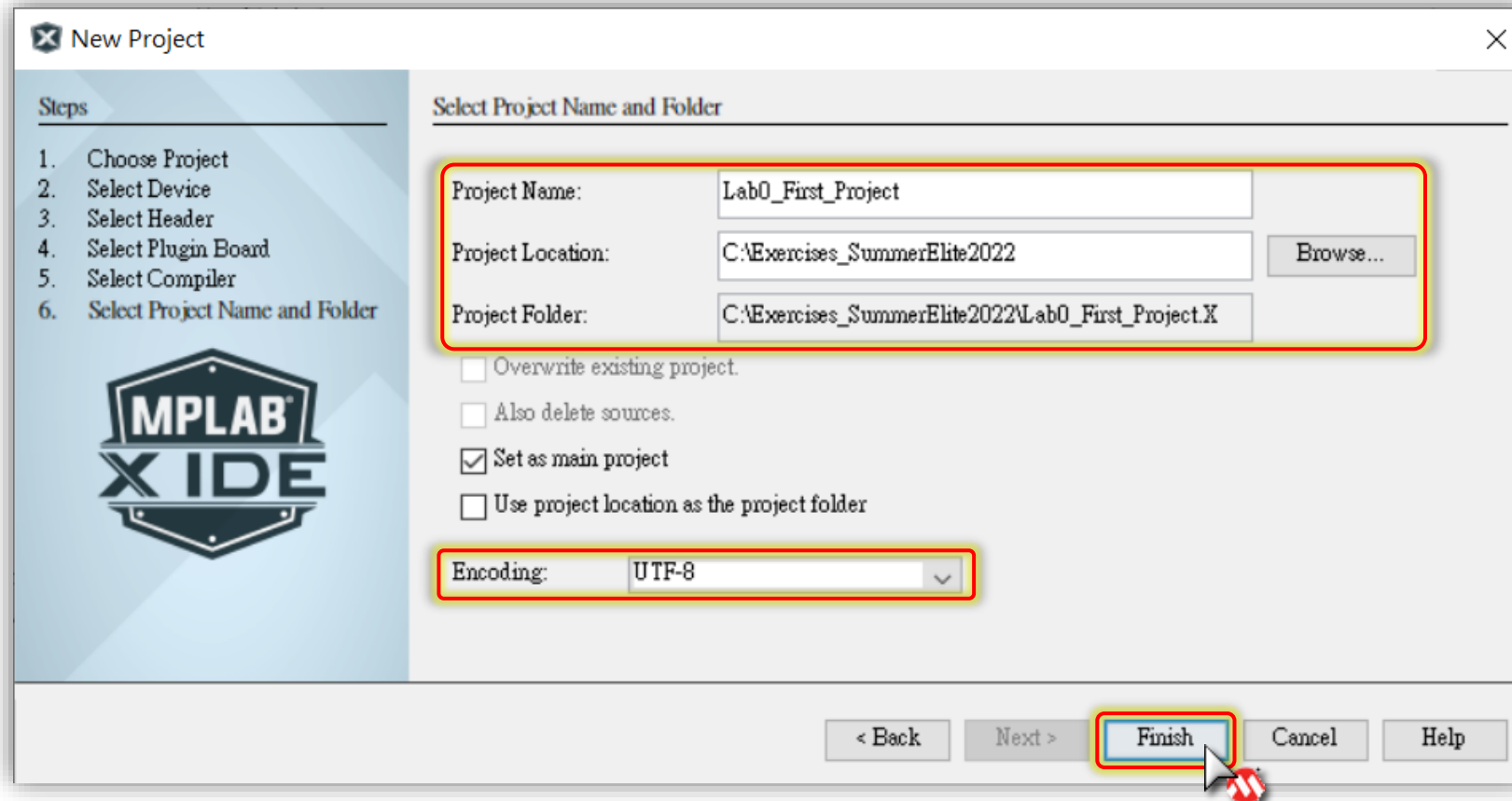
- **Select Compiler :**
 - **Compiler Toolchains : XC16 (v2.00) [C:\Program Files\Microchip\xc16\v2.00\bin]**



Lab0 First Project

Step 5

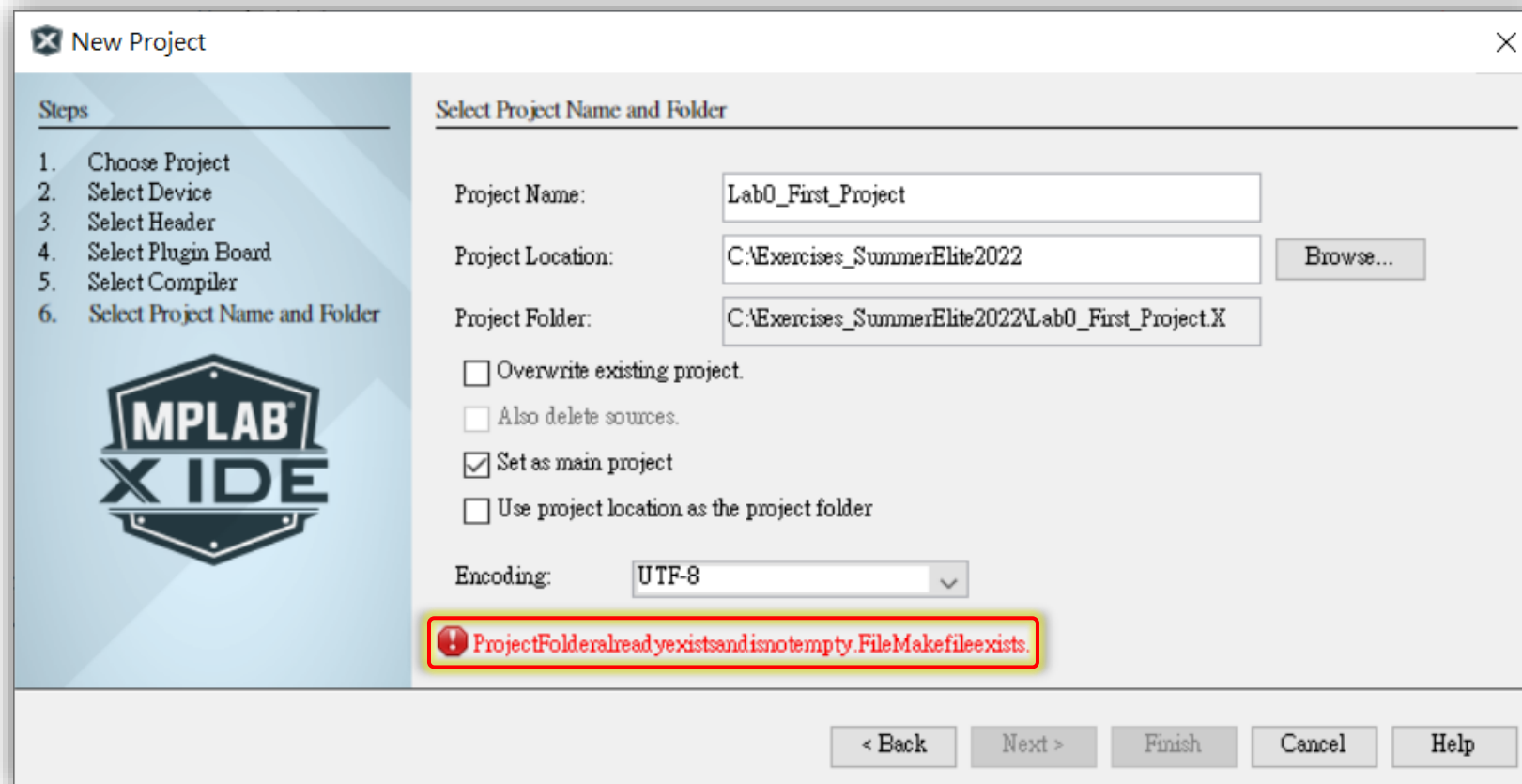
- Select Project Name and Folder :
 - Project Name : **Lab0_First_Project** , Project Location : **C:\Exercises_SummerElite2022**
 - Encoding : **UTF-8**



Lab0 First Project

Notice

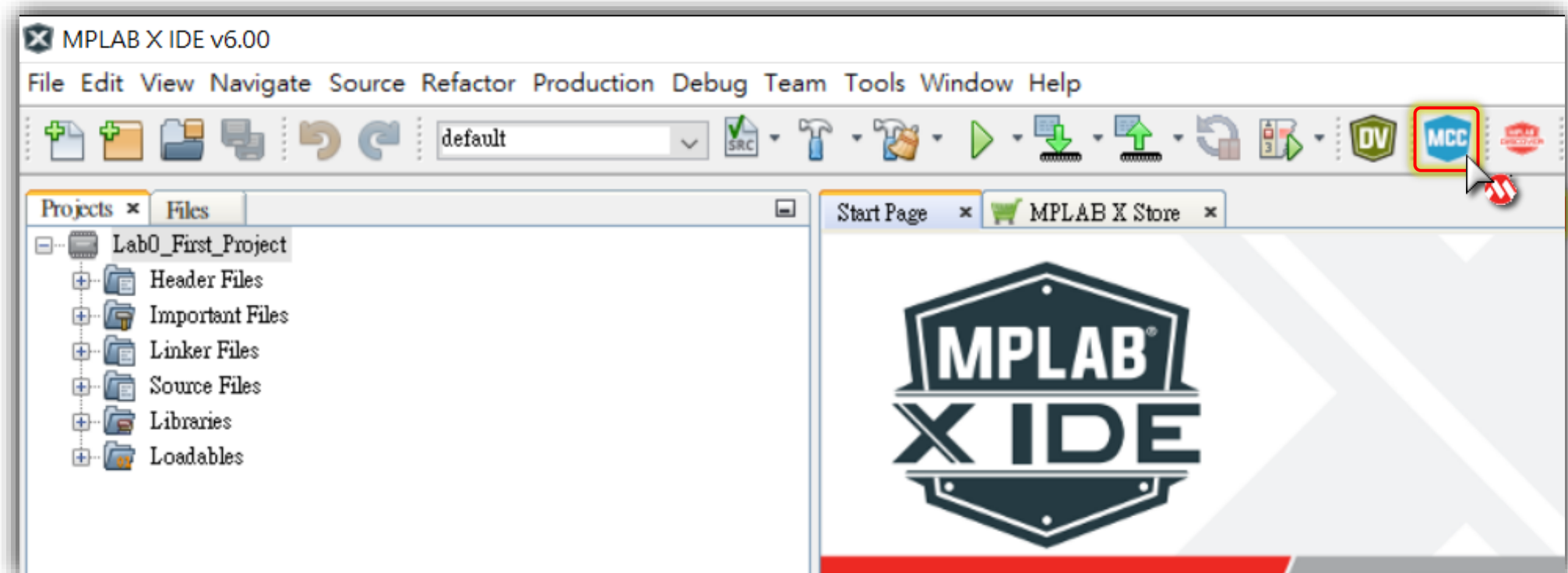
- If an old project already exist at the same location, you must delete it firstly or change to new project folder.



Lab0 First Project

Step 6

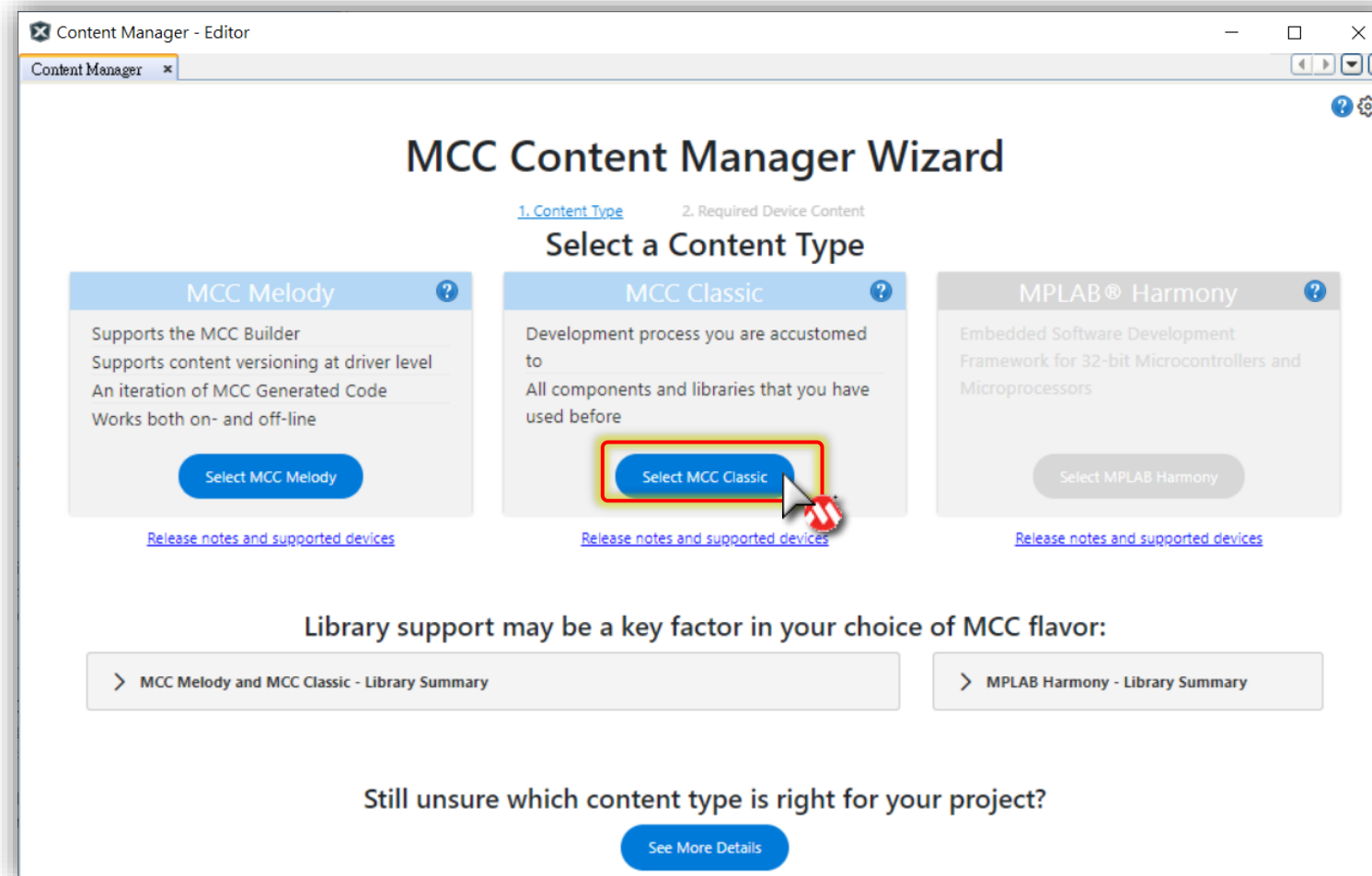
- Execute MCC
- Menu **Tools** ► **Embedded** ► **MPLAB Code Configurator v5: Open/Close** or Click 



Lab0 First Project

Step 7

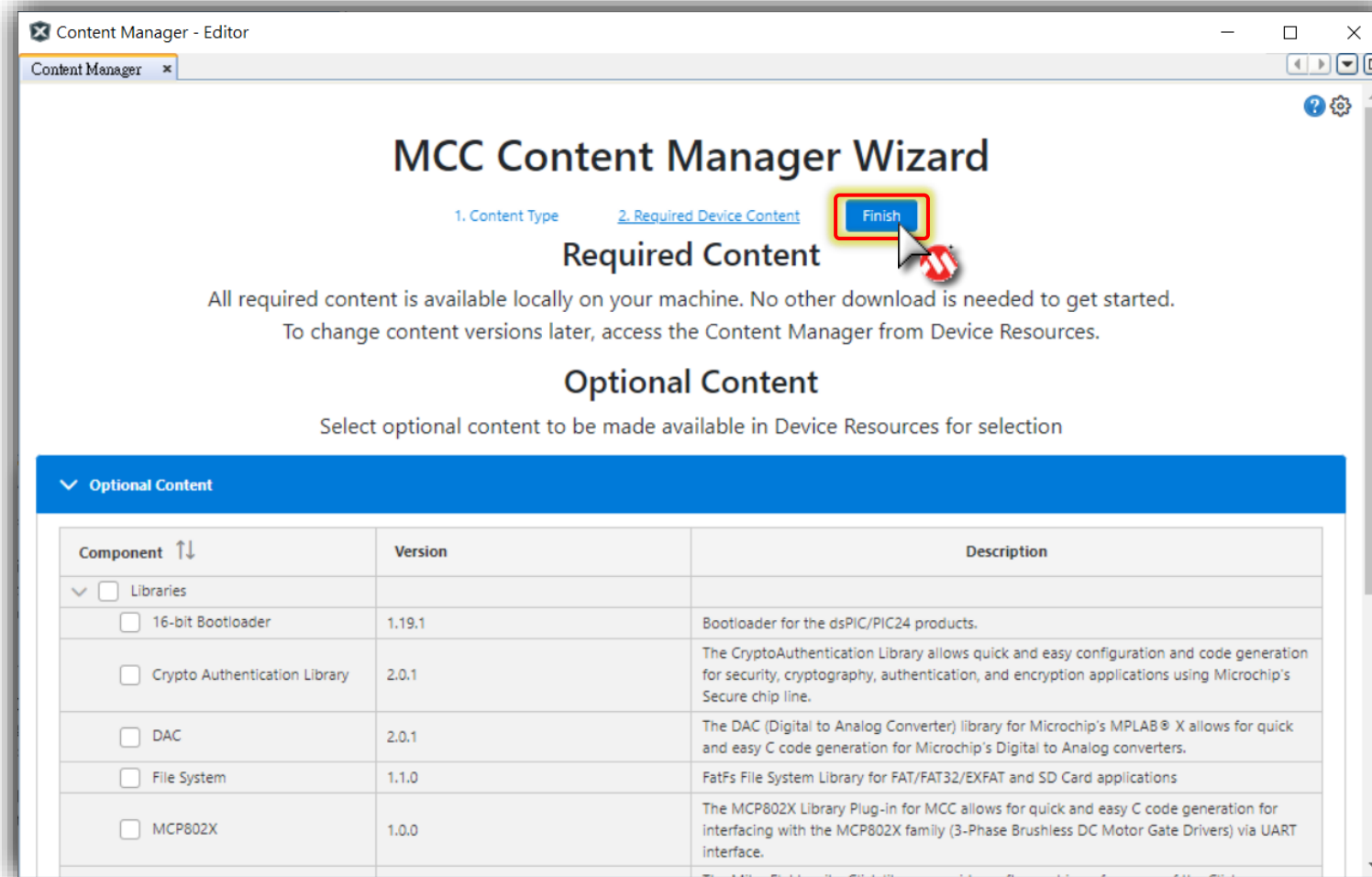
- "Select MCC Classic"



Lab0 First Project

Step 8

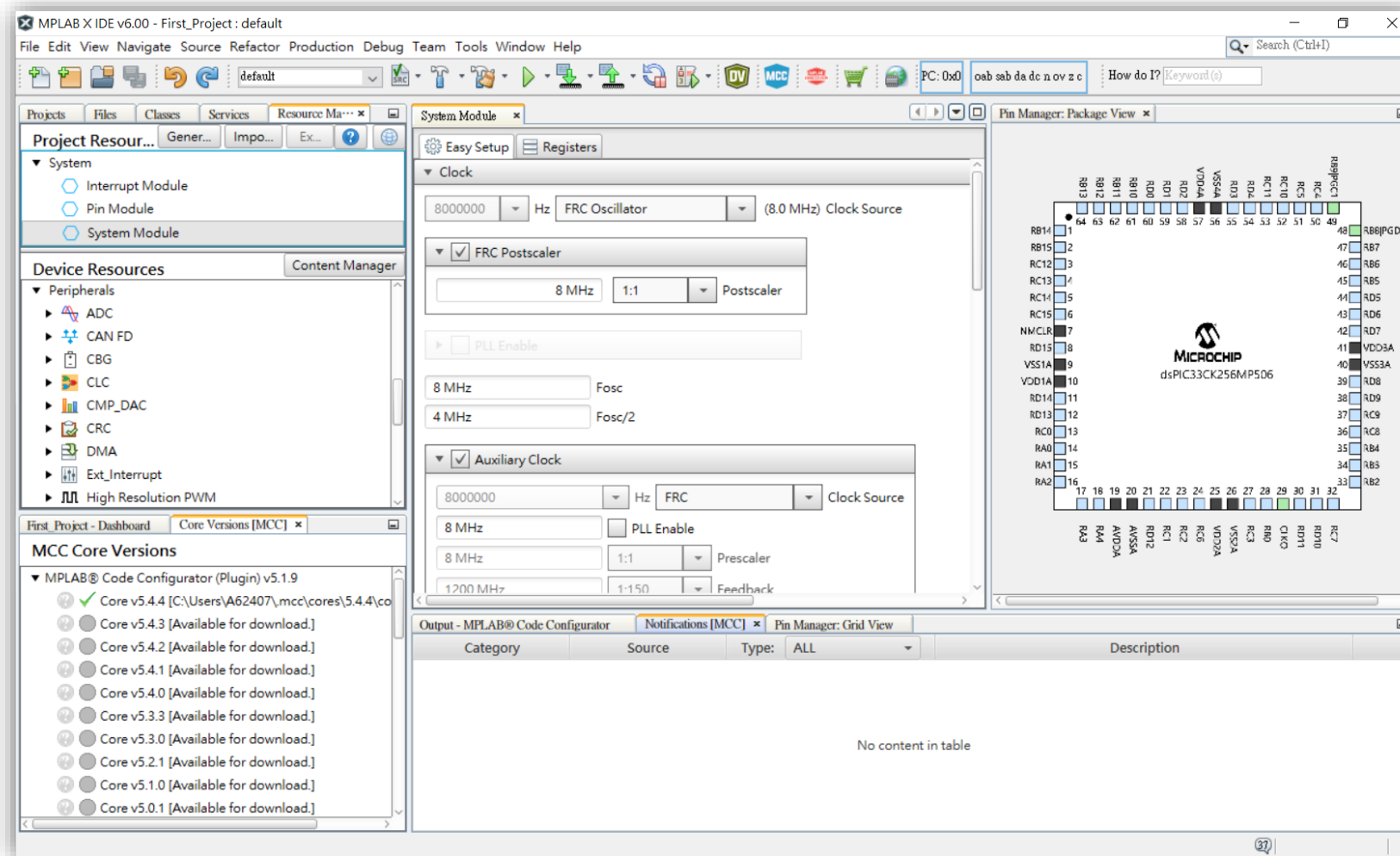
- You can add the available Libraries provided in the Device resource to the project or click **"Finish"**.



Lab0 First Project

Step 9

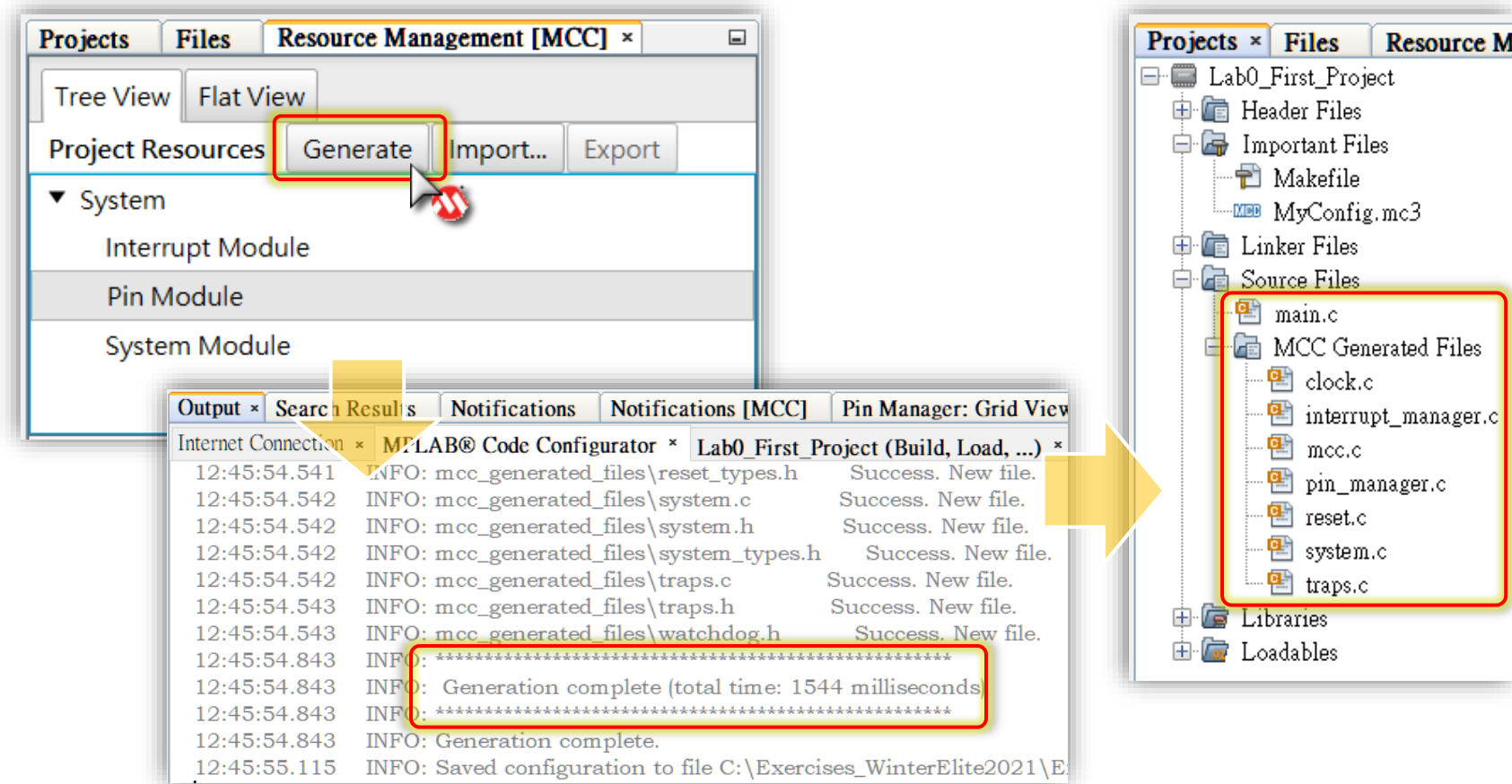
- MCC Configurator interface show up.



Lab0 First Project

Step 10

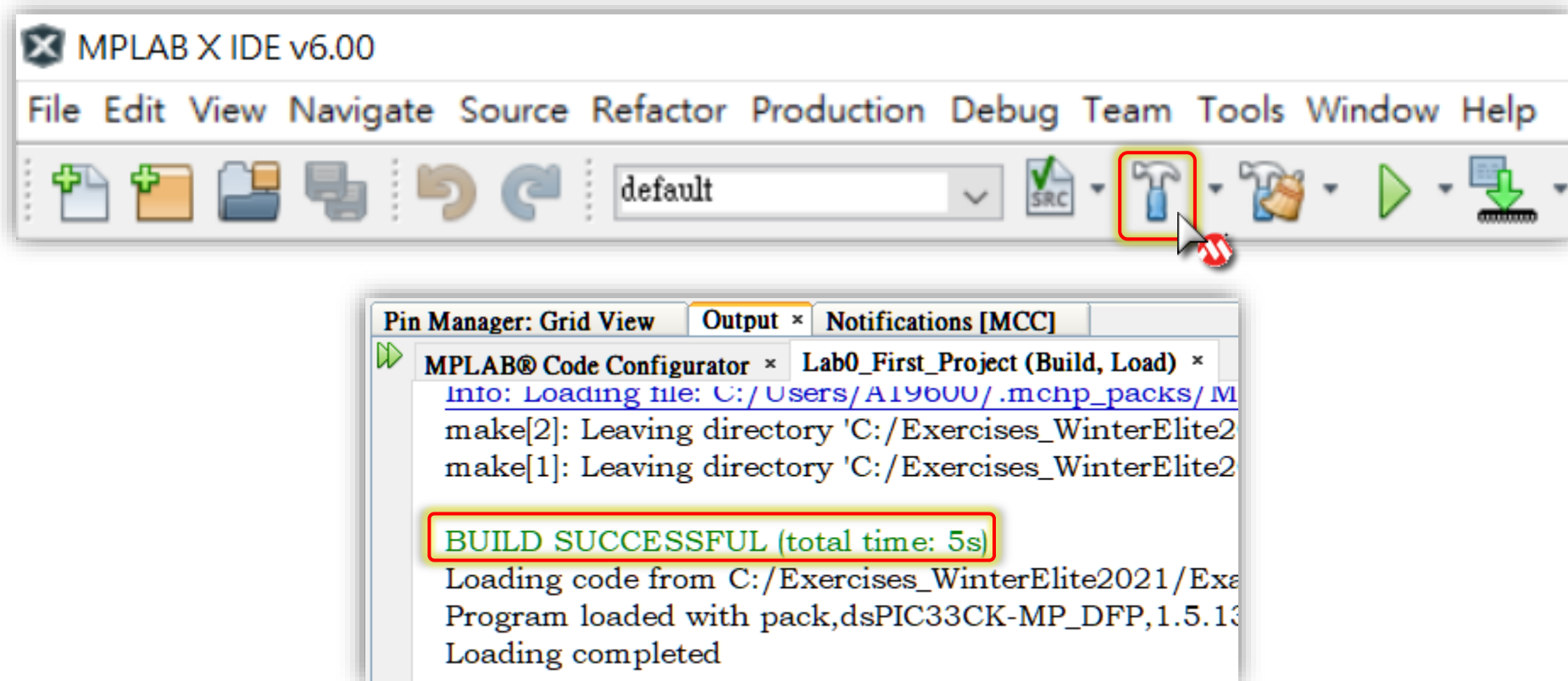
- Click "**Generate**" Button to get your first MCC Project.
- MCC to generate files include "**main.c**" automatically.



Lab0 First Project

Step 11

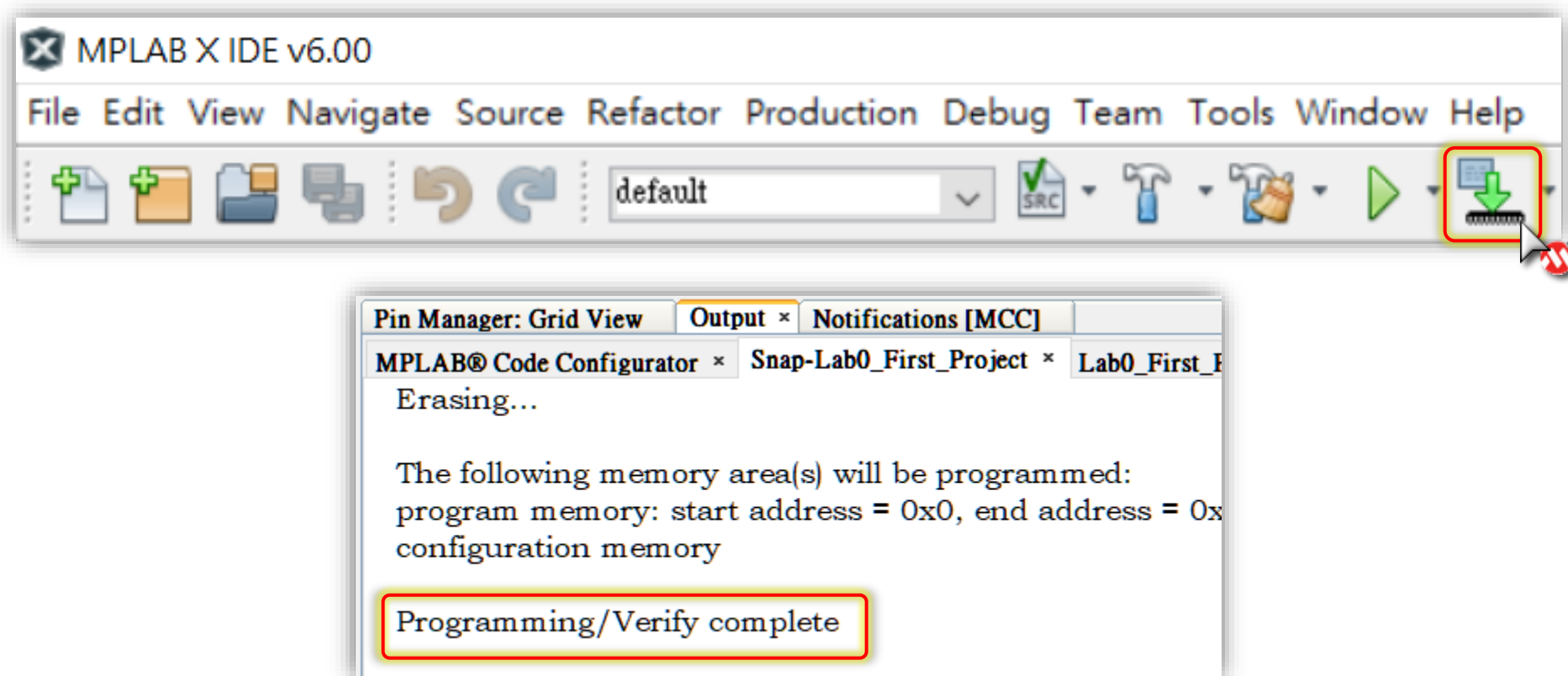
- Build your first MCC Project.
 - Select "**Build Main Project**" icon 
- Make sure compiled result is **BUILD SUCCESSFUL**



Lab0 First Project

Step 12

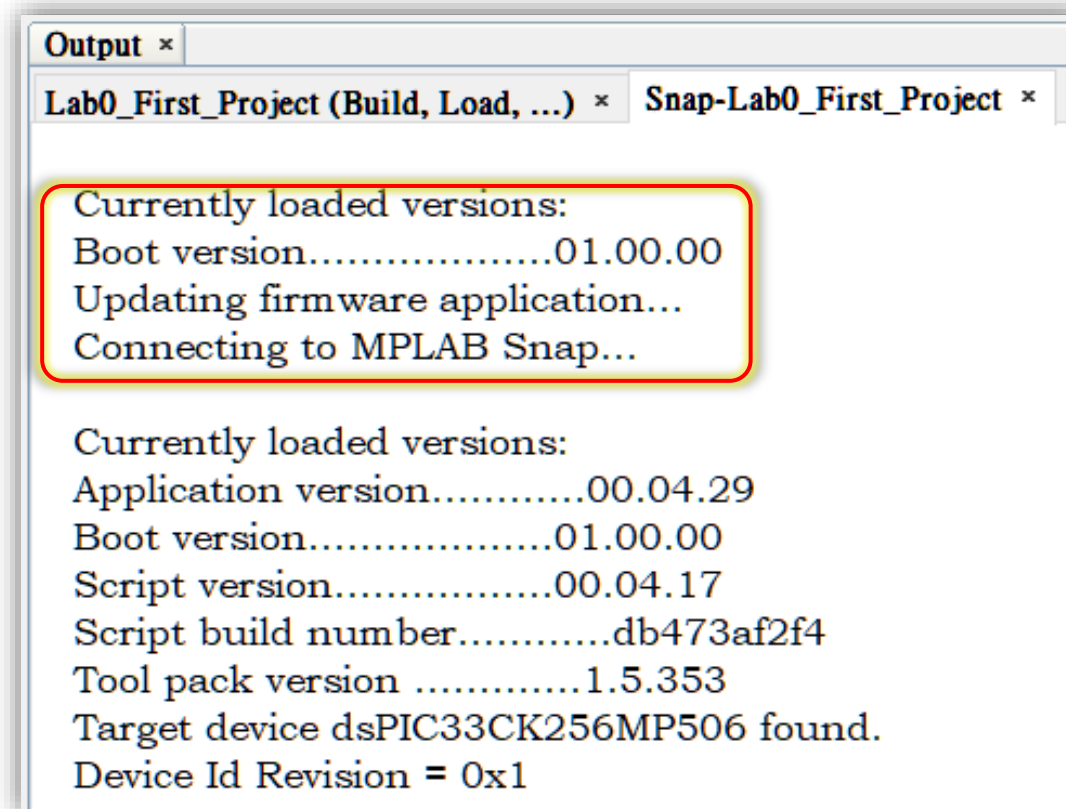
- Try program compiled firmware to your target board.
 - Select "**Make & Program Device Main Project**" icon 
Make sure the result is **Programming/Verify complete**



Lab0 First Project

Notice

- MPLAB X IDE will auto upgrade the firmware of SNAP to keep it up to date, it might take couple minutes to wait for finish. Your project will start programming after this.



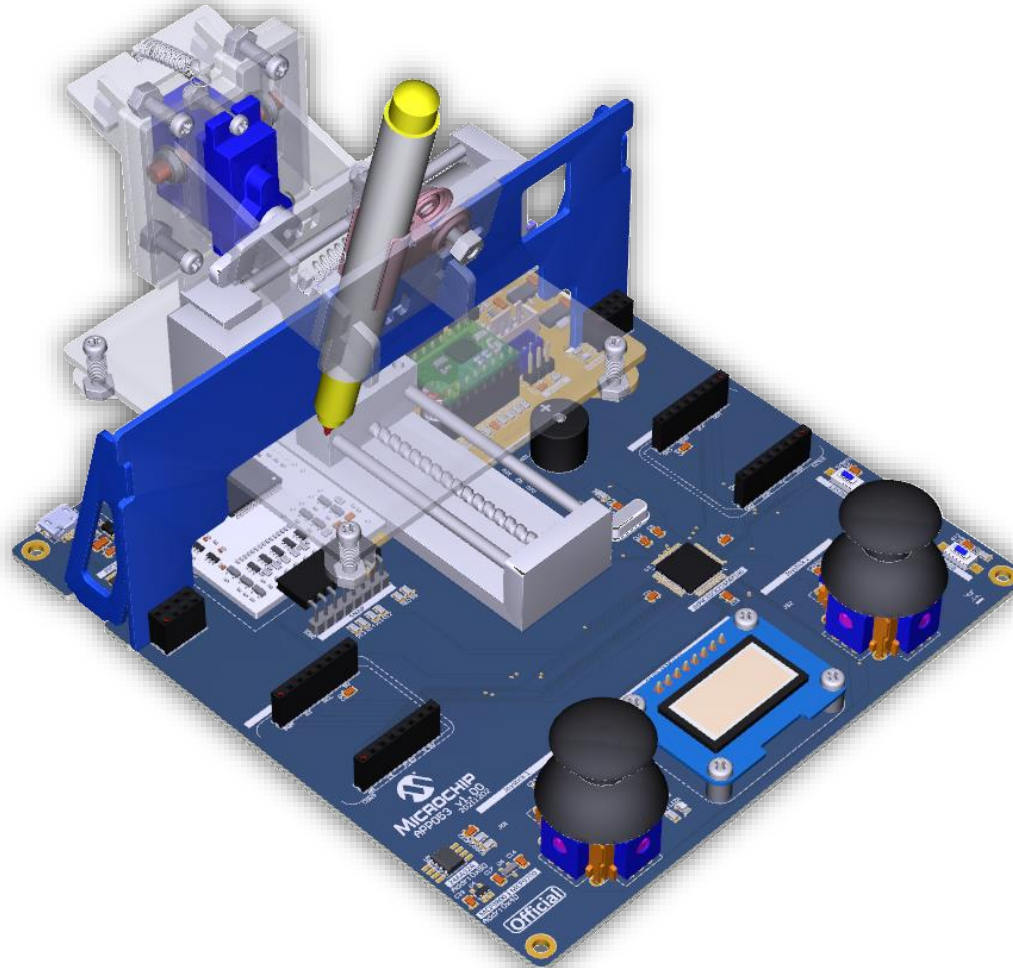
The screenshot shows the 'Output' window in MPLAB X IDE. It contains two tabs: 'Lab0_First_Project (Build, Load, ...)' and 'Snap-Lab0_First_Project'. The 'Snap-Lab0_First_Project' tab is active and displays the following text:

```
Currently loaded versions:  
Boot version.....01.00.00  
Updating firmware application...  
Connecting to MPLAB Snap...  
  
Currently loaded versions:  
Application version.....00.04.29  
Boot version.....01.00.00  
Script version.....00.04.17  
Script build number.....db473af2f4  
Tool pack version .....1.5.353  
Target device dsPIC33CK256MP506 found.  
Device Id Revision = 0x1
```


Lab0 First Project

Check Point

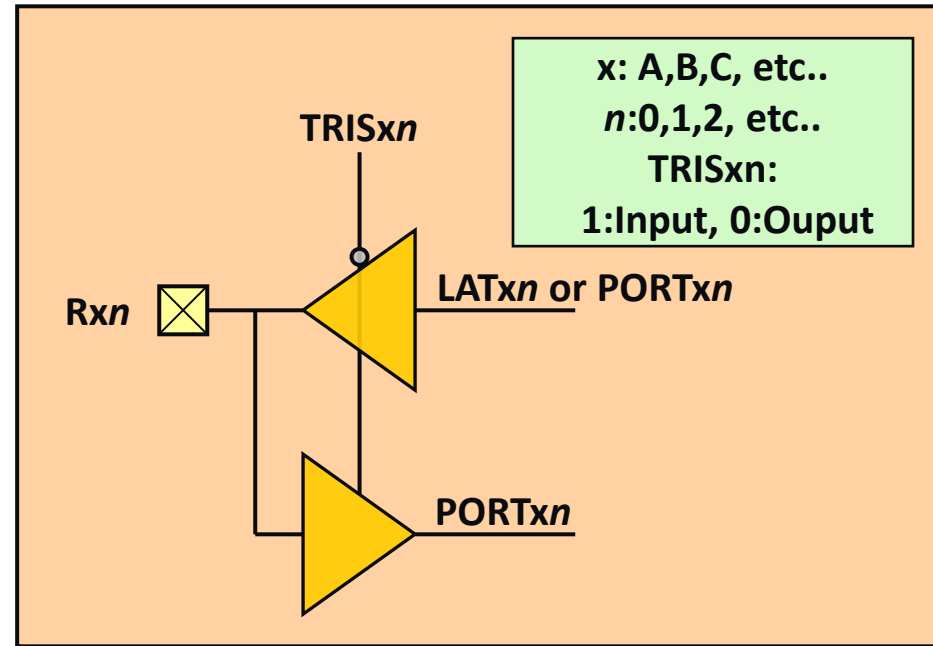
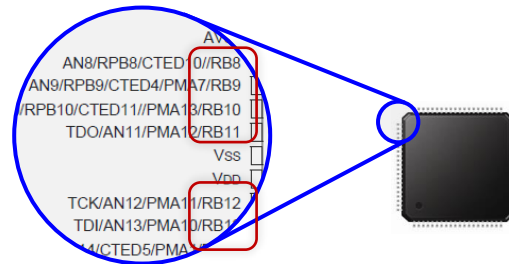
Nothing happen ????



GPIO Architecture

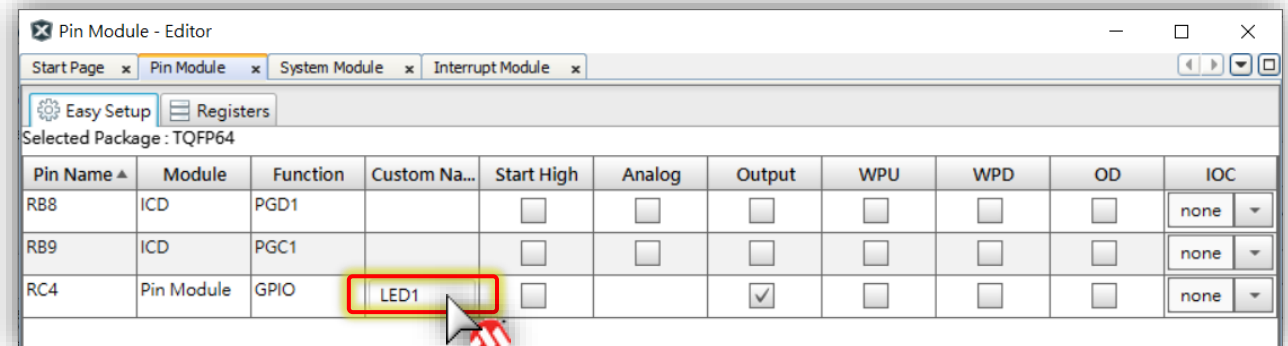
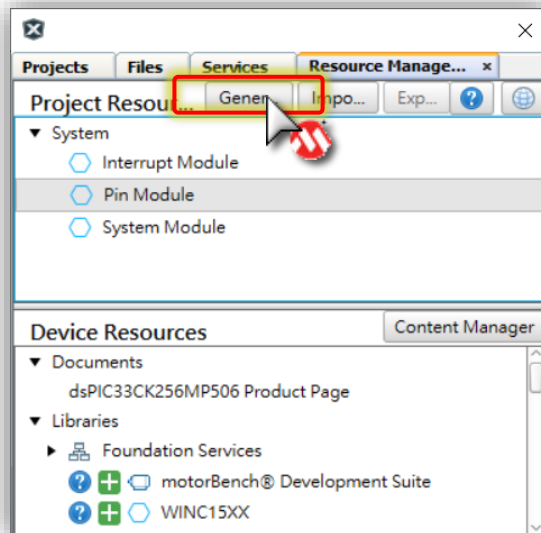
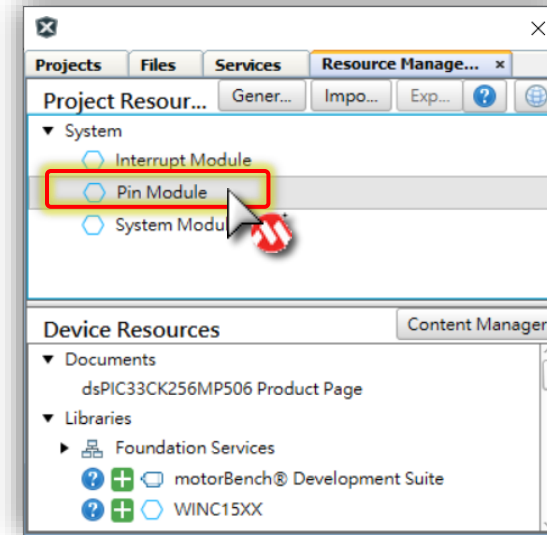
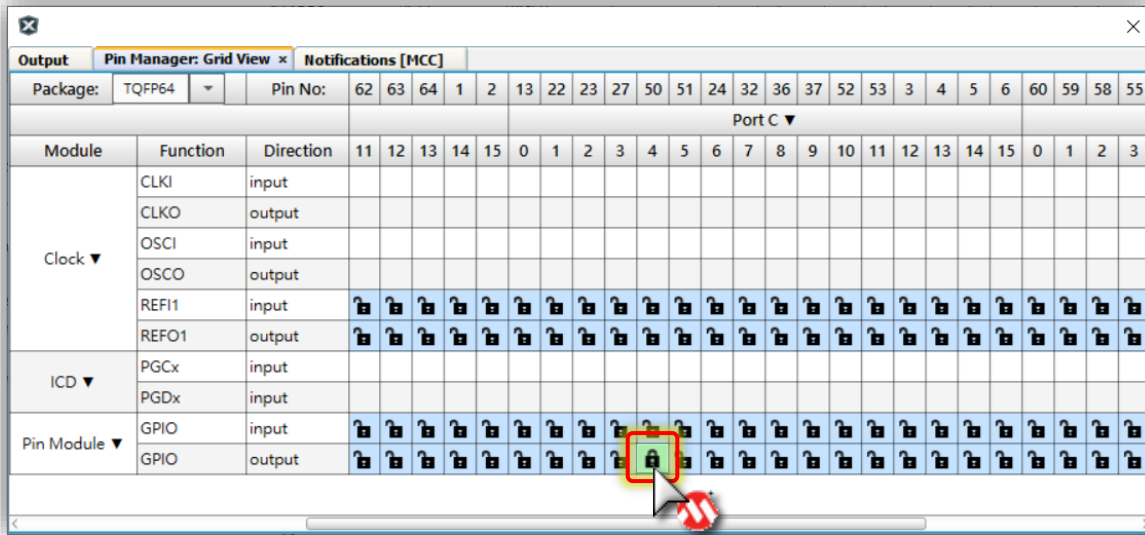
GPIO Block Diagram

- Please refer to the block diagram, this is the concept for programming model. The real and detail block diagram please refer to the following slide.
- **TRIS:** To determine Input or Output.
1 for Input, 0 for Output.
- **LAT:** set output drive value for Output pins.
- **PORT:** reaction logical level from Input pins.



GPIO Setting of MCC

- Just Click & Select. Code Generate Automatically.



Click & Rename

GPIO Functions of MCC

- MCC Provide below functions for GPIO:

Prototype definition : "pin_manager.h"

- OUT operation

*CustomName*_SetHigh();

*CustomName*_SetLow();

*CustomName*_Toggle();

*CustomName*_SetDigitalOutput();

- IN operation

*CustomName*_GetValue();

*CustomName*_SetDigitalInput();

Lab1 Multi-GPIO Output

Lab1 Multi-GPIO Output

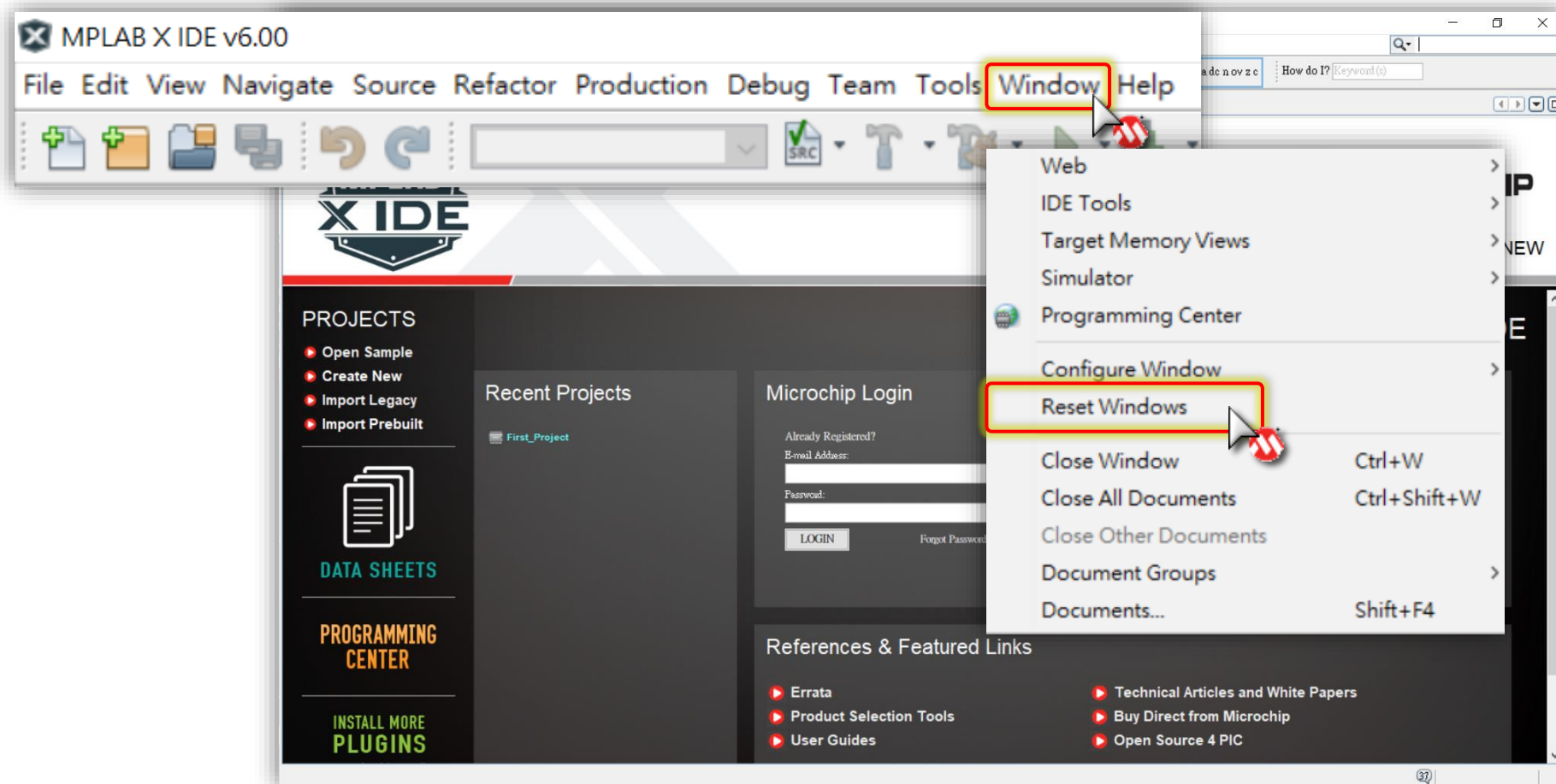
- Base on current project or Open `.\Lab1_xxx.X` to do exercises.
- Try to use MCC to generate your first MCC style code and control GPIO base on MCC style.
- Try to set the four GPIO pins to digital output mode and toggle levels (period around 500ms ~ 1S), individually.
- **RC04** : LED(LED1), Start High.
- **RC05** : LED(LED2).

How to start ?

MPLAB X IDE

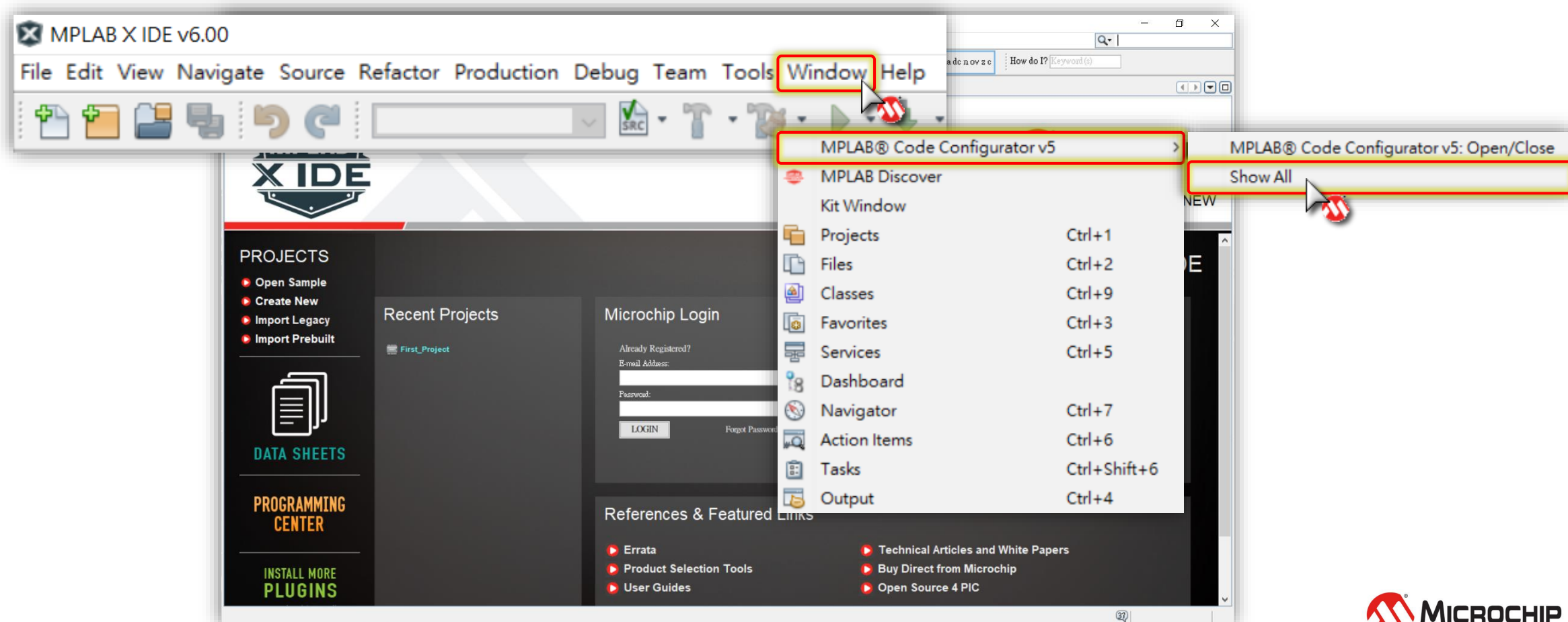
Tips

- Too many windows, You may can't find the window you want. MPLAB X IDE provide named "**Reset Windows**" to go back to default layout.



MCC Hints

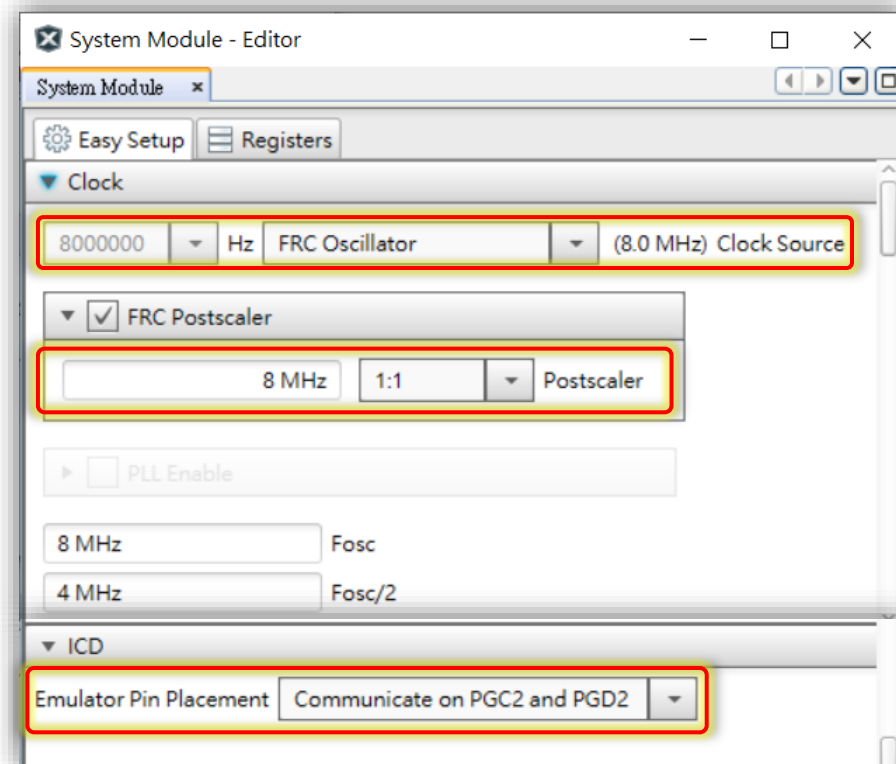
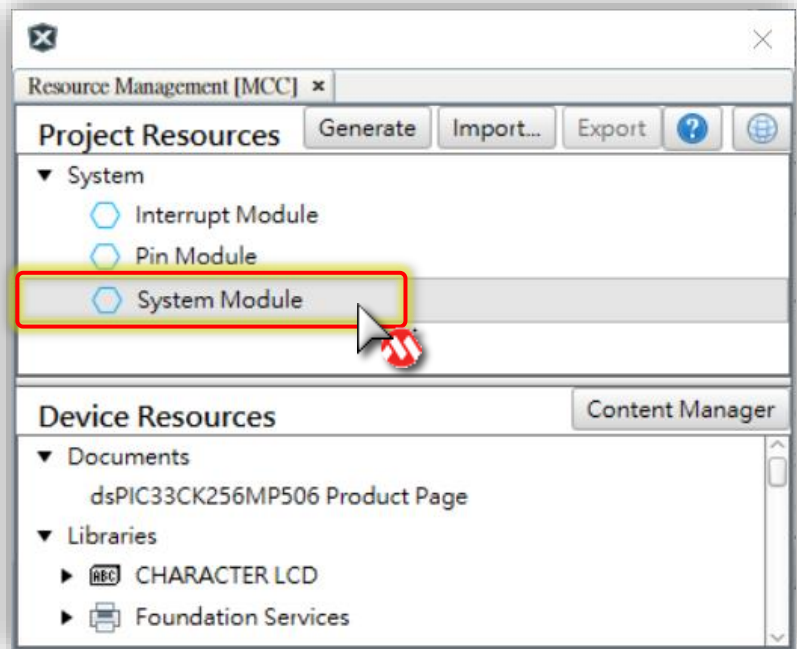
- You may can't find the MCC window you want also. MCC provide named "**Show All**" to show all MCC windows.



Lab1 Multi-GPIO Output

Step 1

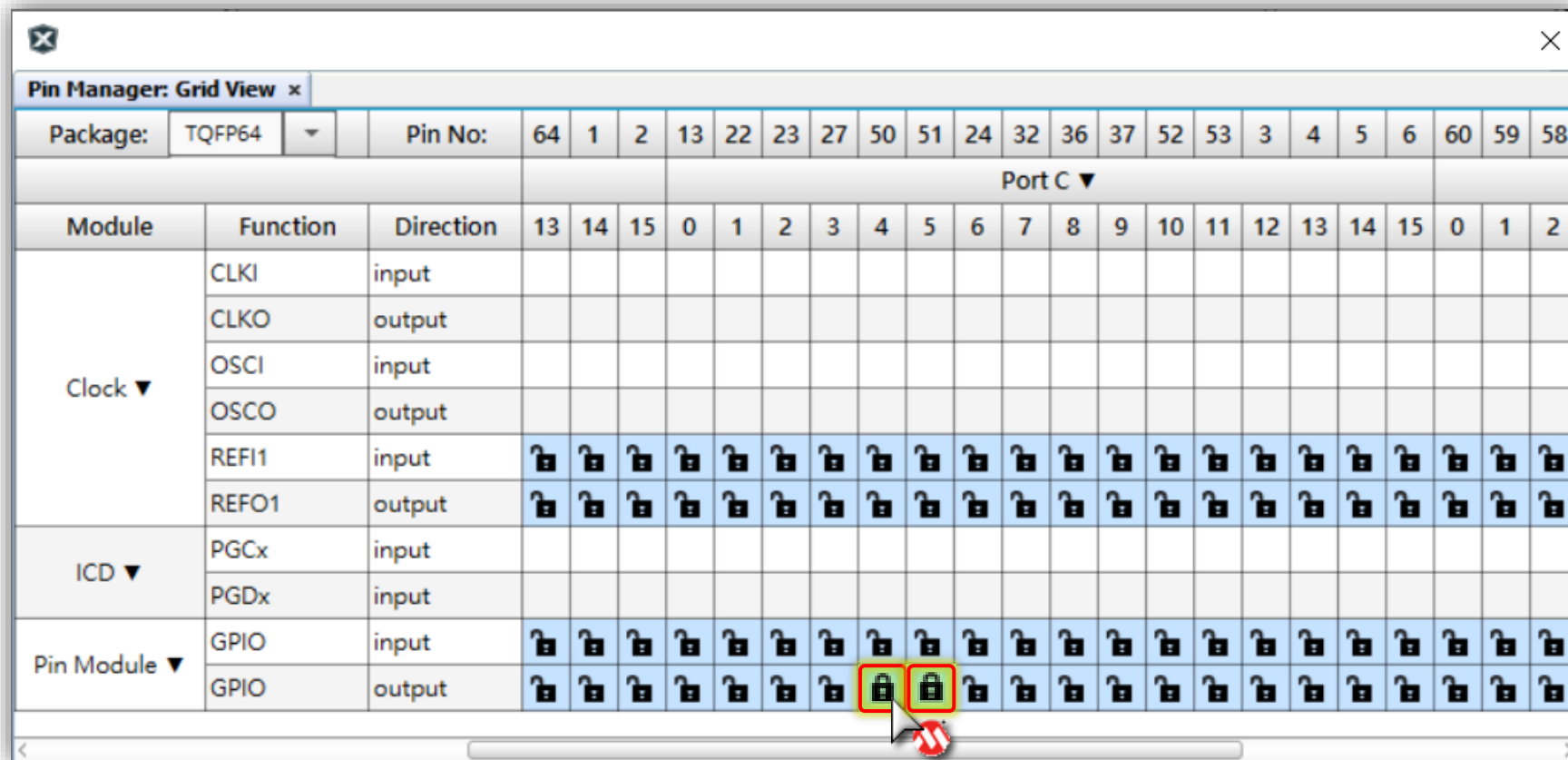
- Set System Clock & Debug Interface.
 - System Module : Clock : FRC Oscillator, Postscaler 1:1.
 - System Module : ICD : PGC2 and PGD2.



Lab1 Multi-GPIO Output

Step 2

- Set **PORTC** : Pin4 (RC4), Pin5 (RC5) to digital output mode.
 - Pin Manager Grid View : Click RC4, RC5 to lock to GPIO output.



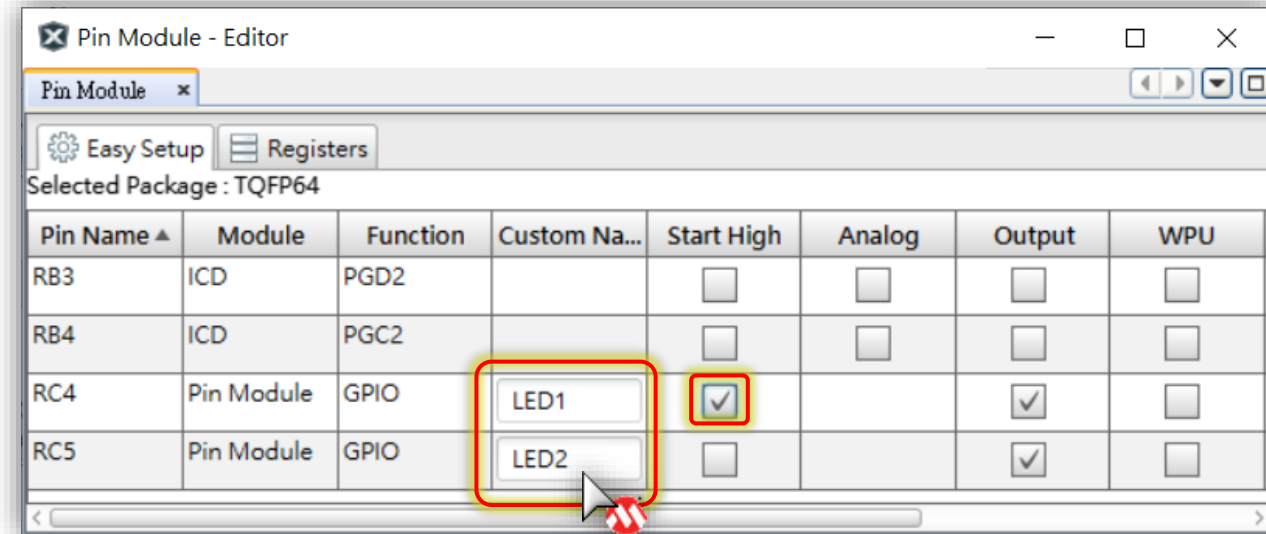
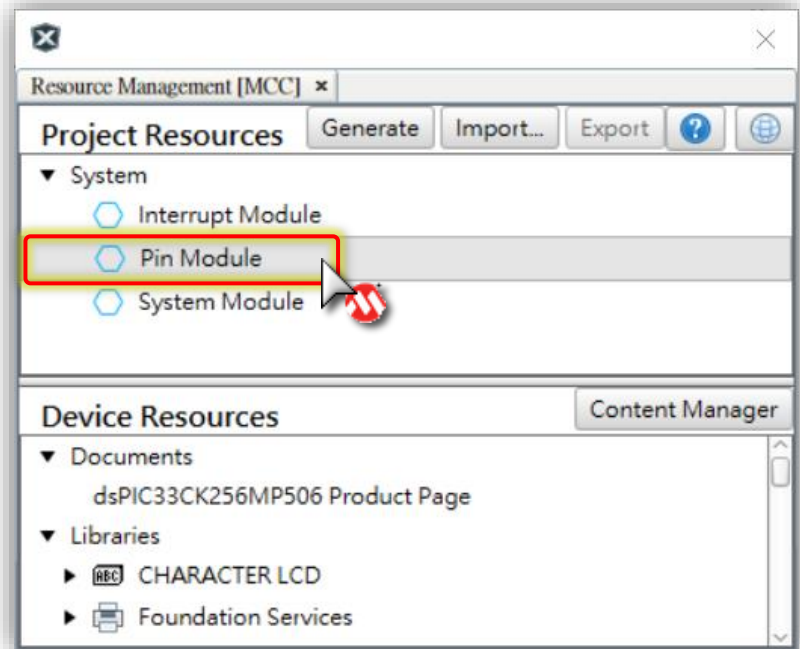
The screenshot shows the 'Pin Manager: Grid View' window. The 'Package' is set to 'TQFP64'. The 'Pin No.' list includes 64, 1, 2, 13, 22, 23, 27, 50, 51, 24, 32, 36, 37, 52, 53, 3, 4, 5, 6, 60, 59, 58. The 'Port C' dropdown is selected. The table below shows the configuration for various modules and functions across the pins.

Module	Function	Direction	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
Clock ▼	CLKI	input																						
	CLKO	output																						
	OSCI	input																						
	OSCO	output																						
	REFI1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	REFO1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ICD ▼	PGCx	input																						
	PGDx	input																						
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒

Lab1 Multi-GPIO Output

Step 3

- Set Alias for RC4, RC5.
 - Pin Module : RC4 Custom Name > LED1, Start high, Check Output.
 - Pin Module : RC5 Custom Name > LED2, Check Output.

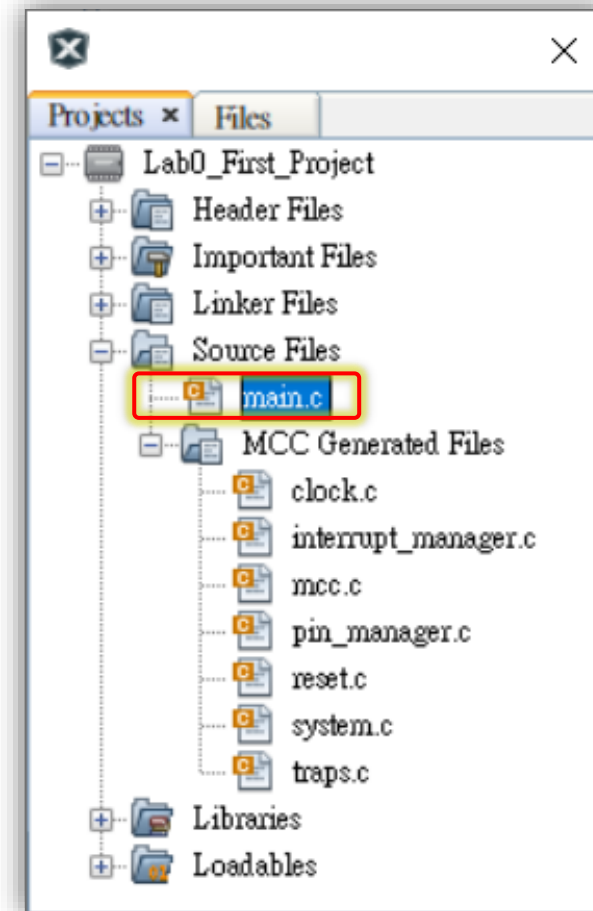
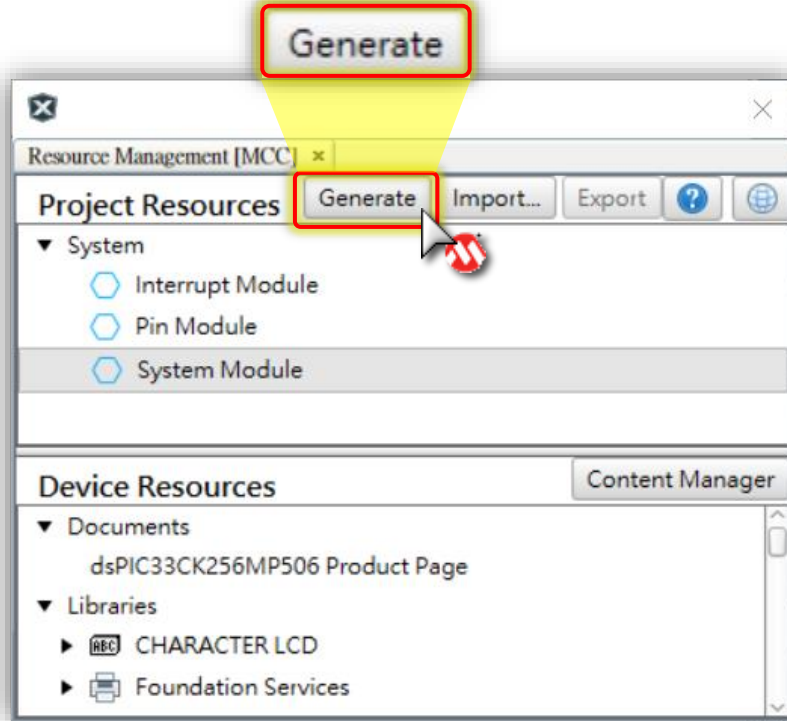


Click & Rename

Lab1 Multi-GPIO Output

Step 4

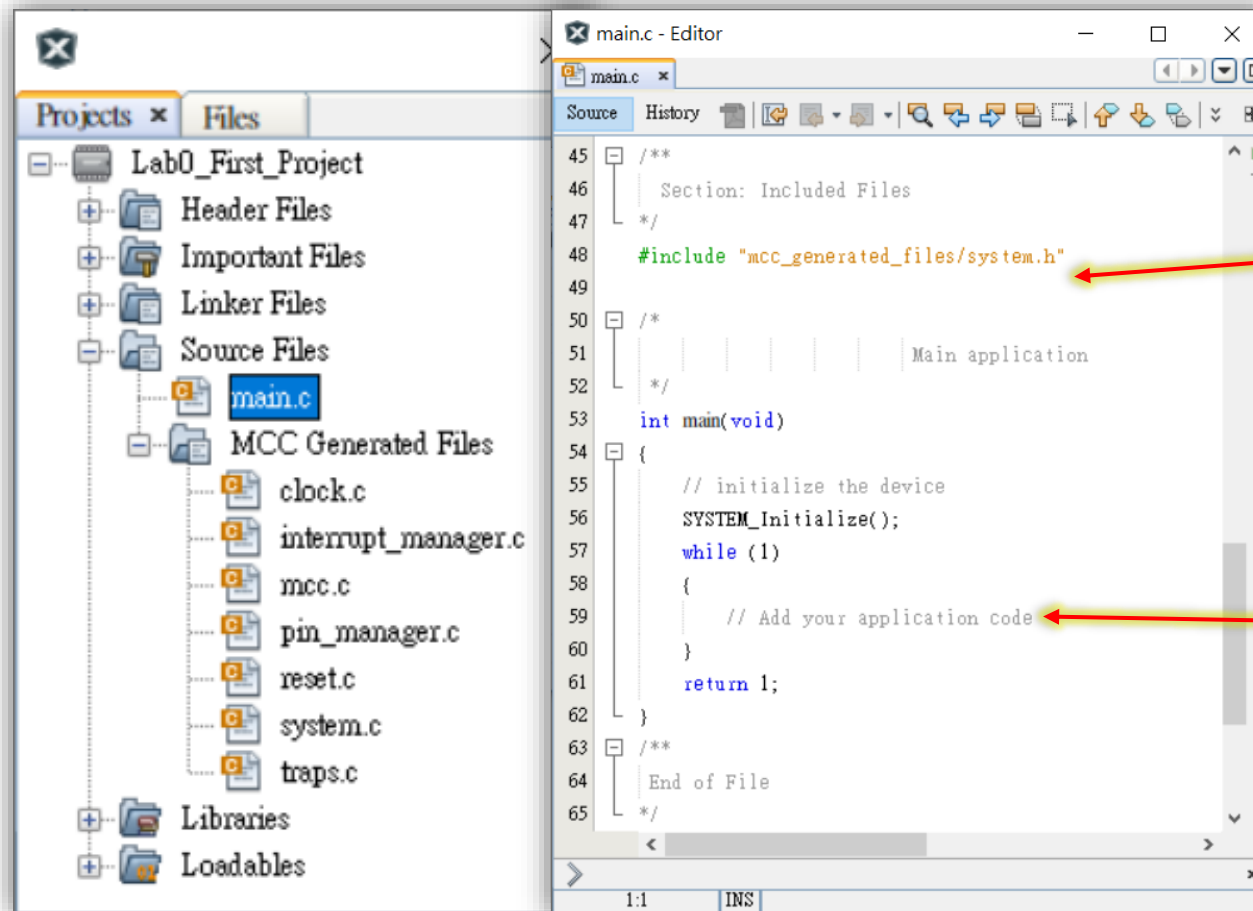
- Click "**Generate**" Button to generate your code.
- All files generate by MCC include "**main.c**" automatically.



Lab1 Multi-GPIO Output

Step 5

- Try to add code to main function double click "**main.c**" to view & add below code segment in it.



Add below code to main()

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"

uint32_t i = 0;

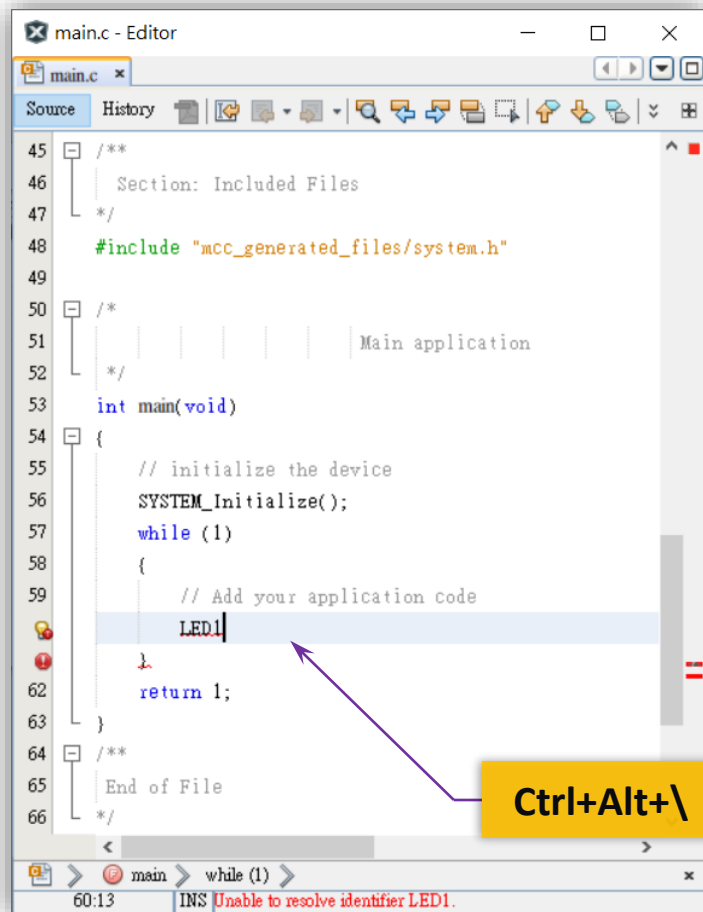
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    while (1)
    {
        // Add your application code
        LED1_Toggle();
        LED2_Toggle();
        for (i = 0; i < 80000; i++);
    }

    return 1;
}
```

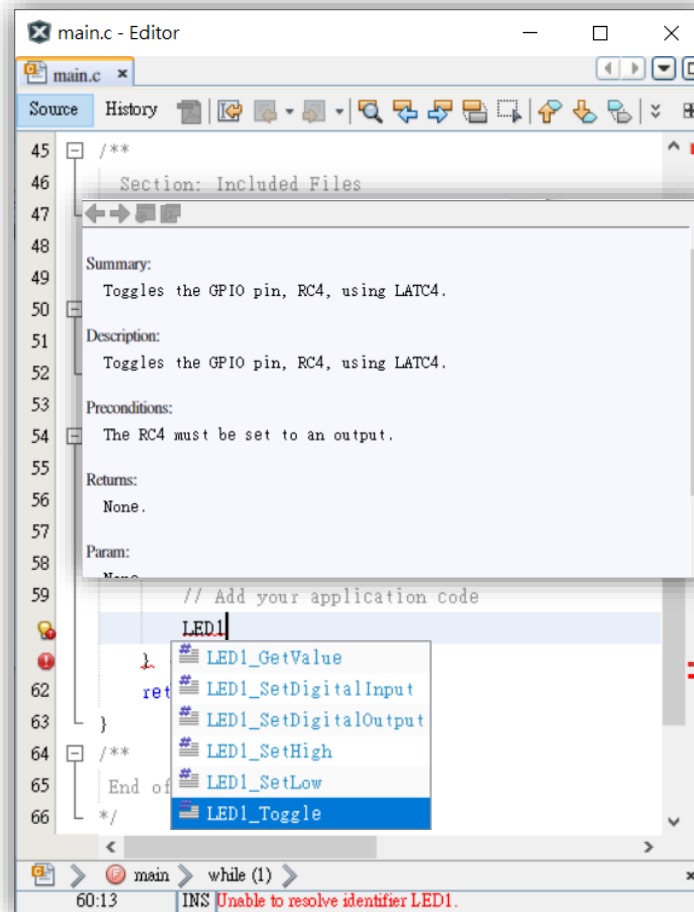
MPLAB X IDE Hints

- Hot key "**Ctrl + Alt + **"
 - Type LED1 first, then use "**Ctrl + Alt + **" to find you want.



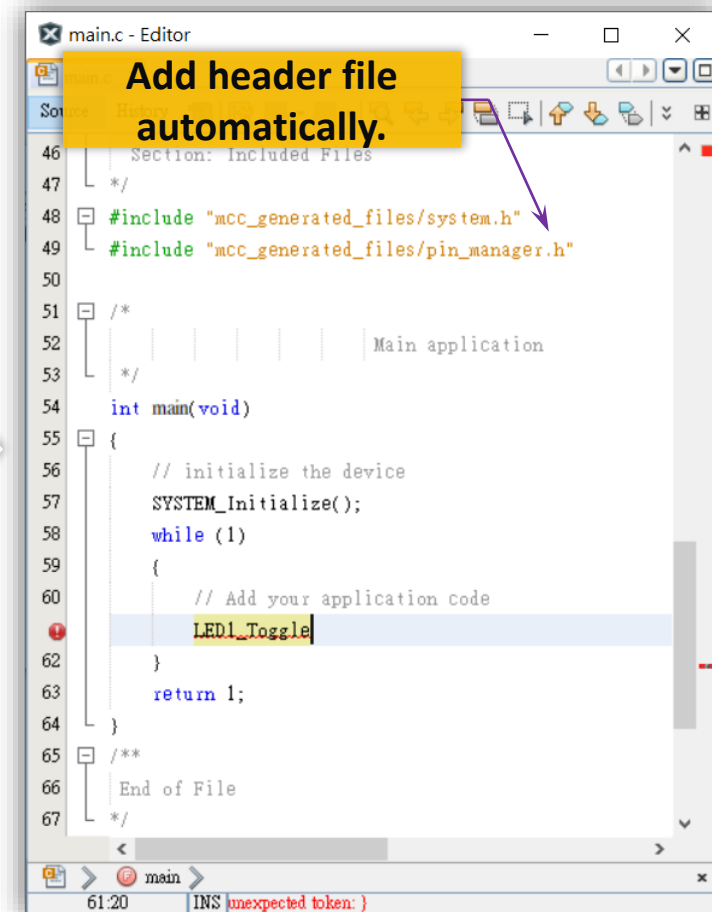
```
45  /**
46  |  Section: Included Files
47  |  */
48  |  #include "mcc_generated_files/system.h"
49  |
50  |  /**
51  |  |  Main application
52  |  |  */
53  |  int main(void)
54  |  {
55  |  |  // initialize the device
56  |  |  SYSTEM_Initialize();
57  |  |  while (1)
58  |  |  {
59  |  |  |  // Add your application code
60  |  |  |  LED1
61  |  |  |
62  |  |  |  return 1;
63  |  |  }
64  |  |  /**
65  |  |  End of File
66  |  |  */
```

60:13 | INS | Unable to resolve identifier LED1.



```
45  /**
46  |  Section: Included Files
47  |  */
48  |  #include "mcc_generated_files/system.h"
49  |
50  |  /**
51  |  |  Main application
52  |  |  */
53  |  int main(void)
54  |  {
55  |  |  // initialize the device
56  |  |  SYSTEM_Initialize();
57  |  |  while (1)
58  |  |  {
59  |  |  |  // Add your application code
60  |  |  |  LED1
61  |  |  |
62  |  |  |  return 1;
63  |  |  }
64  |  |  /**
65  |  |  End of File
66  |  |  */
```

60:13 | INS | Unable to resolve identifier LED1.



```
46  |  Section: Included Files
47  |  */
48  |  #include "mcc_generated_files/system.h"
49  |  #include "mcc_generated_files/pin_manager.h"
50  |
51  |  /**
52  |  |  Main application
53  |  |  */
54  |  int main(void)
55  |  {
56  |  |  // initialize the device
57  |  |  SYSTEM_Initialize();
58  |  |  while (1)
59  |  |  {
60  |  |  |  // Add your application code
61  |  |  |  LED1_Toggle
62  |  |  |
63  |  |  |  return 1;
64  |  |  }
65  |  |  /**
66  |  |  End of File
67  |  |  */
```

61:20 | INS | unexpected token: ;

Lab1 Multi-GPIO Output

Code Example

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"

/*
    Main application
*/

uint32_t i=0;

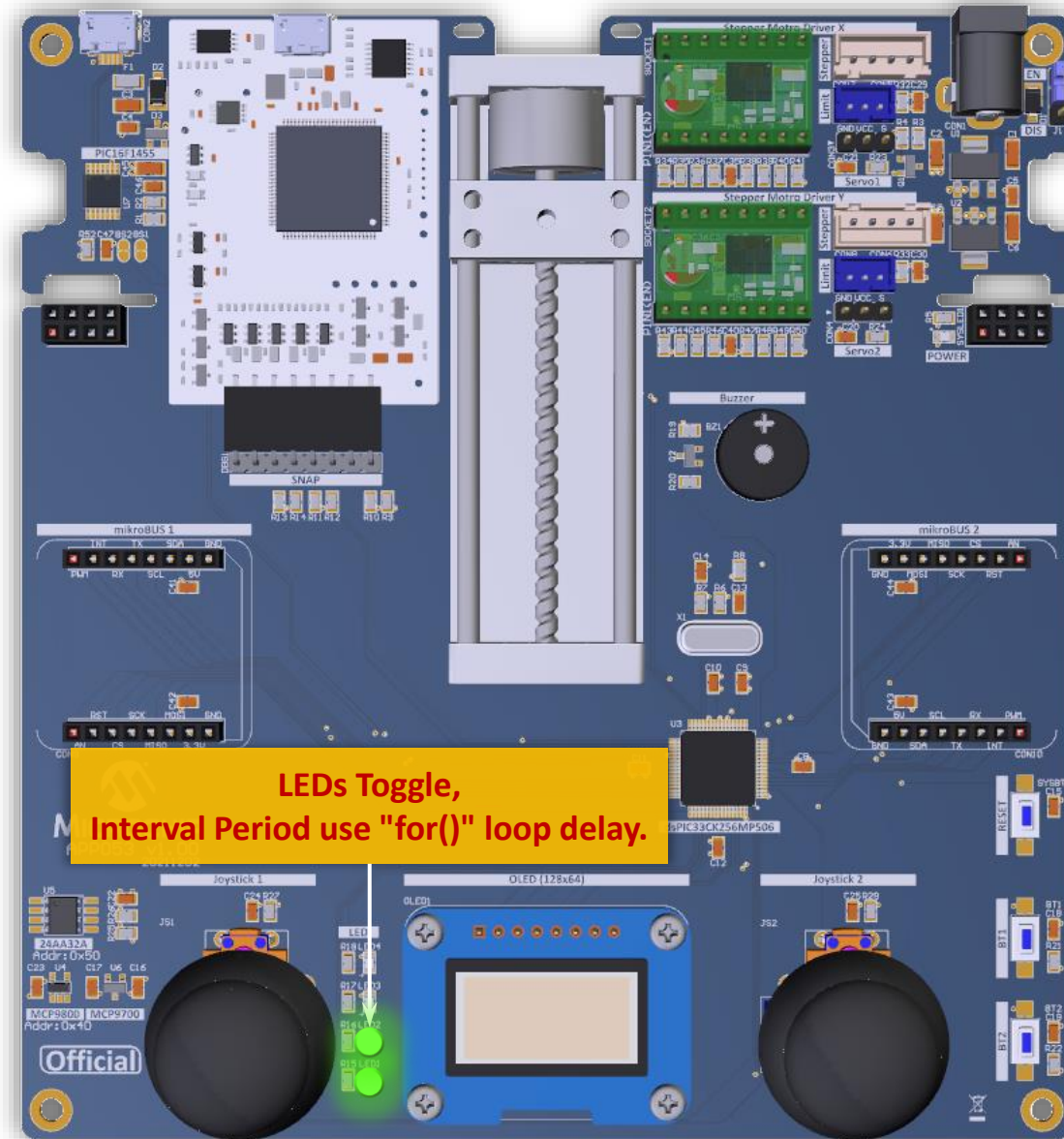
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    while (1)
    {
        // Add your application code
        LED1_Toggle();
        LED2_Toggle();
        for (i = 0; i < 80000; i++);
    }
    return 1;
}
/**
End of File
*/
```



```
main.c - Editor
main.c
Source History
48 #include "mcc_generated_files/system.h"
49 #include "mcc_generated_files/pin_manager.h"
50
51 /*
52 | Main application
53 */
54 uint32_t i=0;
55
56 int main(void)
57 {
58     // initialize the device
59     SYSTEM_Initialize();
60     while (1)
61     {
62         // Add your application code
63         LED1_Toggle();
64         LED2_Toggle();
65         for(i=0; i<80000; i++);
66     }
67     return 1;
68 }
69 /**
70 | End of File
71 */
```

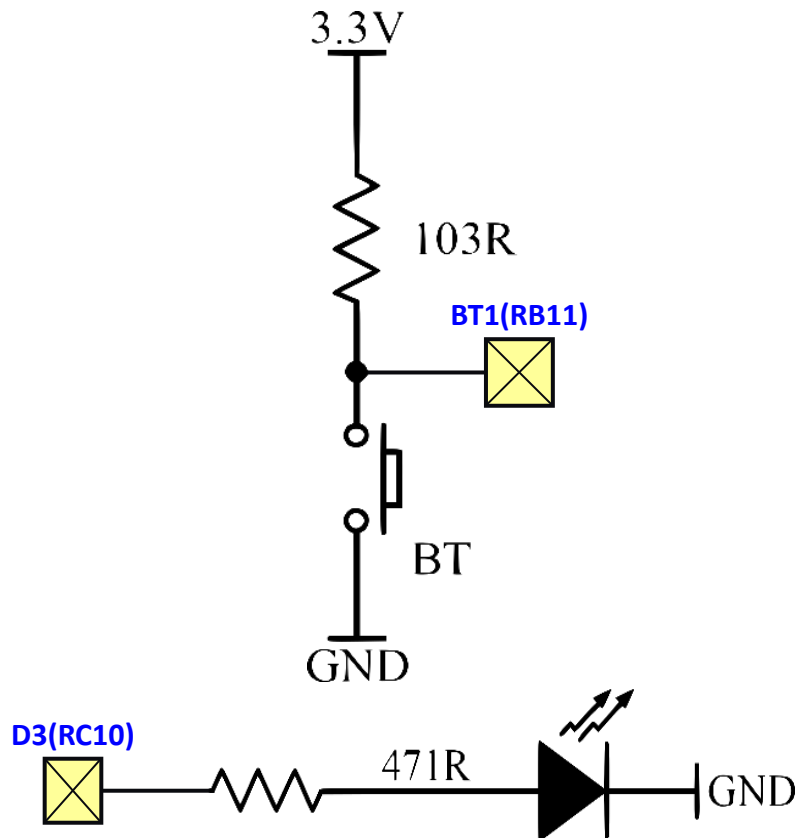

Lab1 Multi-GPIO Output

Result



Lab2 GPIO Input

- *CustomName_GetValue();* function is used for get digital input level from outside. The simple example code shows below:



```
if(BT1_GetValue())
{
    D3_SetLow();
}
else
{
    D3_SetHigh();
}
```

Lab2 GPIO Input

Lab2 GPIO Input

- Base on current project or Open `.\Lab2_xxx.X` to do exercises.
- Try to set **RB11** 、 **RB10** to digital input as get button status.
- Try use button to control LED light or dark.
 - BT1 Pressed > D3 Light, BT1 Released > D3 Dark.
 - BT2 Pressed > D4 Light, BT2 Released > D4 Dark.
- **RB11** to **Button(BT1)**, **RB10** to **Button(BT2)**.
- **RC10** to **LED(LED3)**, **RC11** to **LED(LED4)**.

Let's go!

Lab2 GPIO Input

Step 1

- Set **PORTB** : **Pin11 (RB11)**, **Pin10 (RB10)** to digital input mode.
- Set **PORTC** : **Pin10 (RC10)**, **Pin11 (RC11)** to digital output mode.

×

Pin Manager: Grid View ×

Package: TQFP64 ▾

Pin No: 46 47 48 49 61 62 63 64 1 2 13 22 23 27 50 51 24 32 36 37 52 53 3

Port B ▾

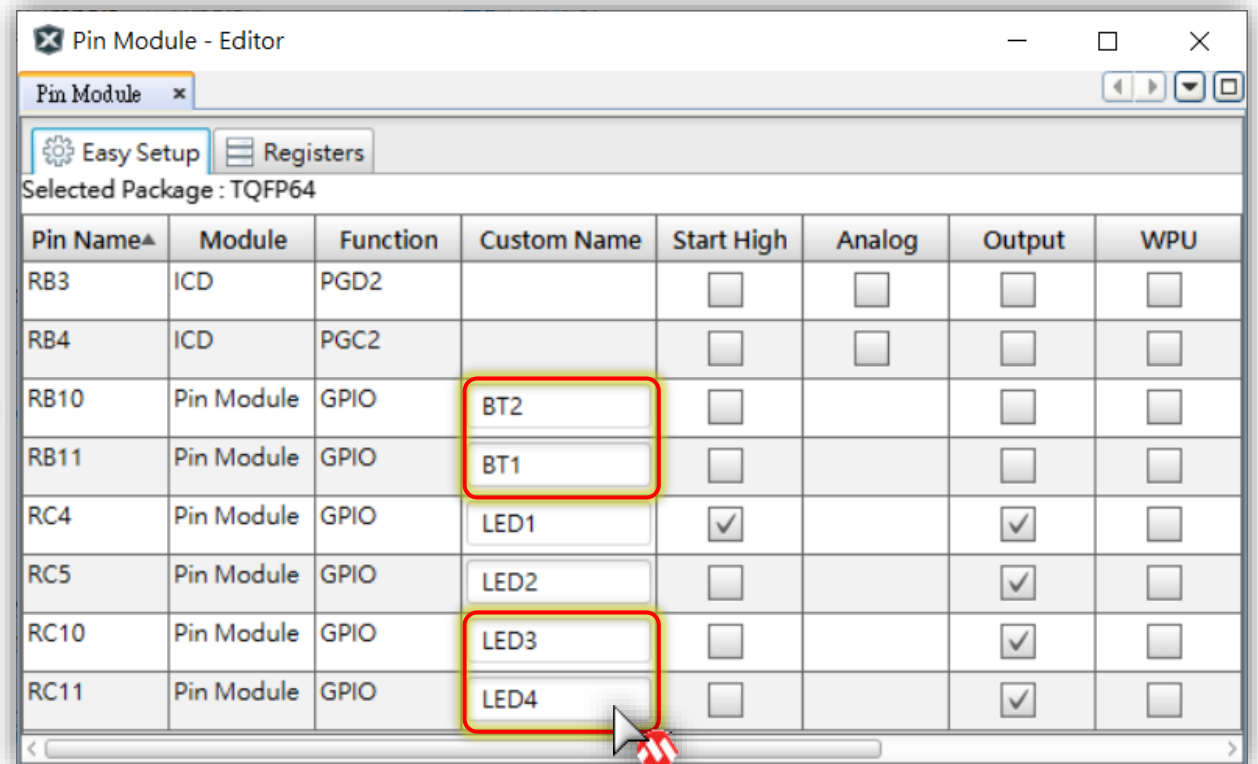
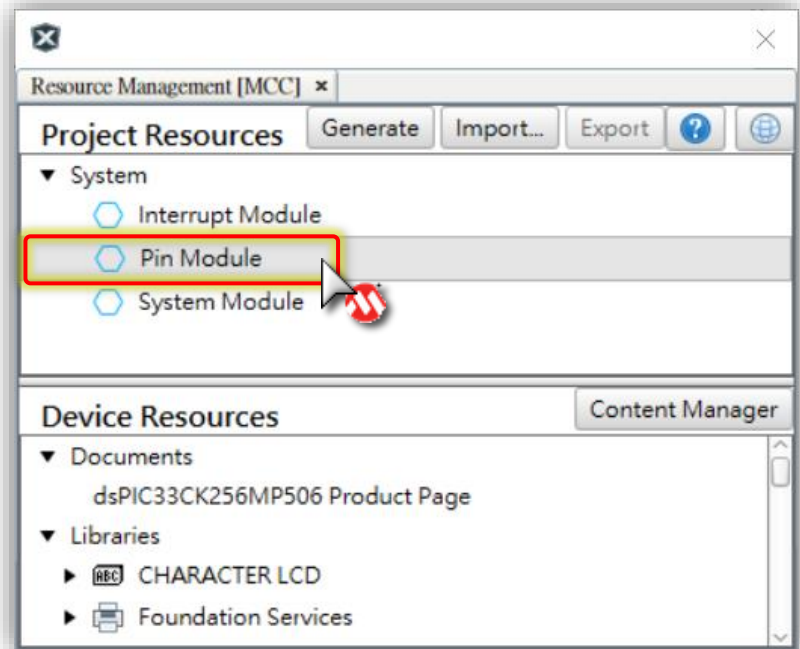
Port C ▾

Module	Function	Direction	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
Clock ▾	CLKI	input																							
	CLKO	output																							
	OSCI	input																							
	OSCO	output																							
	REFI1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	REFO1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ICD ▾	PGCx	input	🔒			🔒																			
	PGDx	input			🔒																				
Pin Module ▾	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒

Lab2 GPIO Input

Step 2

- Set Alias for RB10, RB11, RC10, RC11.
 - Pin Module : RB10 Custom Name > BT2, RB11 Custom Name > BT1.
 - Pin Module : RC10 Custom Name > LED3, RC11 Custom Name > LED4.

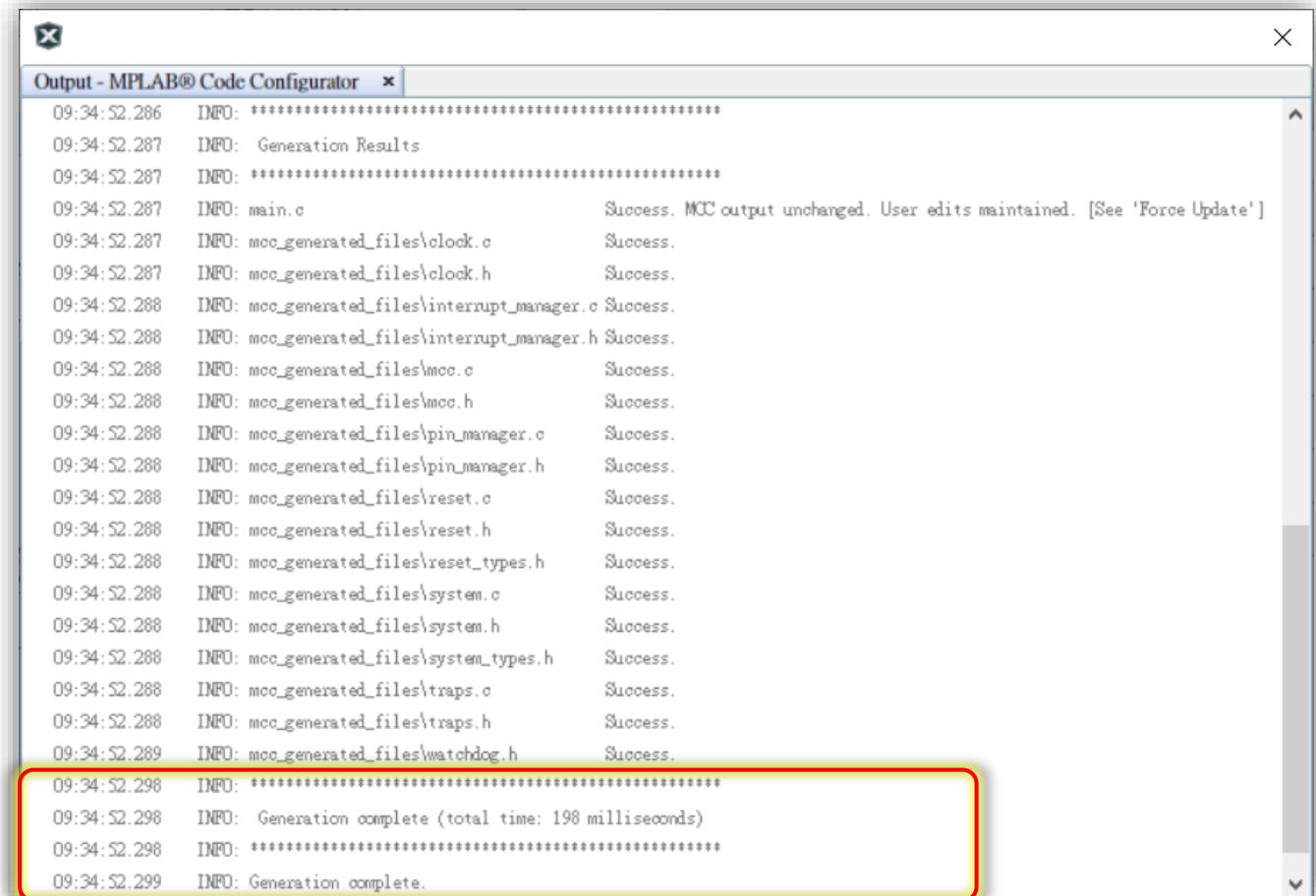
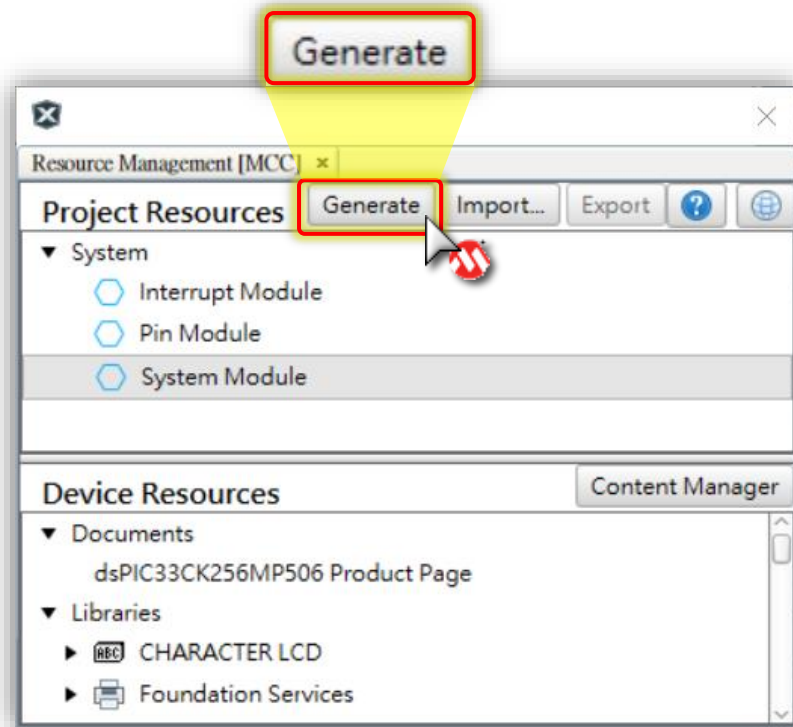


Click & Rename

Lab2 GPIO Input

Step 3

- Click "**Generate**" Button to generate your code.



Lab2 GPIO Input

Code Example

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"

/*
    Main application
*/
uint32_t i = 0;

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    while (1)
    {
        // Add your application code
        LED1_Toggle();
        LED2_Toggle();
        for (i = 0; i < 80000; i++);

        if(BT1_GetValue())
            LED3_SetLow();
        else
            LED3_SetHigh();

        if(BT2_GetValue())
            LED4_SetLow();
        else
            LED4_SetHigh();
    }
    return 1;
}
/**
End of File
*/
```

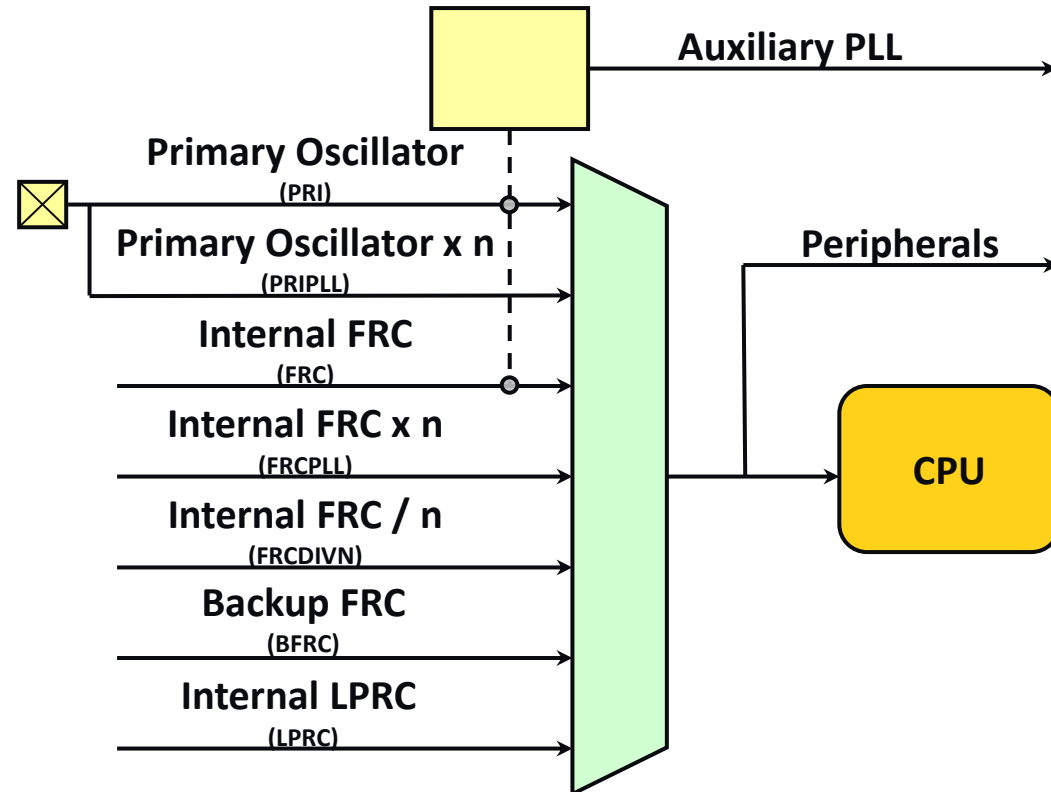
Result



Oscillator Architecture

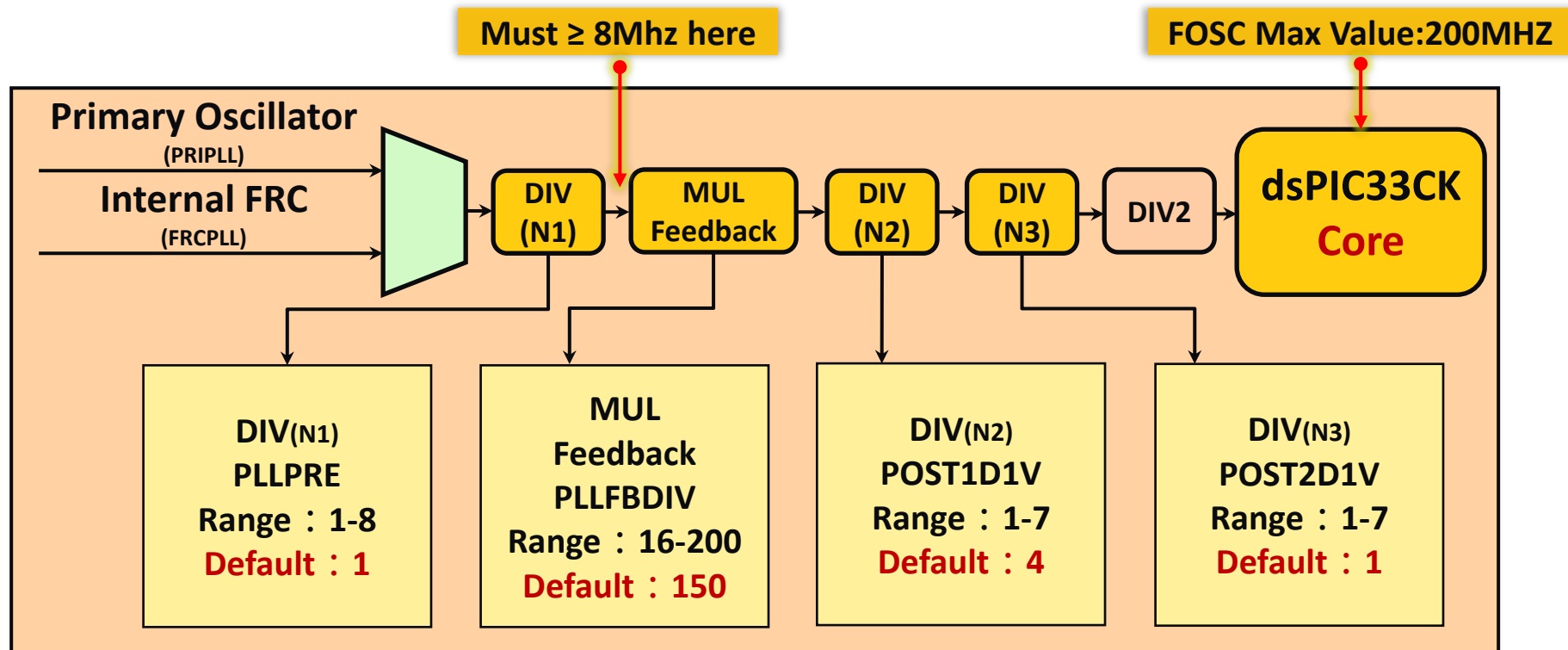
dsPIC33CK Series Clocks

- Microchip dsPIC33CK Series MCU available clock sources are **internal Fast RC**, **Low Power RC** and **External Crystal** or **Oscillator**, etc..
- The clocks can be divided or multiply by PLL.
- Auxiliary PLL (APLL) Clock Generator to Boost Operating Frequency for Peripherals
- Software-controllable switching between various clock sources.

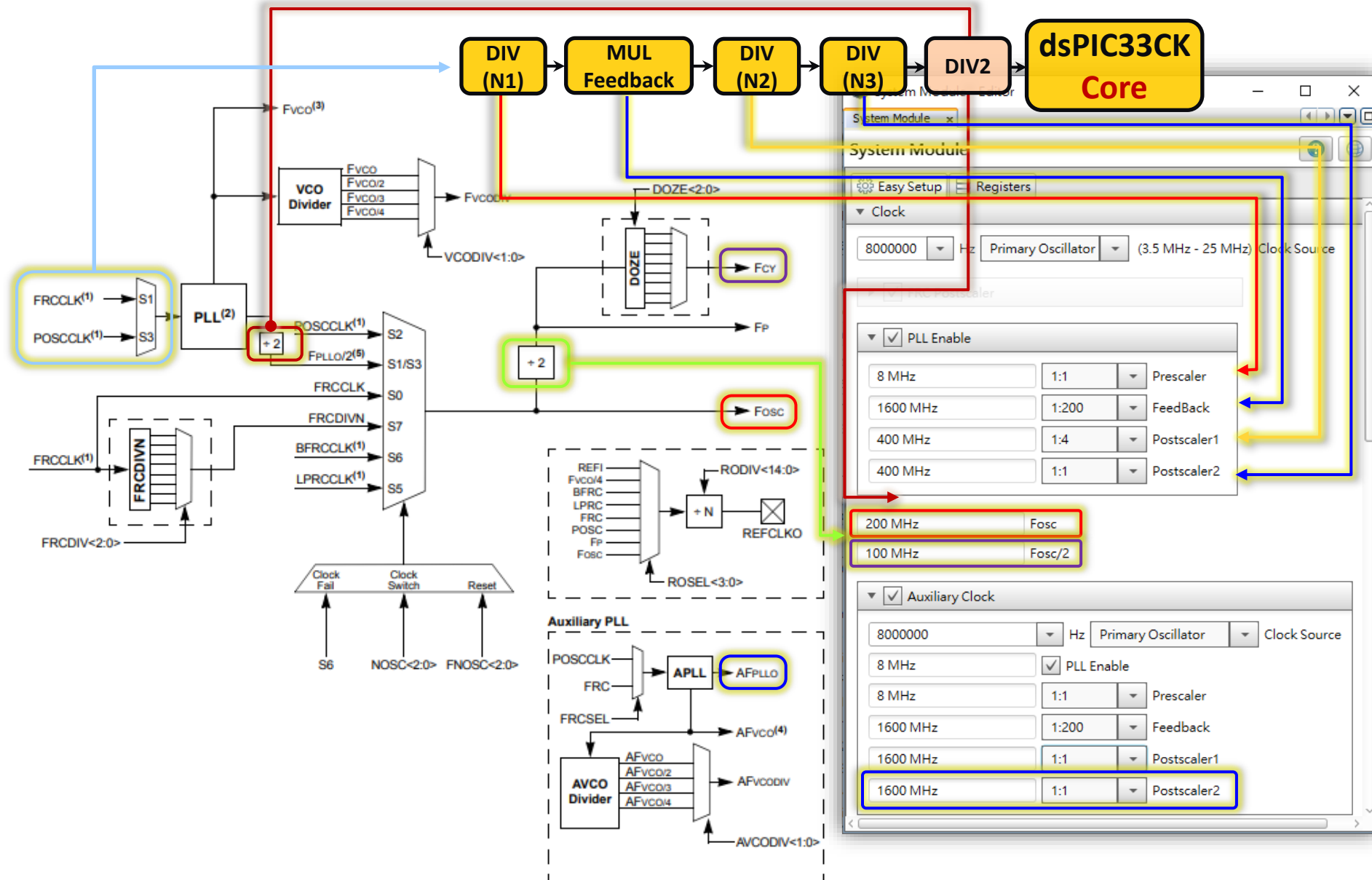


Phase Locked Loop

- Clock input must 8MHz after PLLPRE, which can use **Primary Oscillators** (Crystal or External Clock) or **Internal Fast RC** as clock source.



Oscillator Setting of MCC



Lab3 Clock PRIPLL

Lab3 Clock PRIPLL

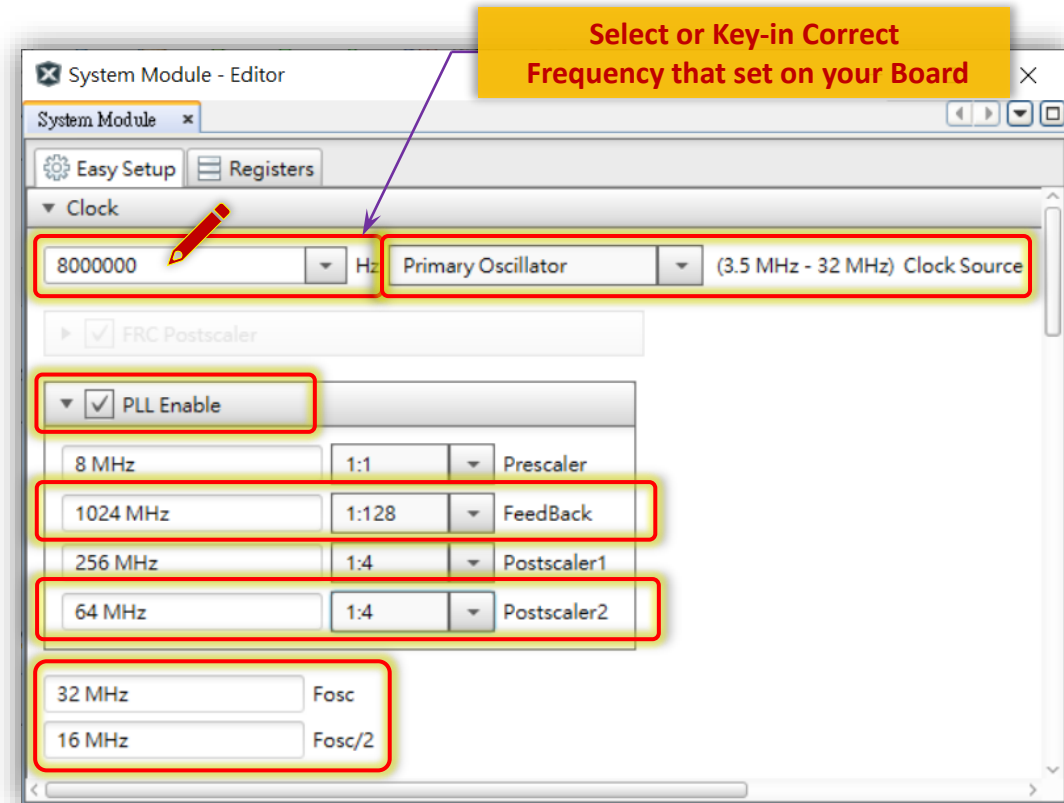
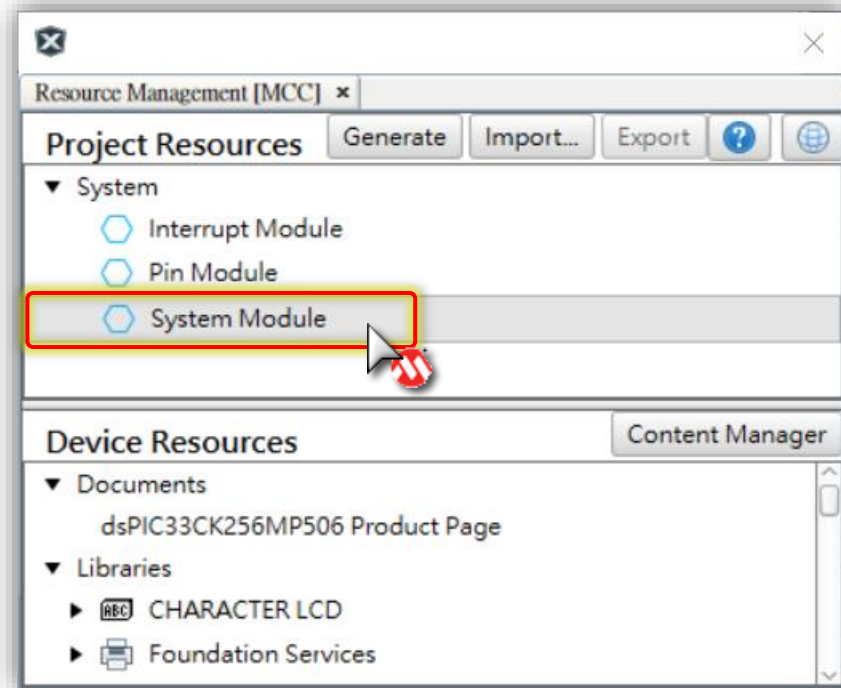
- Base on current project or Open . \Lab3_xxx.X to do exercises.
- Try to change system clock from internal Fast RC to **External Crystal (8MHz)** and **enable PLL**.
- Try to provide maximum frequency **(16MHz)** to MCU.

Let's go!

Lab3 Clock PRIPLL

Step 1

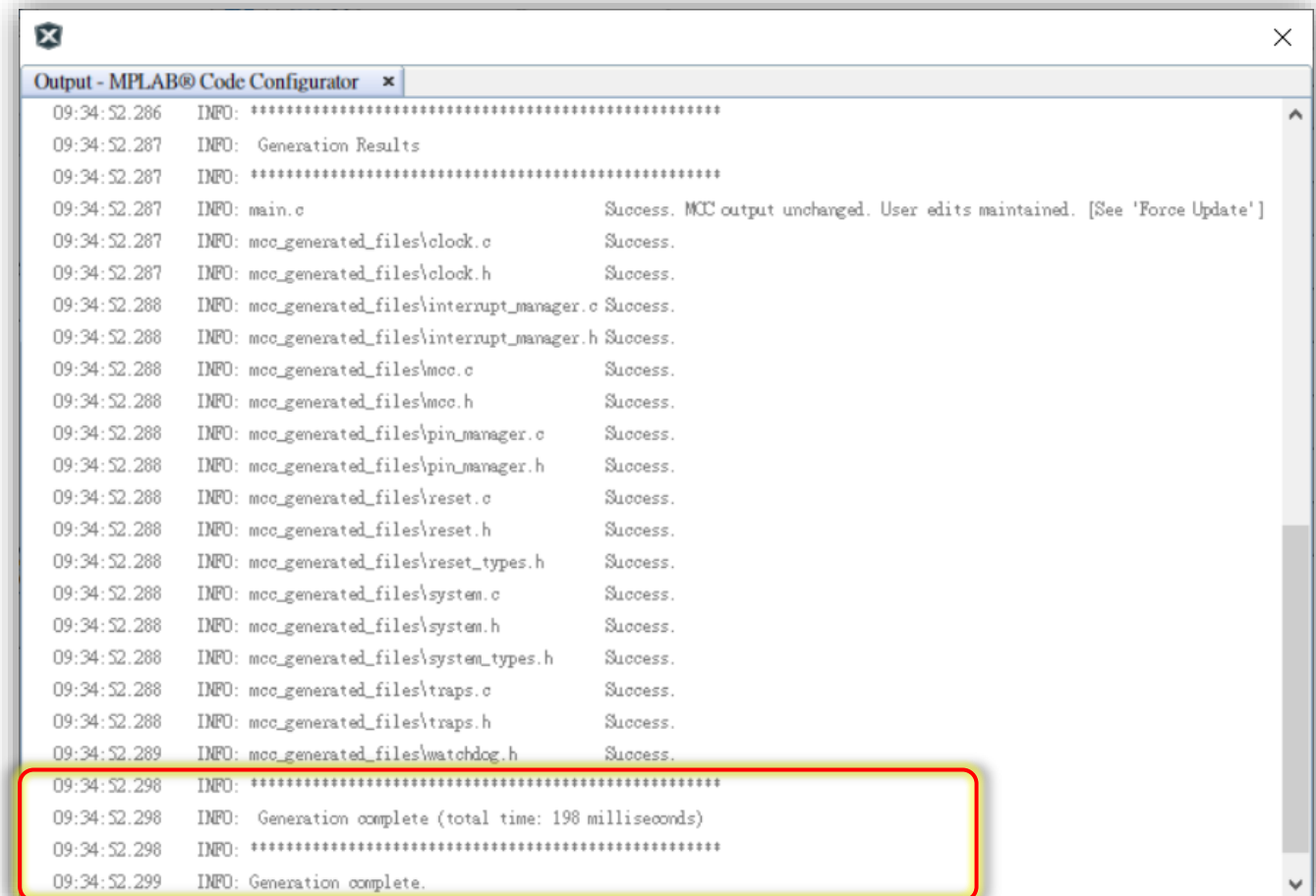
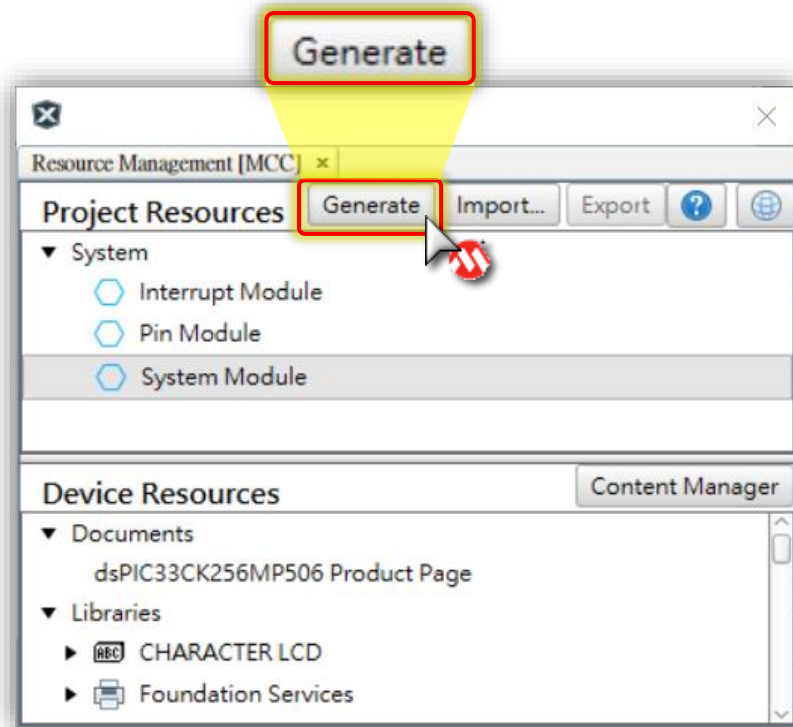
- Set System Module
 - Set **Primary Oscillator 8MHz**, enable PLL, **FeedBack > 1:128**, **Postscaler2 > 1:4**.



Lab3 Clock PRIPLL

Step 2

- Click "**Generate**" Button to generate your code.



Result



Interrupt Architecture

What's Interrupt ?

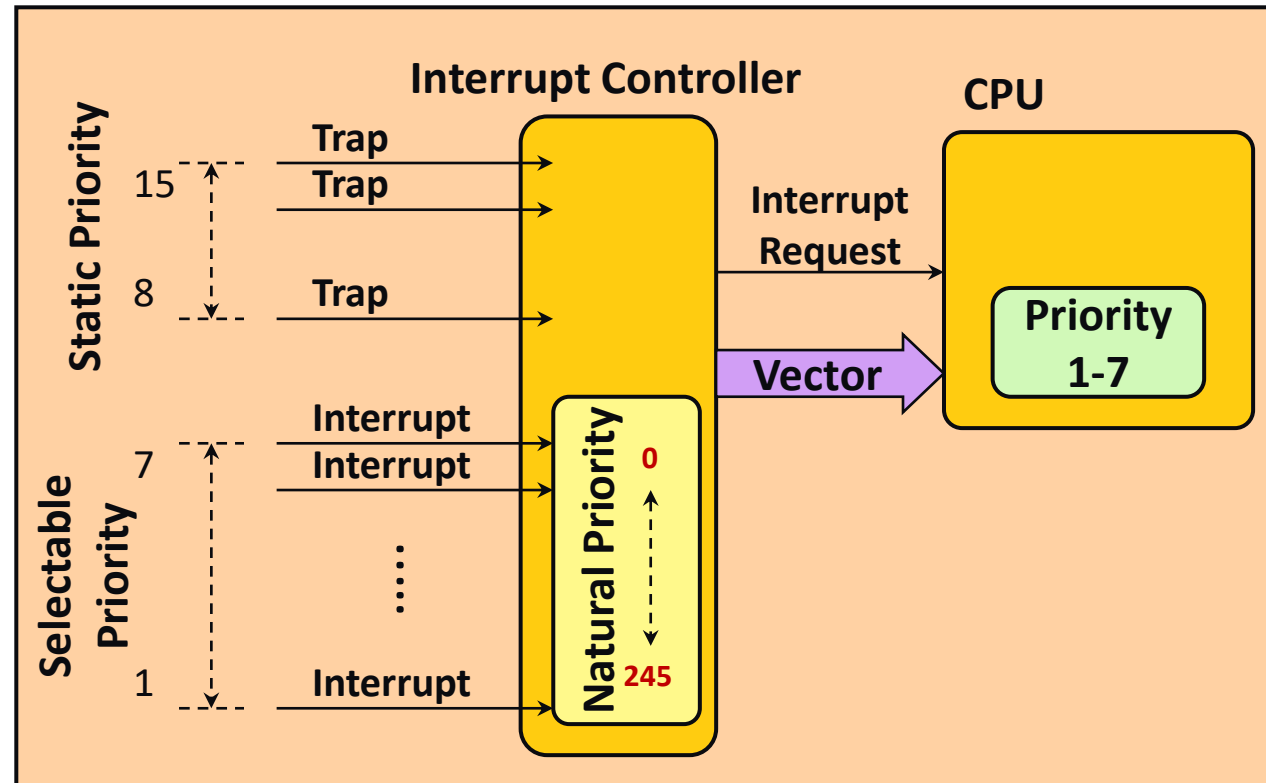
- In general, processor is executed in sequence according to the program's flow.
- However, if some peripherals or event needs immediate attention then an interrupt flag been set as high-priority and requiring interrupting current process to serve it.
- Processor will jump to execute the interrupt service routine (ISR, Interrupt Service Routine) to service peripherals or events.

Which events need Interrupt ?

- **For General (maskable)**
 - Time critical event
As soon as possible, when event occurred.
 - Unpredicted event
Polling too waste computing time.
- **For Emergency (non maskable)**
 - Stack Overflow/Underflow
 - Math Error
 - Address Error
 - etc..

dsPIC33CK Series Interrupt Architecture

- dsPIC33's Interrupt is Nested Vector Interrupt Controller.
- Provide up to 8 processor exceptions and software traps
- Provide 7 (1~7) user-selectable priority levels. **0 is disable, 7 is the Highest priority.**
- IVT/AIVT provide 246 vectors.
- Fixed interrupt entry and return latencies.



Natural Priority

- If two pending interrupts share the same priority, priority is given to the interrupt with the lowest exception number. ([.\Microchip\xc16\v2.00\docs\vector_docs\PIC33CK256MP506](#))

(lowest interrupt vector address).

Reset – GOTO Instruction	0x000000
Reset – GOTO Address	0x000002
Oscillator Fail Trap Vector	0x000004
Address Error Trap Vector	0x000006
Generic Hard Trap Vector	0x000008
Stack Error Trap Vector	0x00000A
Math Error Trap Vector	0x00000C
Reserved	0x00000E
Generic Soft Trap Vector	0x000010
Reserved	0x000012
Interrupt Vector 0	0x000014
Interrupt Vector 1	0x000016
:	:
:	:
:	:
:	:
Interrupt Vector 52	0x00007C
Interrupt Vector 53	0x00007E
Interrupt Vector 54	0x000080
:	:
:	:
:	:

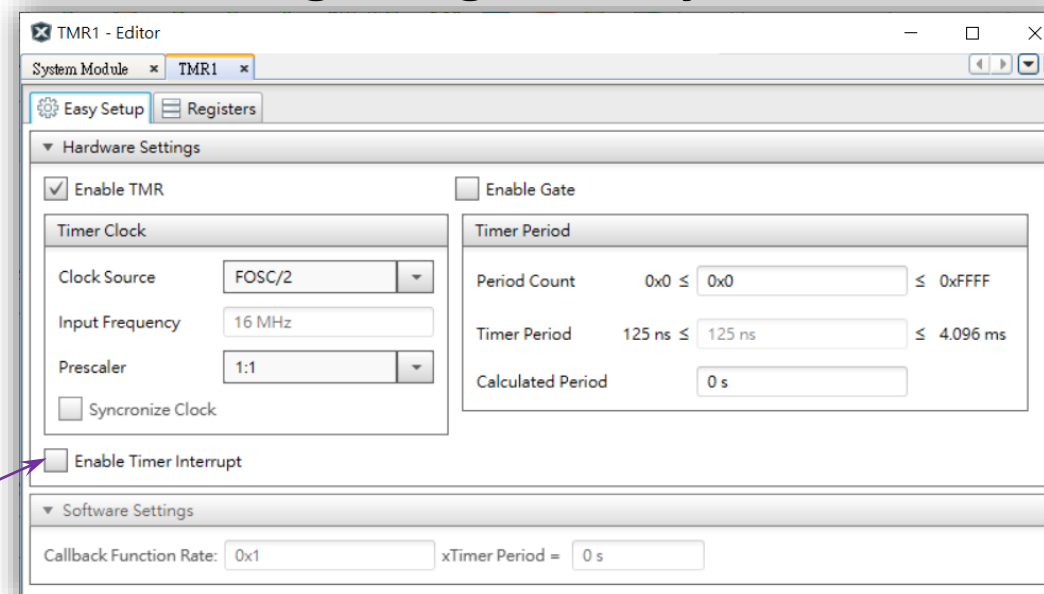
Interrupt Vector Details

IRQ#	Primary Name	Alternate Name	Vector Function
N/A	_OscillatorFail	_AltOscillatorFail	Oscillator Fail Trap vector
N/A	_AddressError	_AltAddressError	Address error Trap vector
N/A	_HardTrapError	_AltHardTrapError	Generic Hard Trap vector
N/A	_StackError	_AltStackError	Stack Error Trap Vector
N/A	_MathError	_AltMathError	Math Error Trap vector
N/A	_SoftTrapError	_AltSoftTrapError	Generic Soft Trap vector
0	_INT0Interrupt	_AltINT0Interrupt	External Interrupt 0
1	_T1Interrupt	_AltT1Interrupt	Timer 1
2	_CNAInterrupt	_AltCNAInterrupt	Change Notification A
3	_CNBInterrupt	_AltCNBInterrupt	Change Notification B
4	_DMA0Interrupt	_AltDMA0Interrupt	DMA Channel 0
6	_CCP1Interrupt	_AltCCP1Interrupt	CCP1 Capture/Compare Event
7	_CCT1Interrupt	_AltCCT1Interrupt	CCP1 Timer Event
8	_DMA1Interrupt	_AltDMA1Interrupt	DMA Channel 1
9	_SPI1RXInterrupt	_AltSPI1RXInterrupt	SPI1 RX
10	_SPI1TXInterrupt	_AltSPI1TXInterrupt	SPI1 TX
11	_U1RXInterrupt	_AltU1RXInterrupt	UART1 RX
12	_U1TXInterrupt	_AltU1TXInterrupt	UART1 TX

Each peripheral has an assigned.
Vector table is based at address 0x00000000

Coding for Interrupt

- **A complete interrupt code segment includes below 3 parts.**
 - Interrupt Service Routine (ISR)
 - Enable Interrupt and Set Interrupt Priority
 - Initial Peripheral to generate Interrupt Request.
- **This is very difficult for beginner, if you use bare metal style to build your code.**
- **MCC could help you to build most all code segment. So, you can get easy to handling interrupts.**
- **Just one click all interrupt relate setting will get ready.**



MCC's Interrupt Function

- MCC enable interrupt, set priority and create interrupt service routine automatically.

Interrupt Base

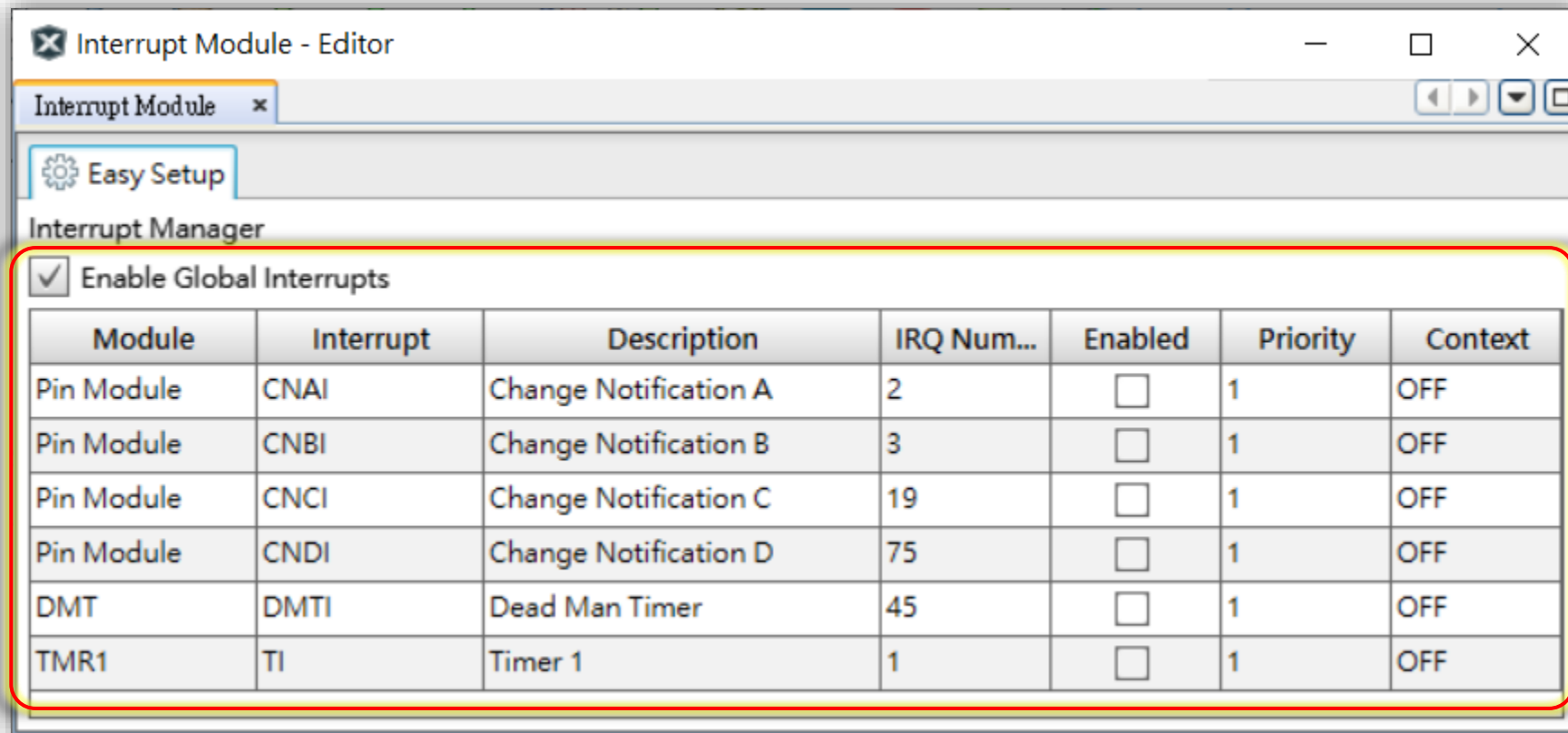
```
tmr1.c - Editor
tmr1.c x
Source History
91 void TMR1_Initialize (void)
92 {
93     //TMR 0;
94     TMR1 = 0x00;
95     //Period = 0.2 s; Frequency = 16000000 Hz; PR 49999;
96     PR1 = 0xC34F;
97     //TCKPS 1:64; PRWIP Write complete; TMWIP Write complete;
98     //TON enabled; TSIDL disabled; TCS FOSC/2; TECS TICK; TSYNC disabled;
99     //TMWDIS disabled; TGATE disabled;
100     T1CON = 0x8020;
101
102     if(TMR1_InterruptHandler == NULL)
103     {
104         TMR1_SetInterruptHandler(&TMR1_Callback);
105     }
106
107     IFS0bits.T1IF = false;
108     IEC0bits.T1IE = true;
109
110     tmr1_obj.timerElapsed = false;
111 }
112
```

ISR

```
tmr1.c - Editor
tmr1.c x
Source History
114
115 void __attribute__ ( ( interrupt, no_auto_psv ) ) _T1Interrupt ( )
116 {
117     /* Check if the Timer Interrupt/Status is set */
118
119     /***User Area Begin
120
121     // ticker function call;
122     // ticker is 1 -> Callback function gets called everytime this ISR executes
123     if(TMR1_InterruptHandler)
124     {
125         TMR1_InterruptHandler();
126     }
127
128     /***User Area End
129
130     tmr1_obj.count++;
131     tmr1_obj.timerElapsed = true;
132     IFS0bits.T1IF = false;
133 }
134
```


MCC's Interrupt Manager

- MCC's Interrupt Module allow user to manager interrupt enable/disable and priority.



Callback Function

- Most interrupt handler function provide named "**callback**" function for user.
- Flag and event handle by handler function automatically, user focus at application only. It's more powerful and easy to understand than operating a bare metal style.
- The callback function own named "**weak**" qualification. This is mean user can redefine your same name function in your code to replace the default function.

```
void __attribute__((weak)) XXX_Callback(void)
{
    // Add your custom callback code here
}
```

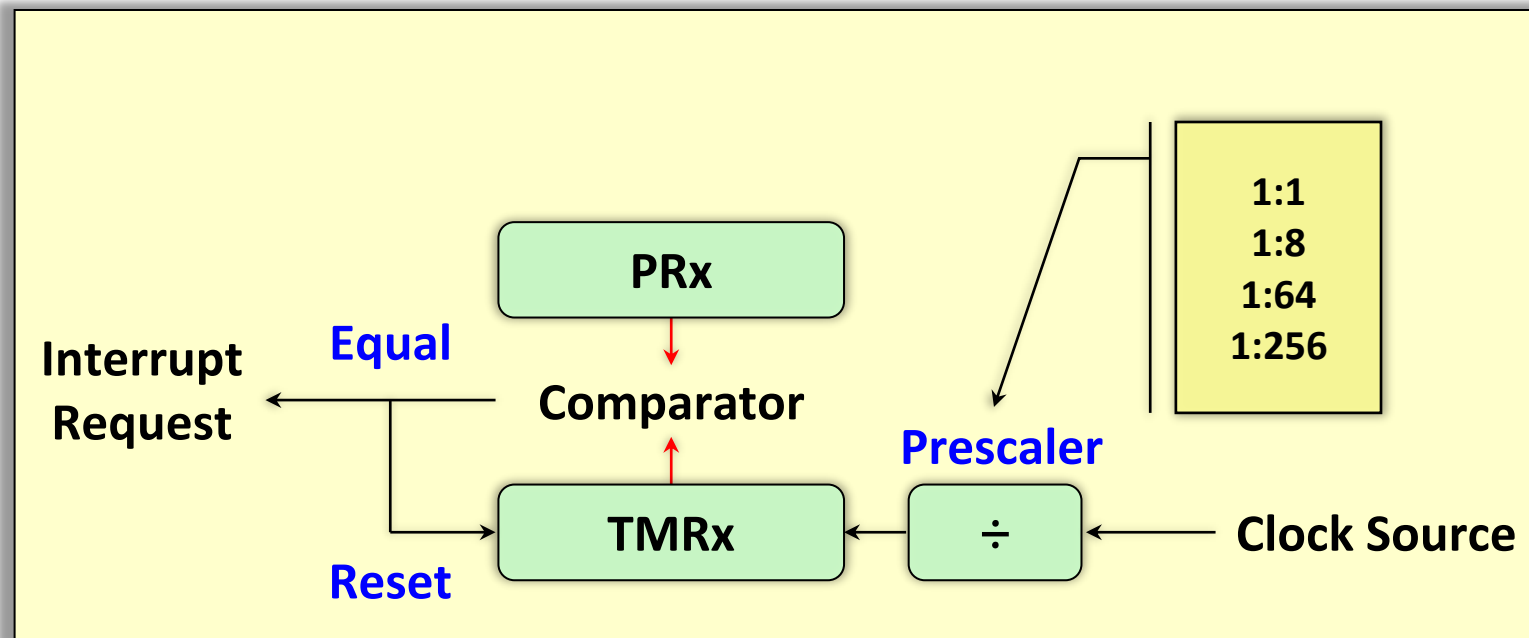
Timer Architecture

Timer or Counter ?

- **Counter or Timer is a clock counter, use to count how many clocks into module after start.**
- **There are two kinds of generally called Timer or Counter. In fact, it's same.**
 - The Input Clock unknown : Just know how many counts, called Counter.
 - The Input Clock known : Can use count and clock period to calculate time, called Timer.
- **Timer can set a notify condition, like overflow, equal some value etc.**
 - If interrupt disable user must check flag, as soon as possible.
 - If interrupt enable : Timer will notify CPU, if condition is true.
- **dsPIC33CKMP506 provide 1 timer named "TMR1" only.**

dsPIC33CK Timer/Counter

- Please refer to the block diagram, Just a concept. The real and detail block diagram please refer to the following slide.
- Clock into TMRx after prescaler. Comparator continuously compares the values of **TMRx** and **PRx**. Asserts a into the interrupt request when equal, and resets TMRx to zero.
- You can fill a value to **PRx** to determine period you want.



Callback and Function Rate

- Timer defined named "**void TMR1_CallBack(void)**" **weak function**. Just define same name function to replace it.

```
void __attribute__((weak)) TMR1_CallBack(void)
{
    // Add your custom callback code here
}
```

- Timer's callback function rate is a software counter. Hardware always Limited. It can't set a long period.
- The "Rate" will involve software counter to extend your hardware timer period.

```
void __attribute__((interrupt, no_auto_psv)) _T1Interrupt()
{
    static volatile unsigned int CountCallBack = 0;
    if (++CountCallBack >= TMR1_INTERRUPT_TICKER_FACTOR)
    {
        TMR1_CallBack();
        CountCallBack = 0;
    }
    ....
}
```

extend your timer period by software artifice.

▼ Software Settings

Callback Function Rate: xTimer Period =

Lab4 Timer1 Interrupt

Lab4 Timer1 Interrupt

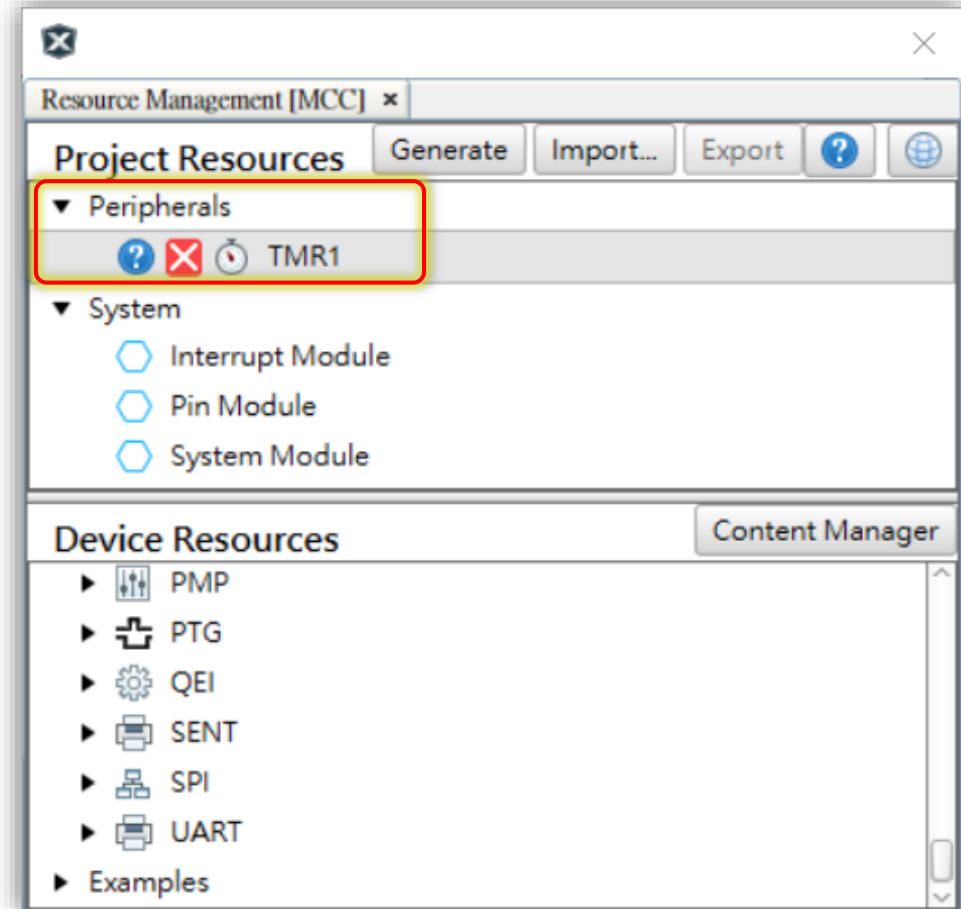
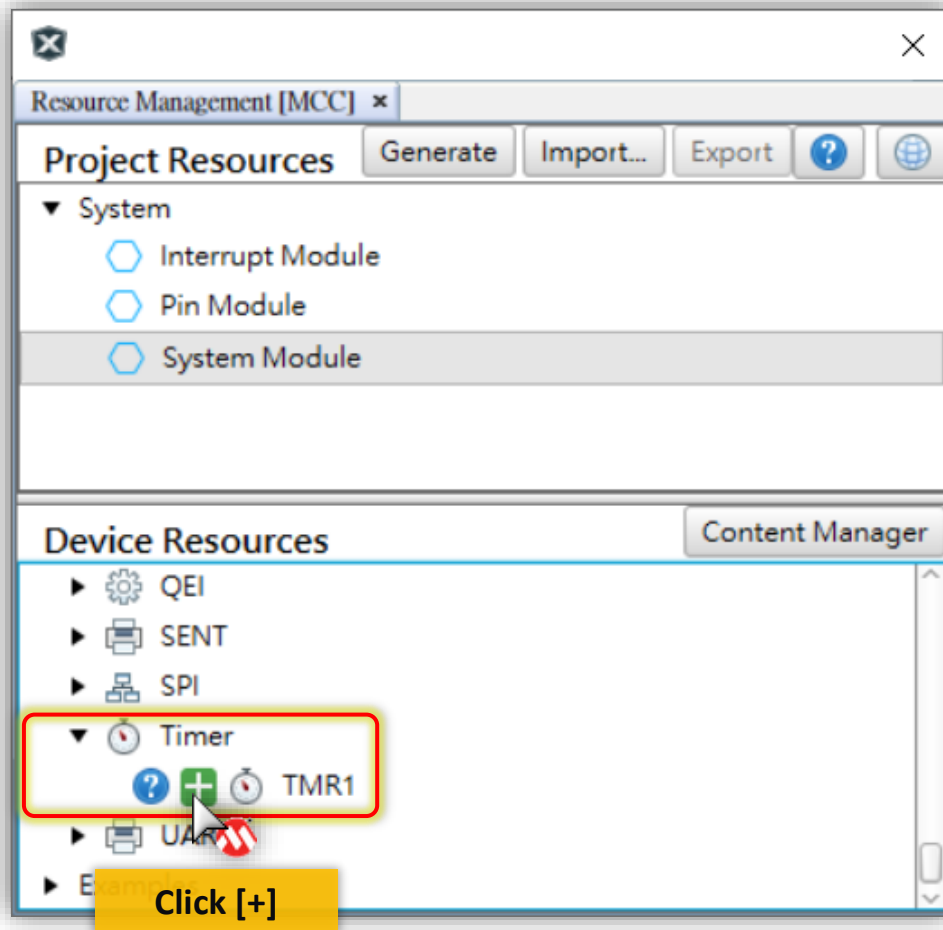
- Base on current project or Open `.\Lab4_xxx.X` to do exercises.
- Try to add TMR1 to replace software delay to control LEDs toggle period.
- The Timer1 setup to timer mode, timer period is 200ms (**100ms x 2**).
- Timer1 interrupt priority set to level **1**.

Let's go!

Lab4 Timer1 Interrupt

Step 1

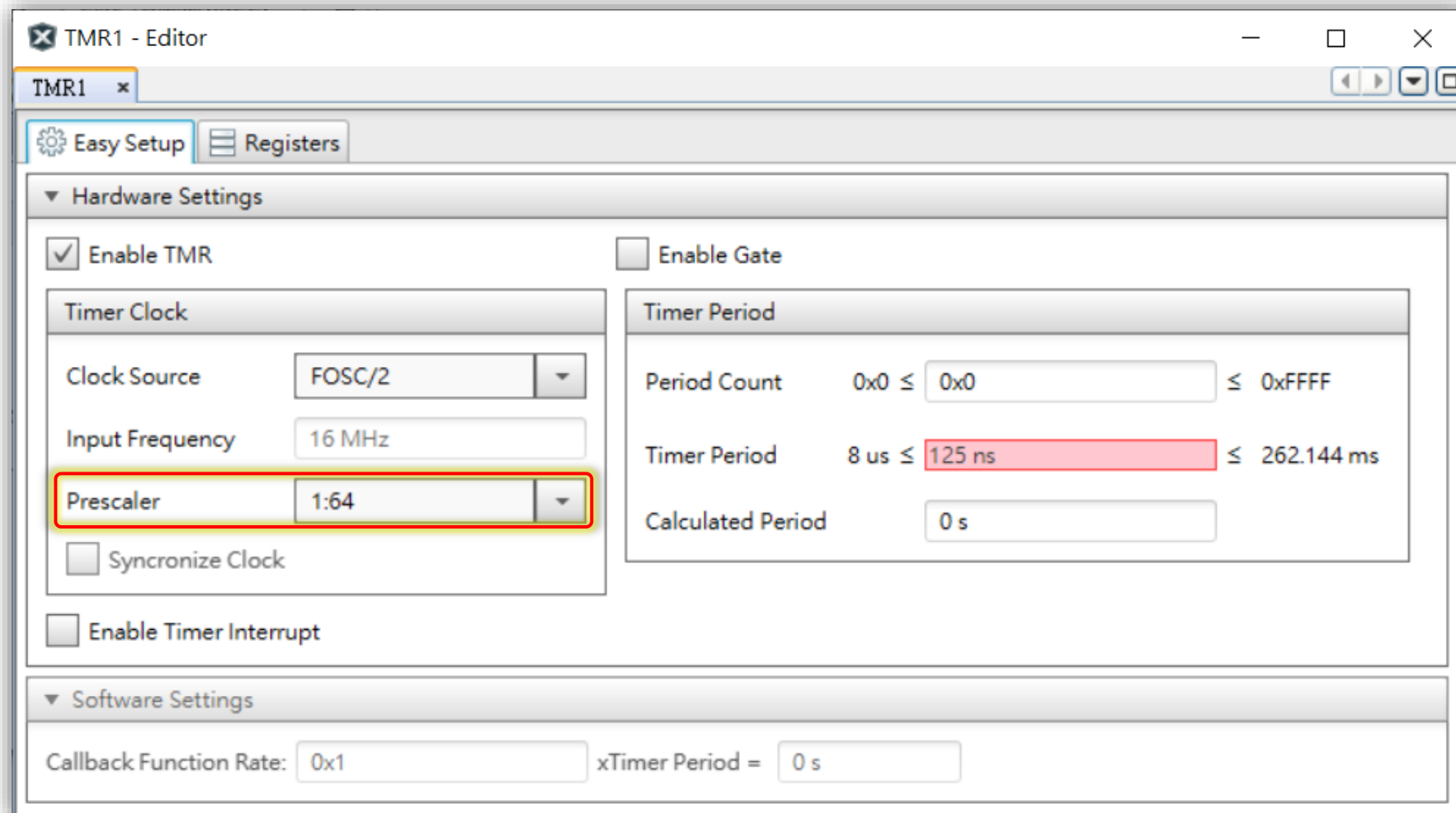
- Click [+] to add **Timer** > **TMR1** Module.



Lab4 Timer1 Interrupt

Step 2

- Set Prescaler : 1:1 > 1:64.



The screenshot shows the 'TMR1 - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing the following configuration:

- ☒ Enable TMR
- ☐ Enable Gate
- Timer Clock**
 - Clock Source: FOSC/2
 - Input Frequency: 16 MHz
 - Prescaler: 1:64** (highlighted with a red box)
 - ☐ Synchronize Clock
- ☐ Enable Timer Interrupt

The 'Timer Period' section shows the following values:

- Period Count: 0x0 ≤ 0x0 ≤ 0xFFFF
- Timer Period: 8 us ≤ 125 ns ≤ 262.144 ms
- Calculated Period: 0 s

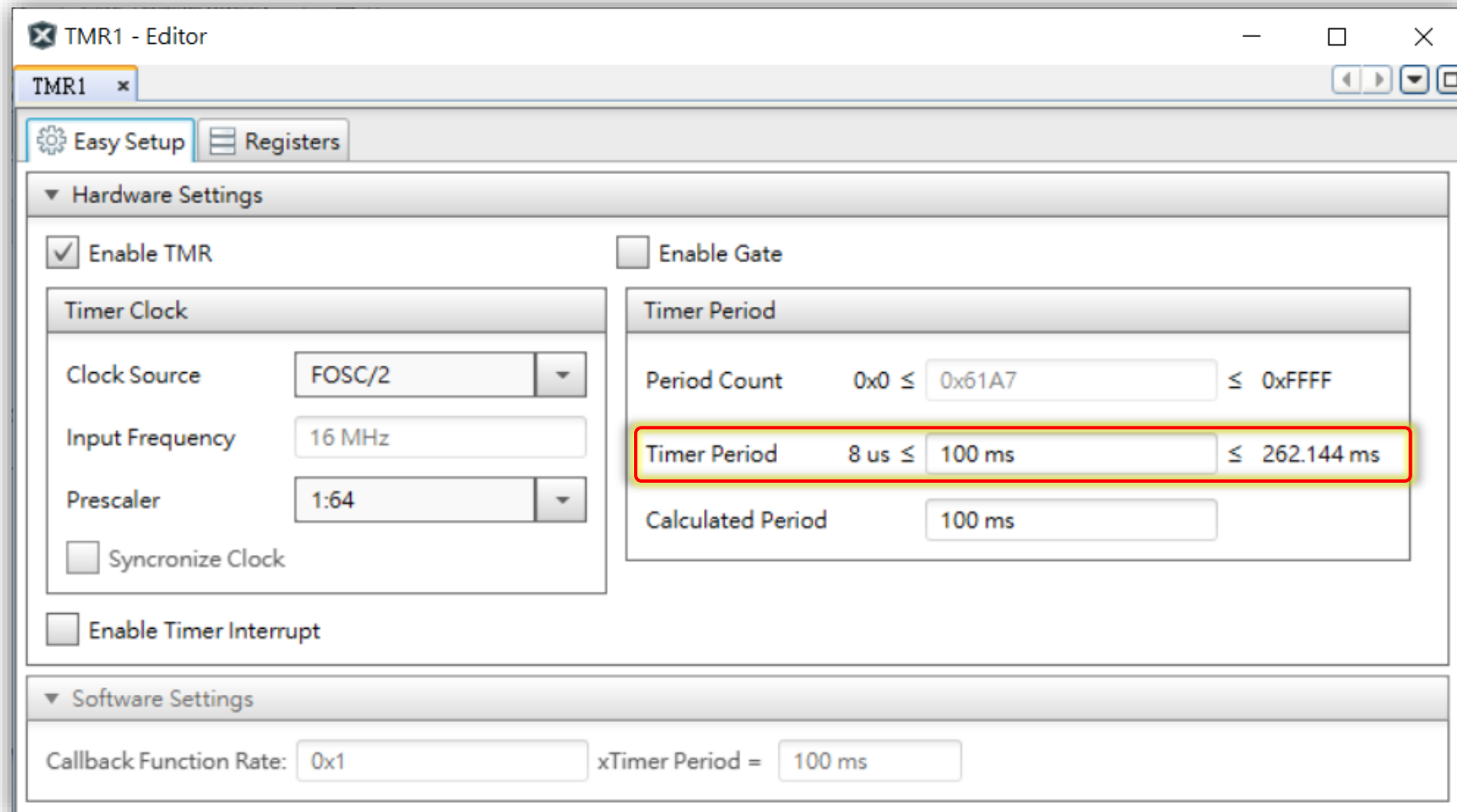
The 'Software Settings' section is also expanded, showing:

- Callback Function Rate: 0x1
- xTimer Period = 0 s

Lab4 Timer1 Interrupt

Step 3

- Set Time Period : 125ns > 100ms.



The screenshot shows the 'TMR1 - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing the following configuration:

- ☒ Enable TMR
- ☐ Enable Gate
- Timer Clock:**
 - Clock Source: FOSC/2
 - Input Frequency: 16 MHz
 - Prescaler: 1:64
 - ☐ Synchronize Clock
- ☐ Enable Timer Interrupt
- Timer Period:**
 - Period Count: 0x0 ≤ 0x61A7 ≤ 0xFFFF
 - Timer Period: 8 us ≤ 100 ms ≤ 262.144 ms** (highlighted with a red box)
 - Calculated Period: 100 ms

The 'Software Settings' section is also expanded, showing:

- Callback Function Rate: 0x1
- xTimer Period = 100 ms

Lab4 Timer1 Interrupt

Step 4

- Set **Enable Timer Interrupt**.

TMR1 - Editor

TMR1 x

Easy Setup Registers

▼ Hardware Settings

☒ Enable TMR ☐ Enable Gate

Timer Clock

Clock Source: FOSC/2

Input Frequency: 16 MHz

Prescaler: 1:64

☐ Synchronize Clock

Timer Period

Period Count: $0x0 \leq 0x61A7 \leq 0xFFFF$

Timer Period: $8 \mu s \leq 100 \text{ ms} \leq 262.144 \text{ ms}$

Calculated Period: 100 ms

☒ Enable Timer Interrupt

▼ Software Settings

Callback Function Rate: 0x1 xTimer Period = 100 ms

Lab4 Timer1 Interrupt

Step 5

- Set Function Rate : **0x01 > 0x02**

The screenshot shows the 'TMR1 - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing the following configuration:

- ☒ Enable TMR
- ☐ Enable Gate
- Timer Clock:**
 - Clock Source: FOSC/2
 - Input Frequency: 16 MHz
 - Prescaler: 1:64
 - ☐ Synchronize Clock
- Timer Period:**
 - Period Count: $0x0 \leq 0x61A7 \leq 0xFFFF$
 - Timer Period: $8 \mu s \leq 100 \text{ ms} \leq 262.144 \text{ ms}$
 - Calculated Period: 100 ms
- ☒ Enable Timer Interrupt

The 'Software Settings' section is also expanded, showing:

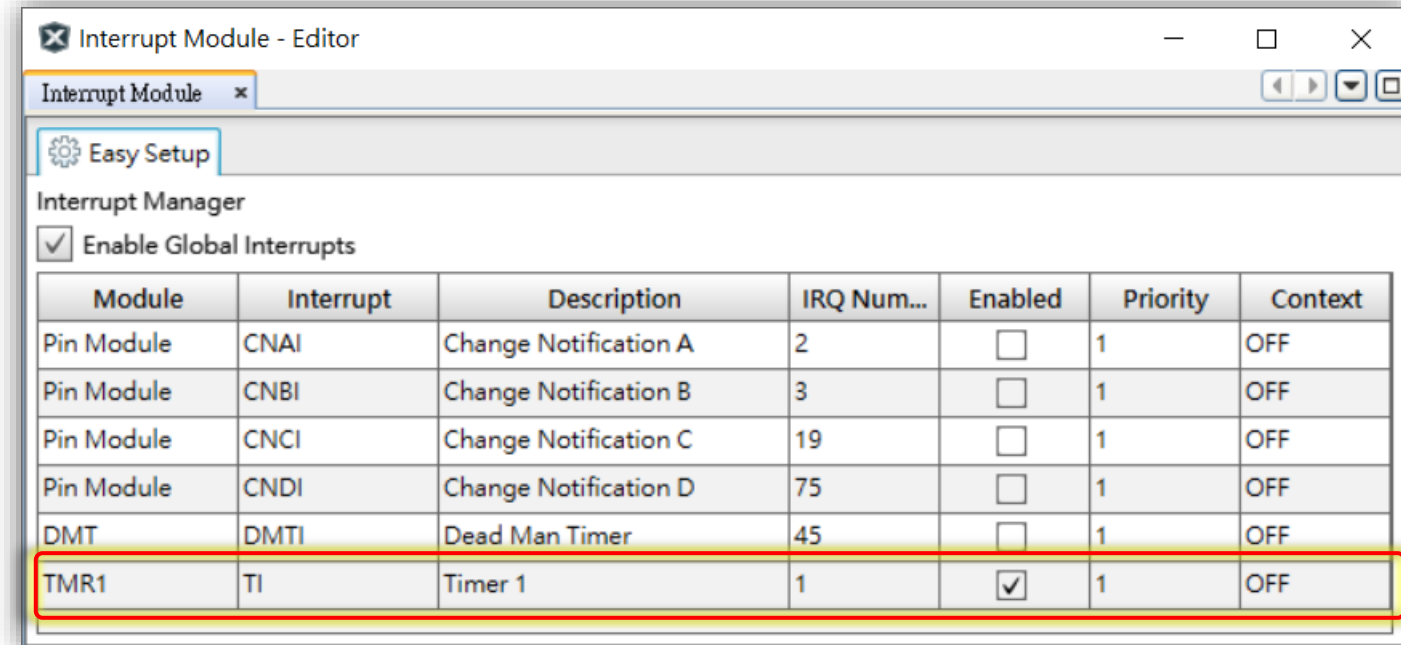
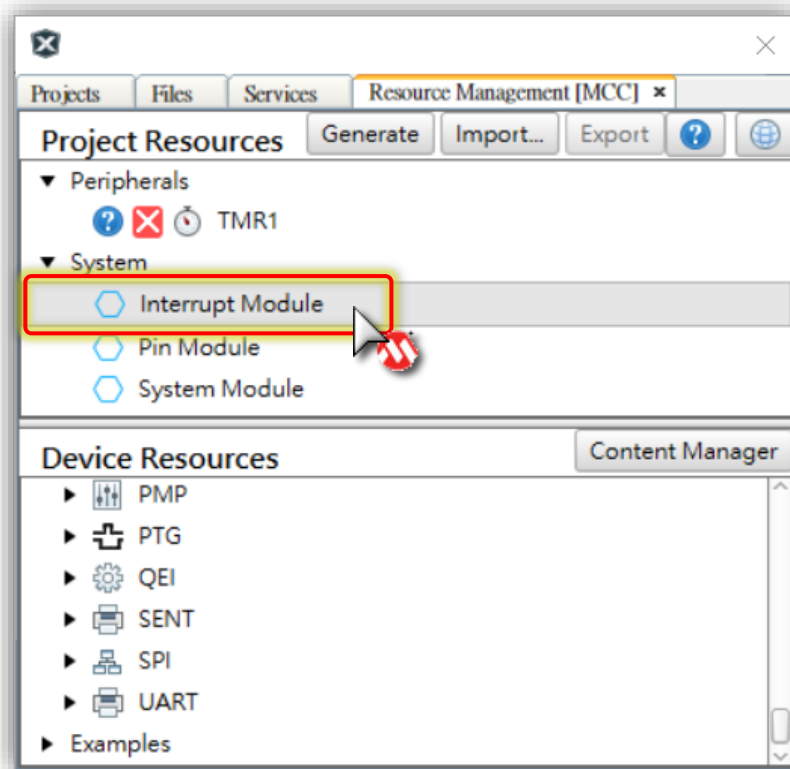
- Callback Function Rate: 0x2
- xTimer Period = 200 ms

The 'Callback Function Rate' and 'xTimer Period' fields are highlighted with a red rectangle.

Lab4 Timer1 Interrupt

Step 6

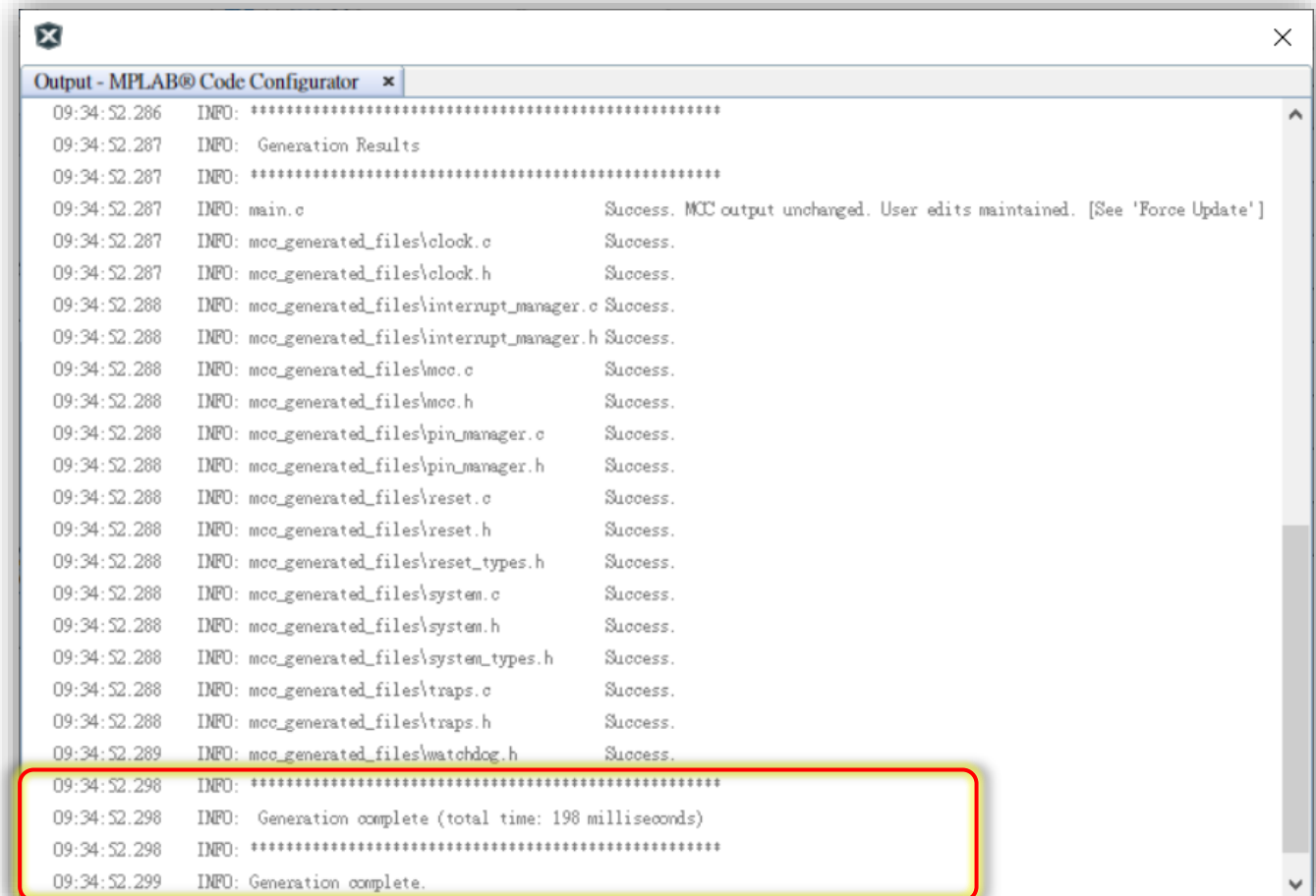
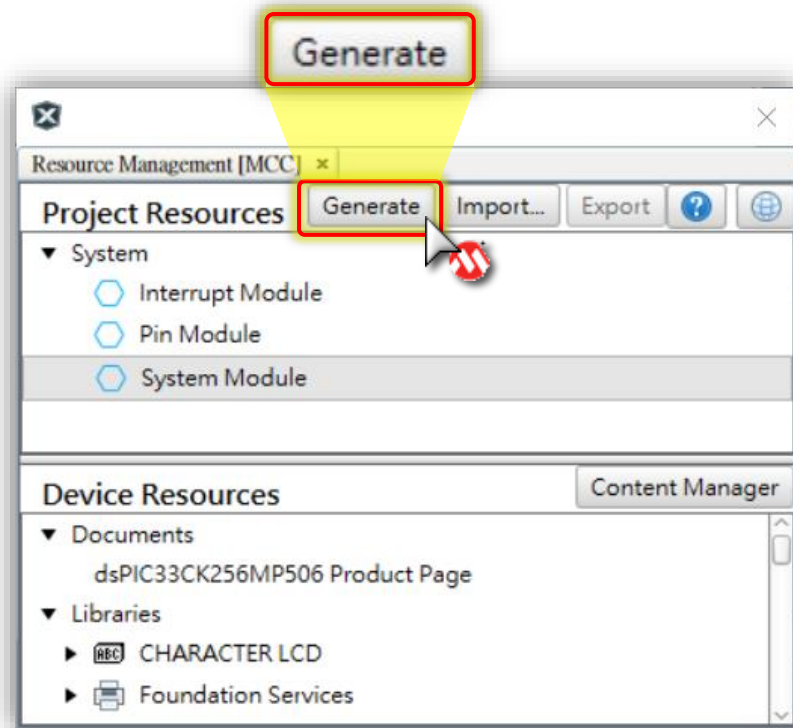
- Check **Enable TMR1 Interrupt**.



Lab4 Timer1 Interrupt

Step 7

- Click "**Generate**" Button to generate your code.



Timer Functions of MCC

- MCC Provide below functions for Timer:

Prototype definition : "tmr1.h"

- void TMR1_Initialize (void);
- void TMR1_Period16BitSet(uint16_t value);
- uint16_t TMR1_Period16BitGet(void);
- void TMR1_Counter16BitSet (uint16_t value);
- uint16_t TMR1_Counter16BitGet(void);
- void TMR1_Callback(void);
- void TMR1_Start(void);
- void TMR1_Stop(void);
- bool TMR1_GetElapsedThenClear(void);
- int TMR1_SoftwareCounterGet(void);
- void TMR1_SoftwareCounterClear(void);

Lab4 Timer1 Interrupt

Code Example

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

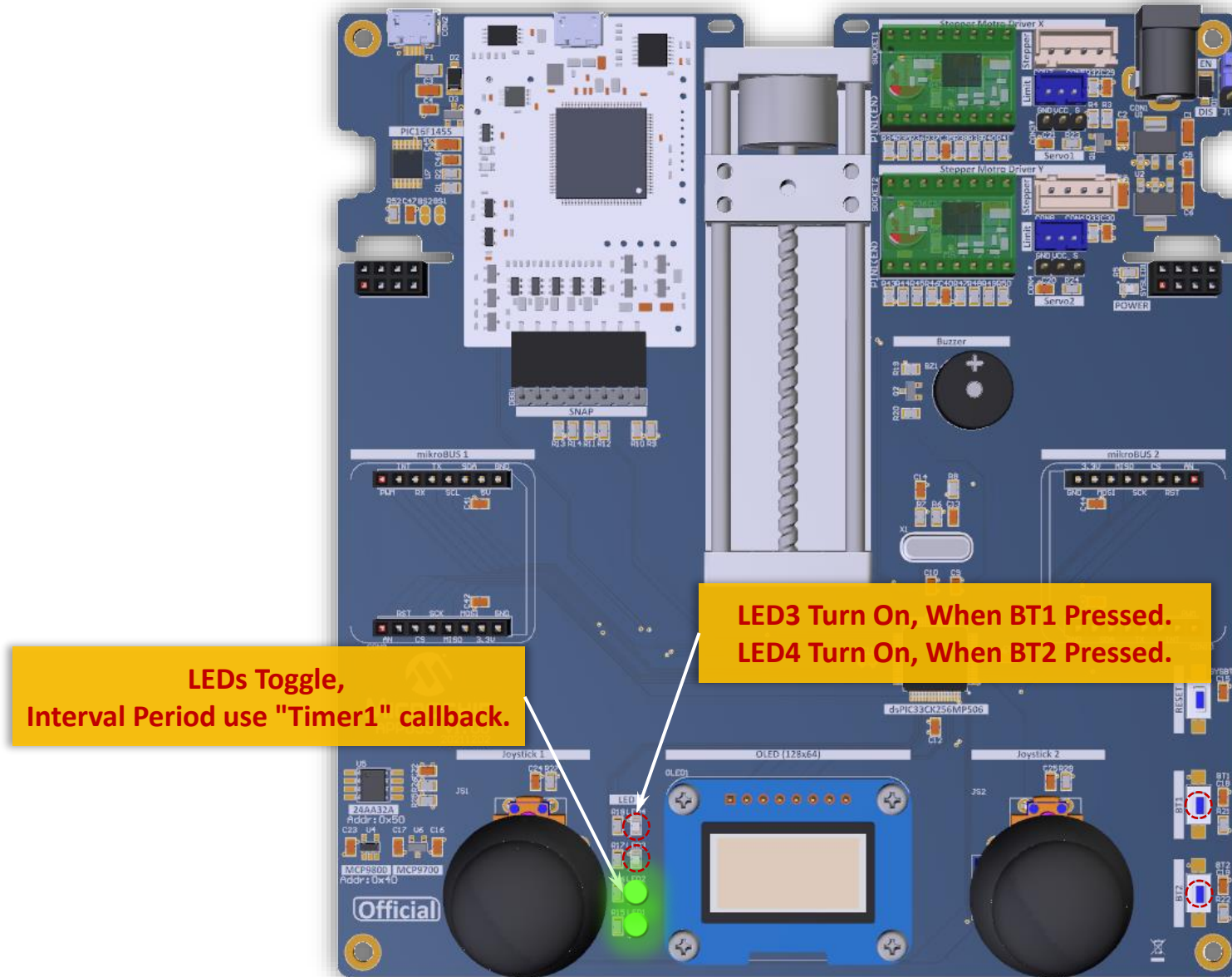
    while (1)
    {
        // Add your application code
        if(BT1_GetValue())
            LED3_SetLow();
        else
            LED3_SetHigh();

        if(BT2_GetValue())
            LED4_SetLow();
        else
            LED4_SetHigh();
    }
    return 1;
}

void TMR1_Callback(void)
{
    LED1_Toggle();
    LED2_Toggle();
}
```

Lab4 Timer1 Interrupt

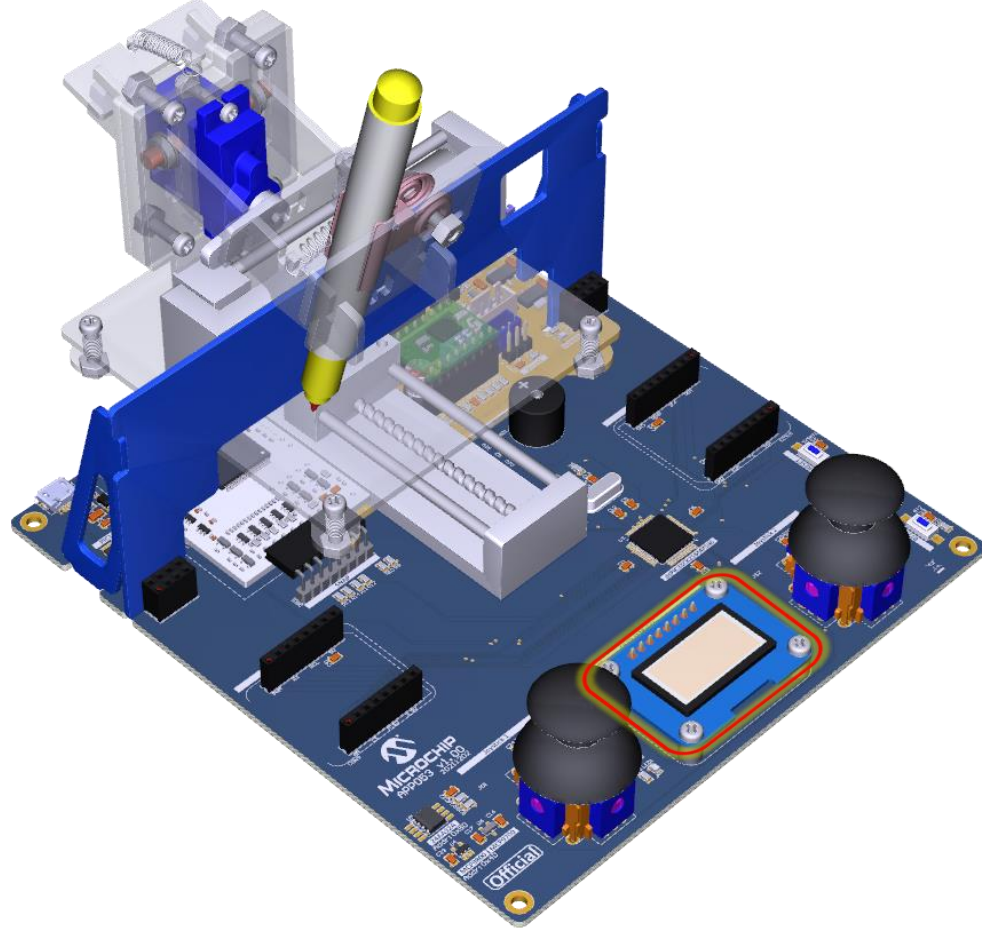
Result



OLED Application PPS & SPI Architecture

OLED Graphic Display

- **OLED (Organic Light-Emitting Diode) with controller**
 - 128 x 64 Dot matrix, Monochrome, **3 Wire SPI** with **3 External control pins**.
- **Controller : SSD1306**
 - 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller and SPI communication interface.

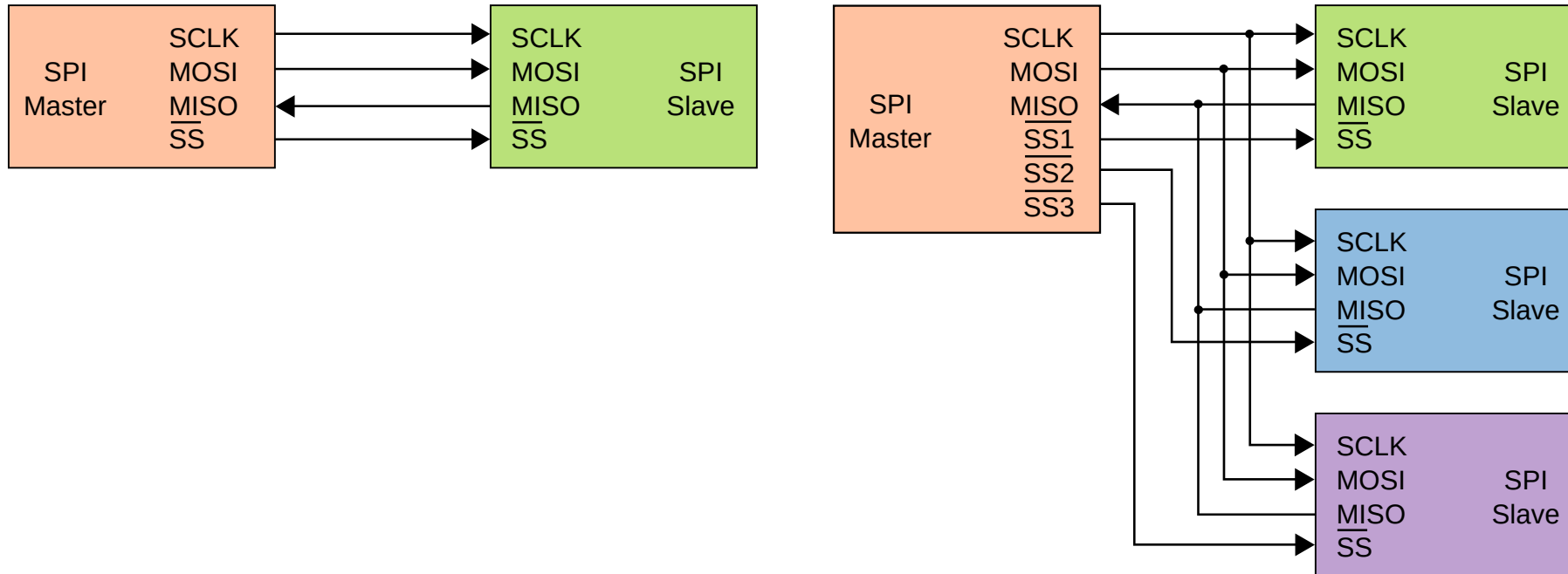


Reources Requirement for OLED

- **Peripherals : SPI**
 - **3 Wire SPI, Mode 0, SCLK = 8 MHz**
 - Chip Select
 - SCK
 - SDO(MOSI)
- **Peripherals : GPIO**
 - **CS : Chip Select (Enable Chip & Initial Communication)**
 - High : Disable.
 - Low : Enable.
 - **RS : Data / Command select**
 - High : Data.
 - Low : Command.
 - **RESET : Module Reset**
 - High : Normal.
 - Low : Hold Reset.
- **Libraries: Delay**
 - **Timing Control (Block)**

What's SPI

- SPI, Serial Peripheral Interface
- It's a serial communication interface.
Synchronous, inside-board level, short distance communication.

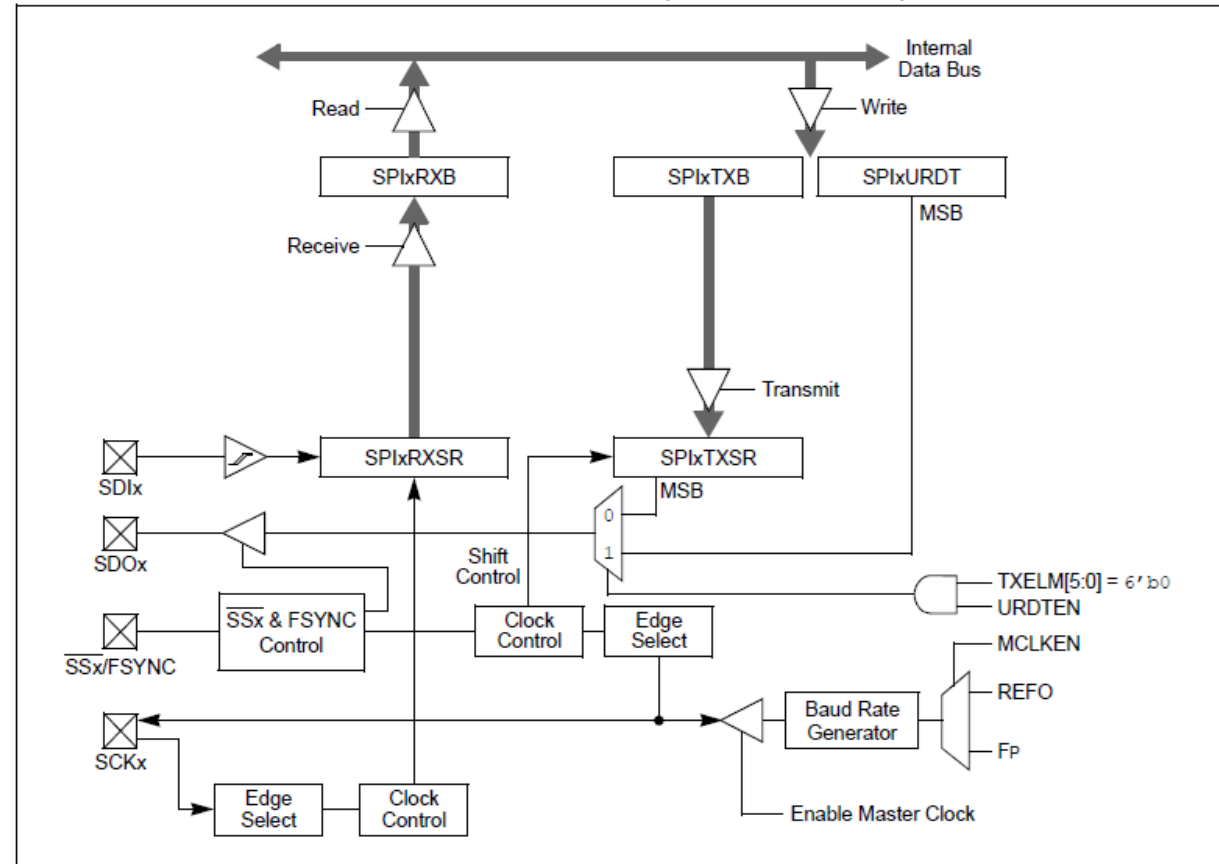


<https://zh.wikipedia.org/wiki/%E5%BA%8F%E5%88%97%E5%91%A8%E9%82%8A%E4%BB%8B%E9%9D%A2>

dsPIC33's SPI Module

- dsPIC33CK256MP506 Provide 3 SPI module, support limited pin remapping.
- Pin Definition :
 - **SDO (MOSI)** :
 - Data Out, dsPIC33 to Device.
 - **SDI (MISO)** :
 - Data In, Device to dsPIC33.
 - **SCK (SCLK)** :
 - Clock Out, dsPIC33 to Device.
 - **SS (FSYNC)** :
 - Active-Low Slave Select or Frame Synchronization I/O Pulse

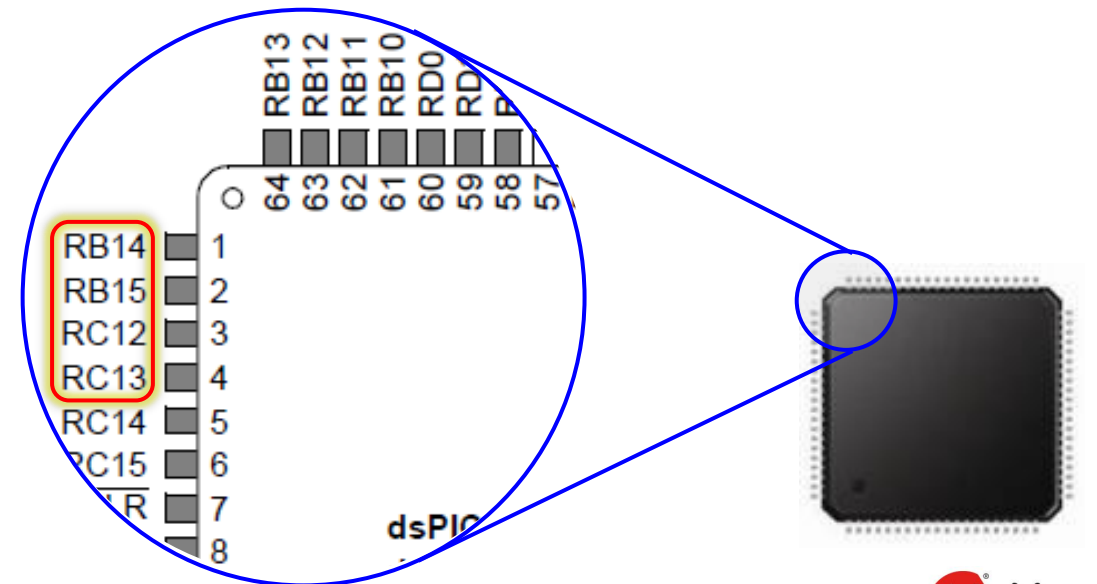
FIGURE 17-1: SPIx MODULE BLOCK DIAGRAM (STANDARD MODE)



What's PPS

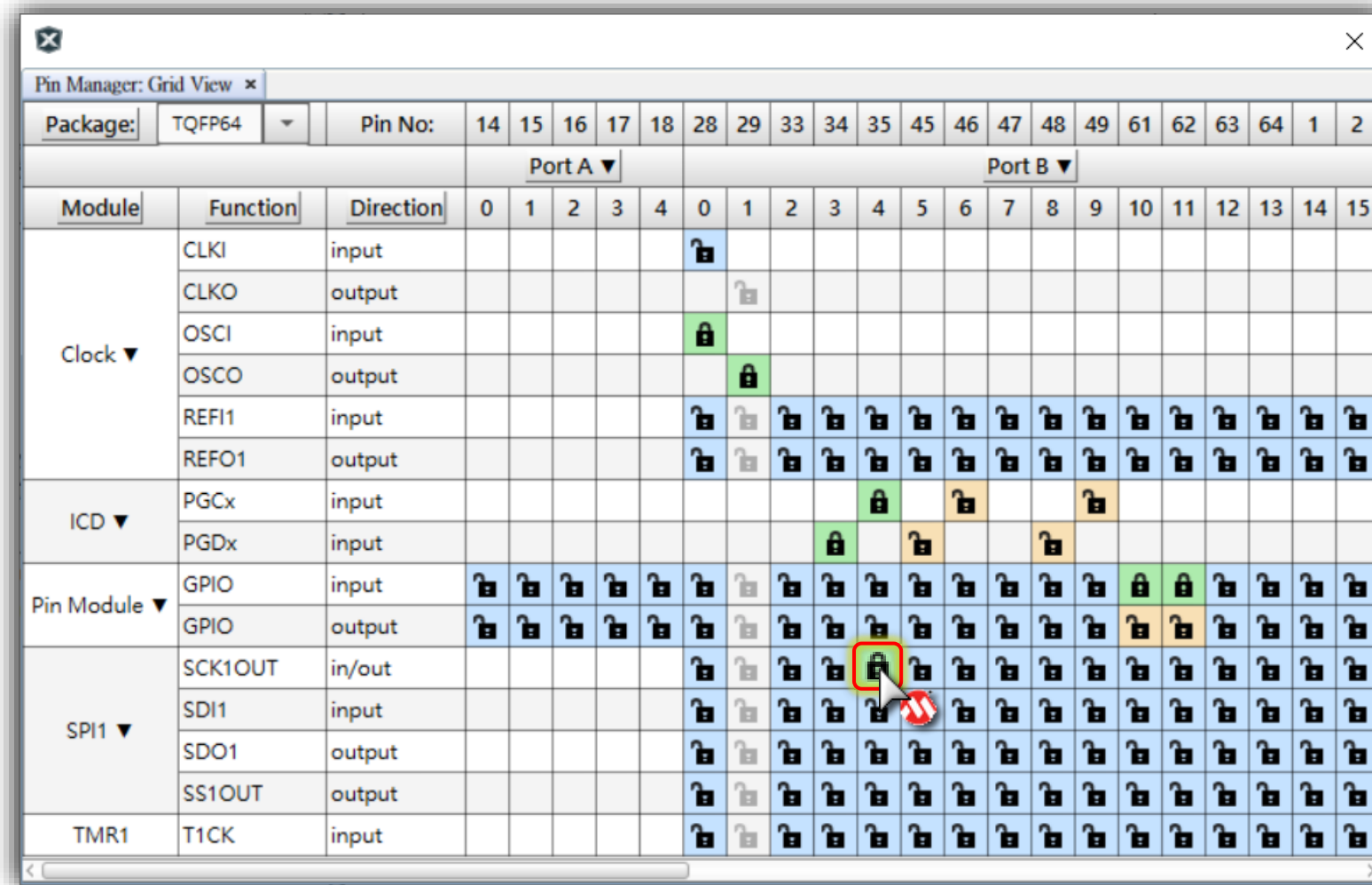
- PPS, Peripheral Pin Select
- Provides a flexible peripheral pins remapping mechanism users can better tailor the microcontroller to their entire application.
- PPS pin definition
 - **RP n** : Input & Output
 - **RPIn** : Input Only
- The peripherals managed by the peripheral pin select are all **digital only peripherals**.

Pin #	Function
1	RP46 PWM1H/PMD5/RB14
2	RP47 PWM1L/PMD6/RB15
3	RP60 PWM8H/PMD7/RC12
4	RP61 PWM8L/PMA5/RC13



PPS Assign Pin at MCC

- Just one Click on that pin you want to config in :
 - Pin Manager : Grid View



Pin Manager: Grid View

Package: TQFP64 Pin No: 14 15 16 17 18 28 29 33 34 35 45 46 47 48 49 61 62 63 64 1 2

Port A Port B

Module	Function	Direction	0	1	2	3	4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Clock ▼	CLKI	input						🔒															
	CLKO	output							🔒														
	OSCI	input						🔒															
	OSCO	output							🔒														
	REFI1	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	REFO1	output						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
ICD ▼	PGCx	input										🔒		🔒			🔒						
	PGDx	input									🔒		🔒			🔒							
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
SPI1 ▼	SCK1OUT	in/out						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	SDI1	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	SDO1	output						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
	SS1OUT	output						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒
TMR1	T1CK	input						🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒

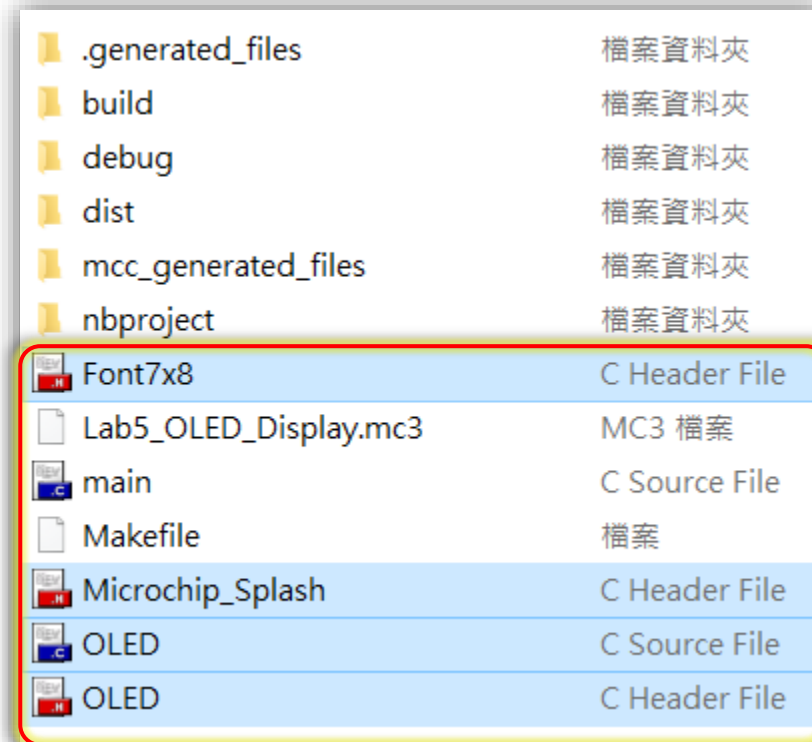
Lab Preparation

Add OLED Libraries

Lab Preparation

Step 1

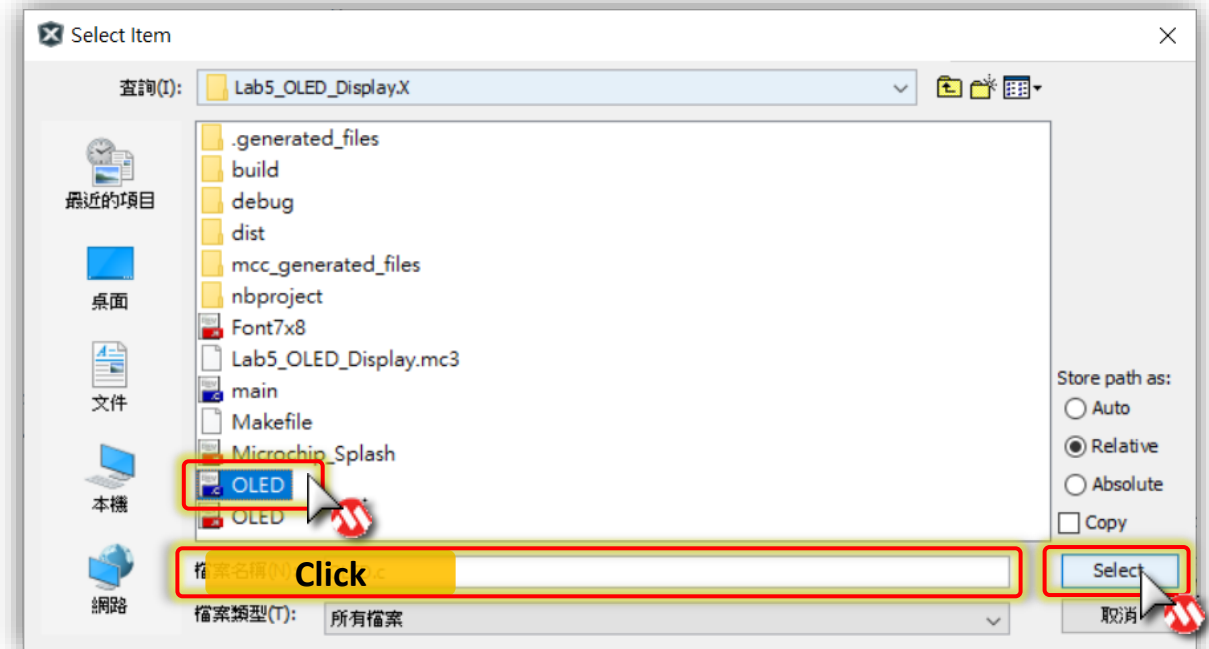
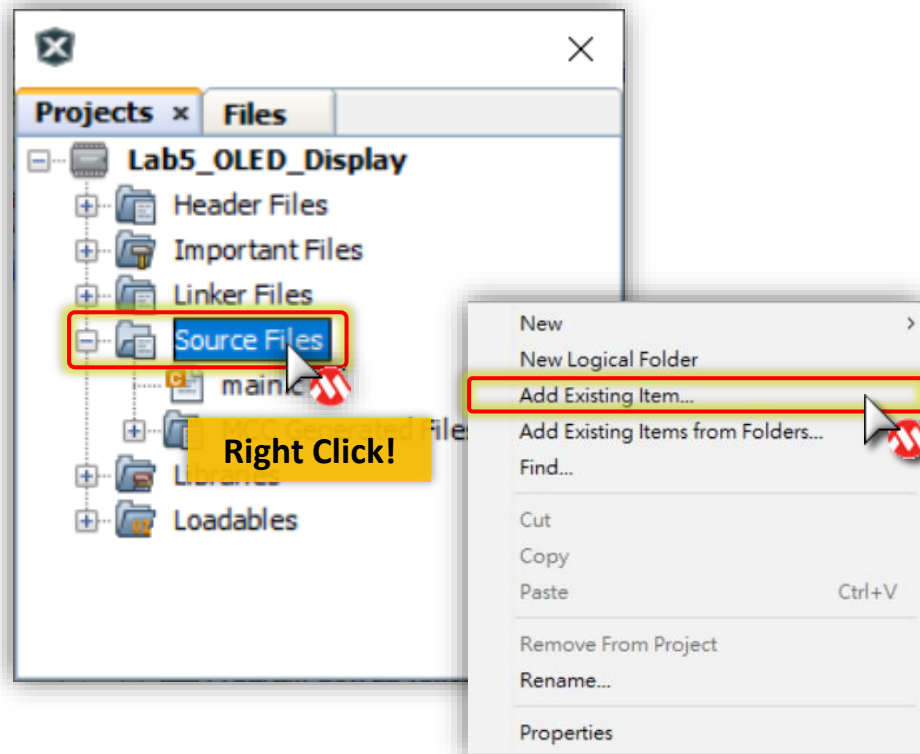
- Copy Source Files and Header Files paste on folder. ([\Appexdix\Libraries](#) to [\Labn_xxx.X](#))
 - Source Files > "OLED.c".
 - Header Files > "OLED.h", "Font7x8", "Microchip_Splash".



Lab Preparation

Step 2

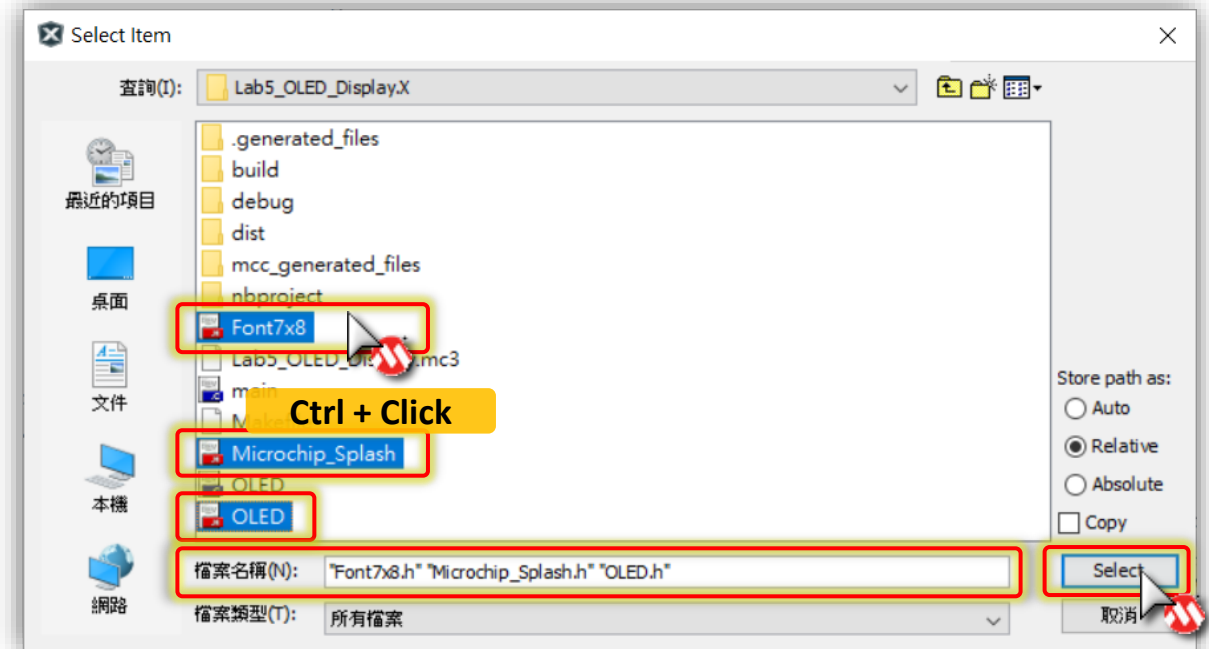
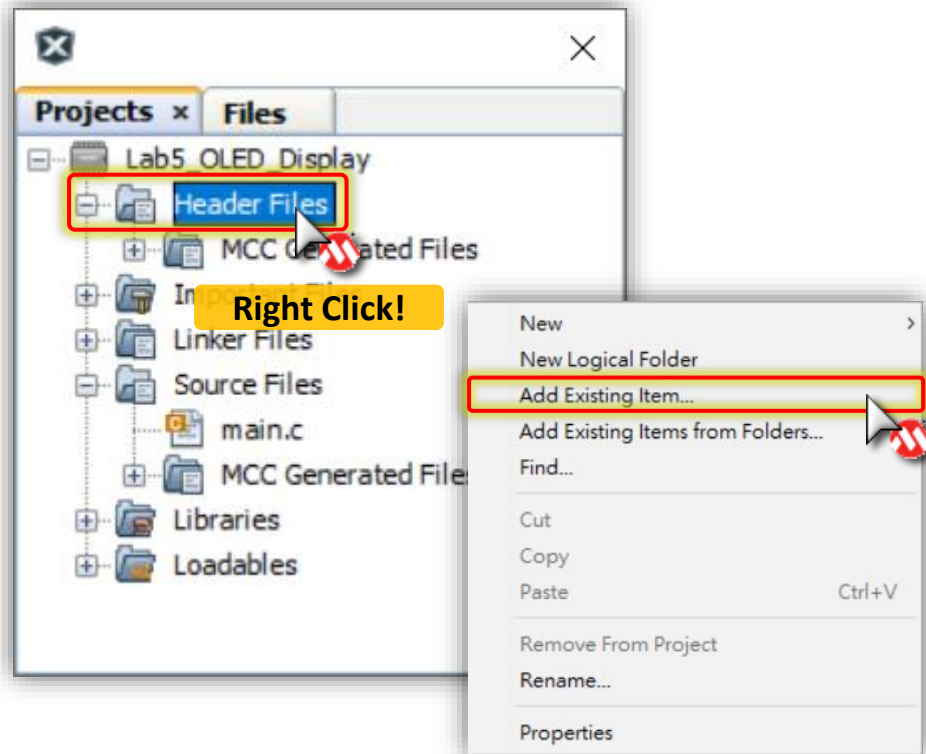
- **Add Existing Folder to Project**
 - Right click on "**Source Files**".
 - Select "**Add Existing Item**" at Projects Windows.
 - Select "**OLED.c**".



Lab Preparation

Step 3

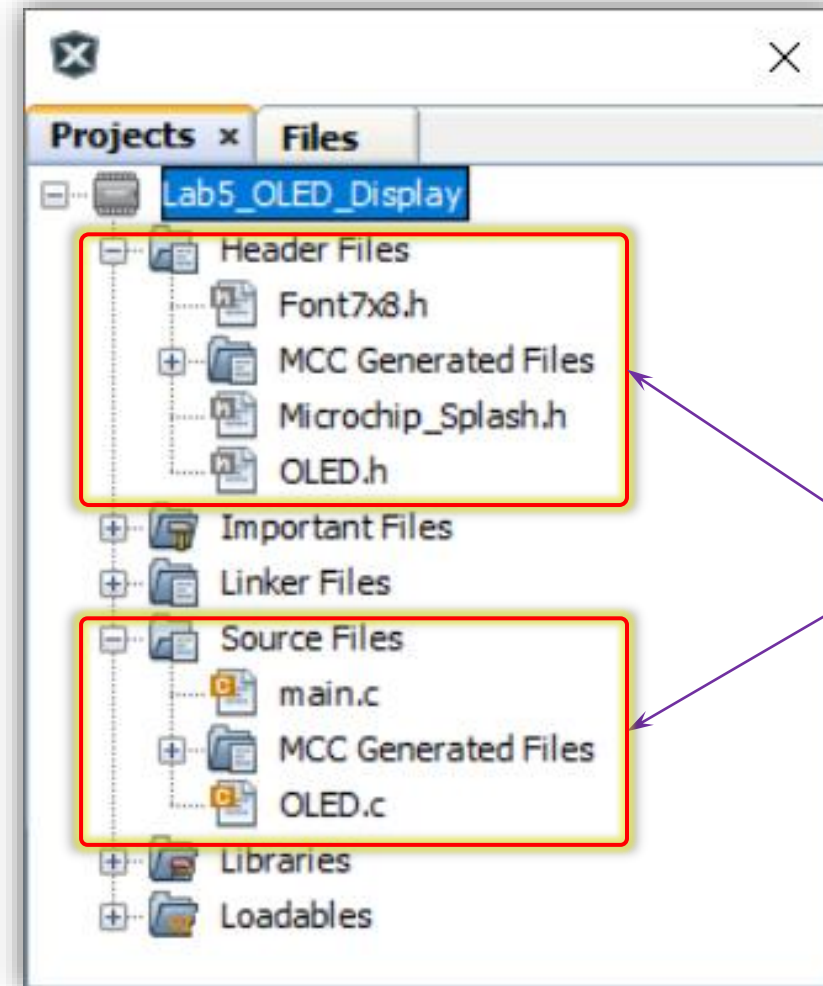
- **Add Existing Folder to Project**
 - Right click on "**Header Files**".
 - Select "**Add Existing Item**" at Projects Windows.
 - Select "**OLED.h**", "**Microchip_Splash.h**", "**Font7x8.h**".



Lab Preparation

Step 4

- Success to add libraries Files in your Project.



Libraries include :
OLED control functions
Raw Data of Fonts
Raw Data of Demo logos

What's Delay

- **MCC provide various software function and protocol stack libraries.**
e.g., Software UART, I2C, Delay, LIN Master/Slave etc..
- **Delay libraries is an easy, simple and block functions.**
Use real instruction cycle (Tcy) to generate time latency to ensure accurate delay.

Lab5 OLED Display

Lab5 OLED Display

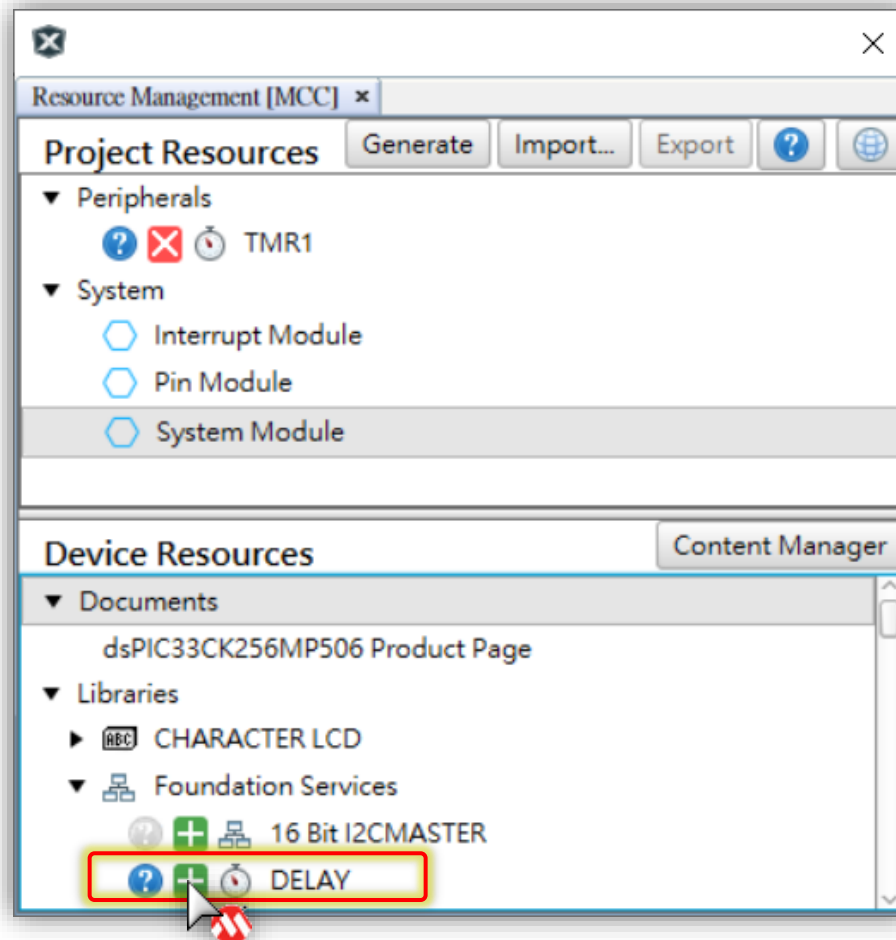
- Base on current project or **Open .\Lab5_xxx.X to do exercises.**
- Try to add OLED Library and related resources to project then initial OLED display.
- Try to display some information and bitmap logo on OLED display.

Let's go!

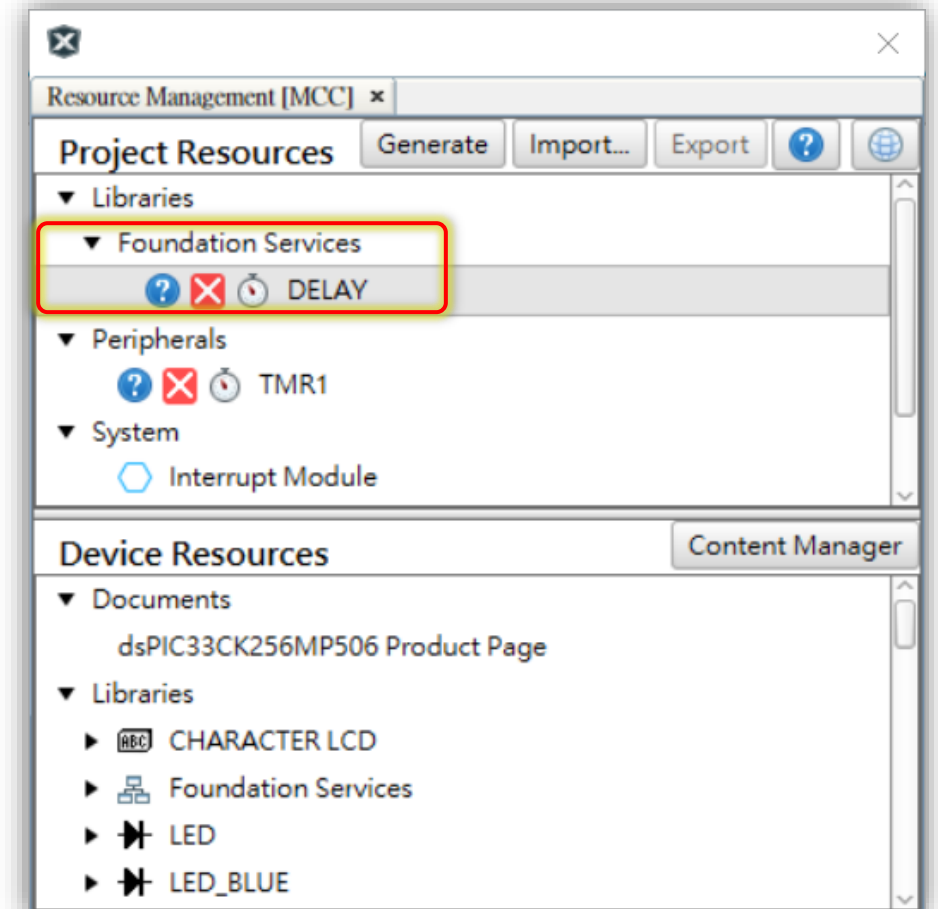
Lab5 OLED Display

Step 1

- Click [+] to add **Foundation Services** > **DELAY** Module.



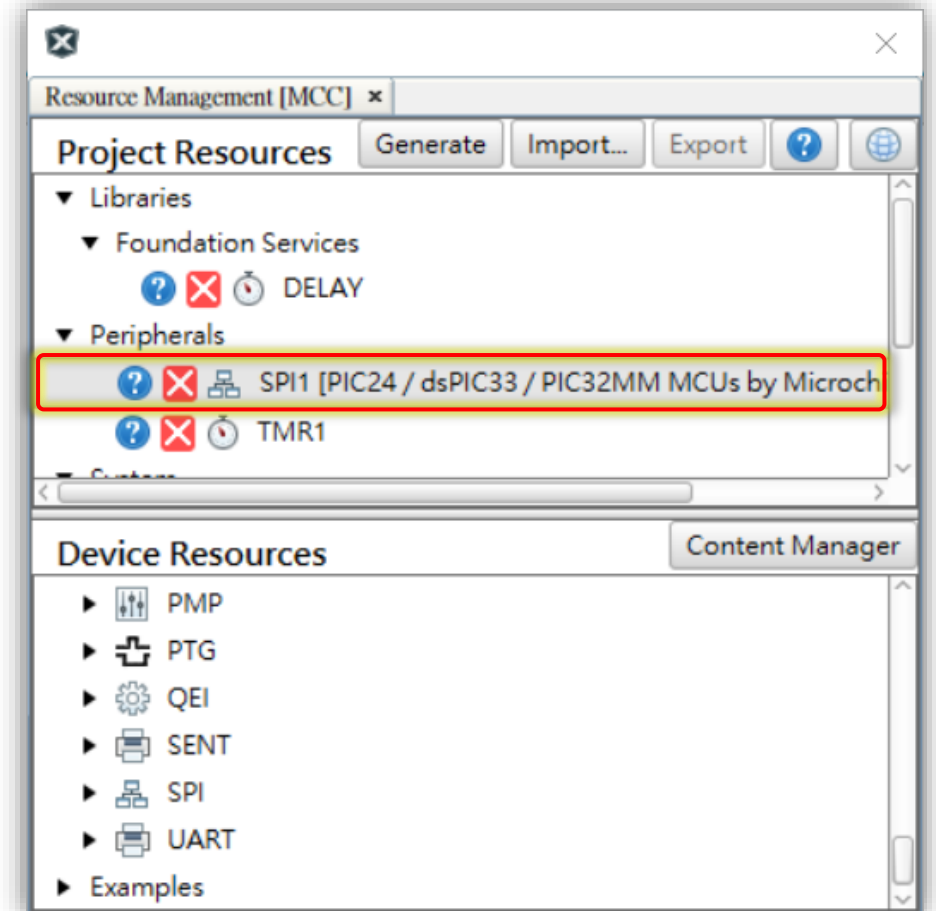
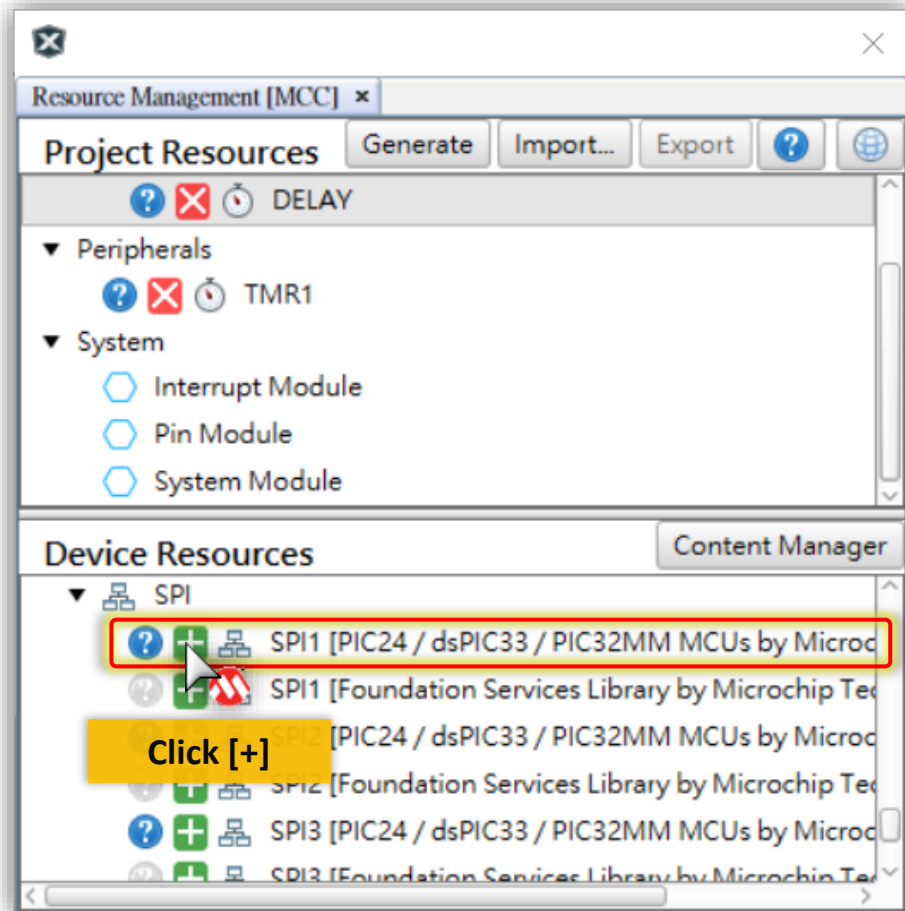
Click [+]



Lab5 OLED Display

Step 2

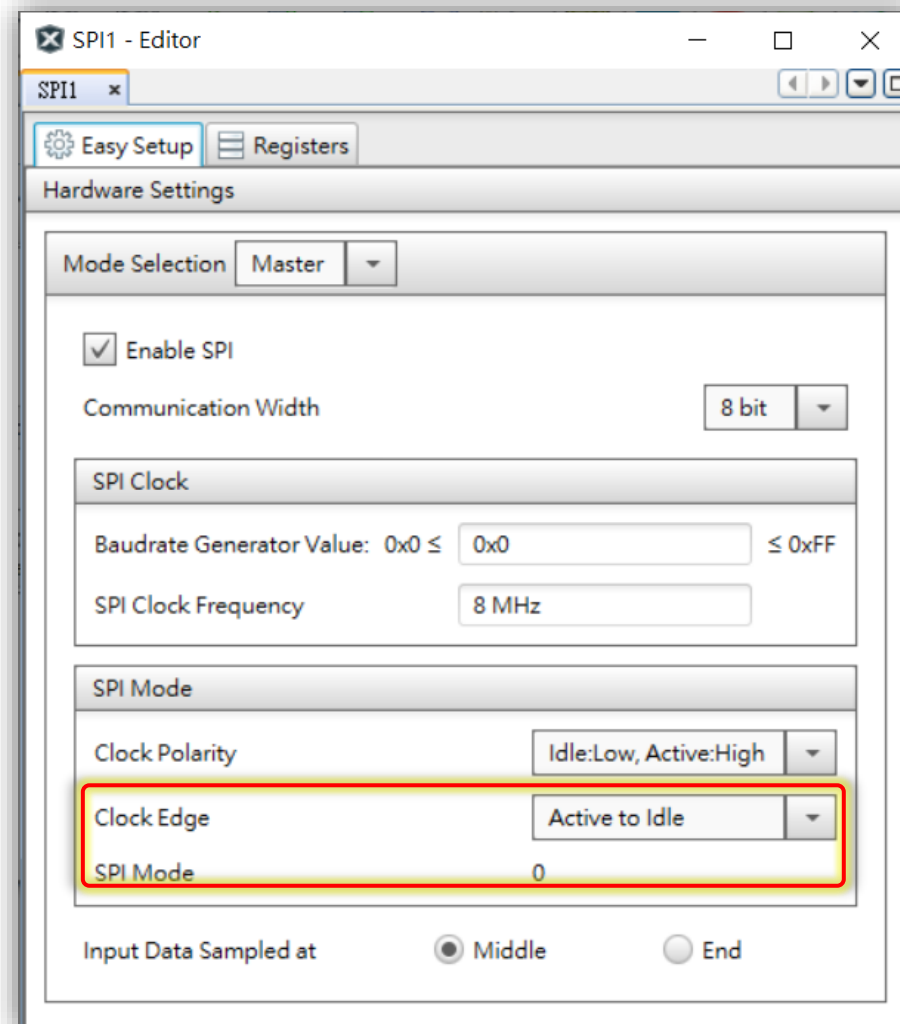
- Click [+] to add **SPI** > **SPI1** Module.



Lab5 OLED Display

Step 3

- Set Clock Edge : Idle to Active > Active to Idle.



Lab5 OLED Display

Step 5

- Set **PORTD** : Pin0 (**RD0**), Pin1 (**RD1**), Pin2 (**RD2**) to digital output mode.

✕

Pin Manager: Grid View ✕

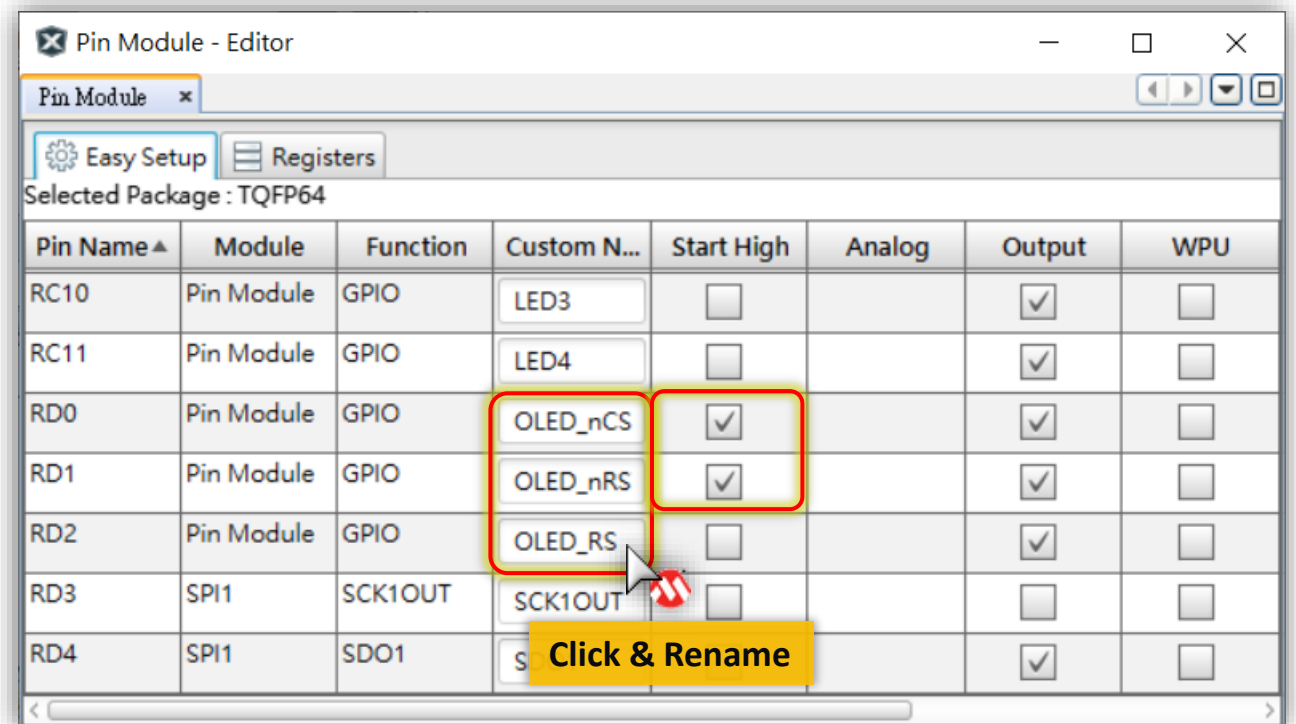
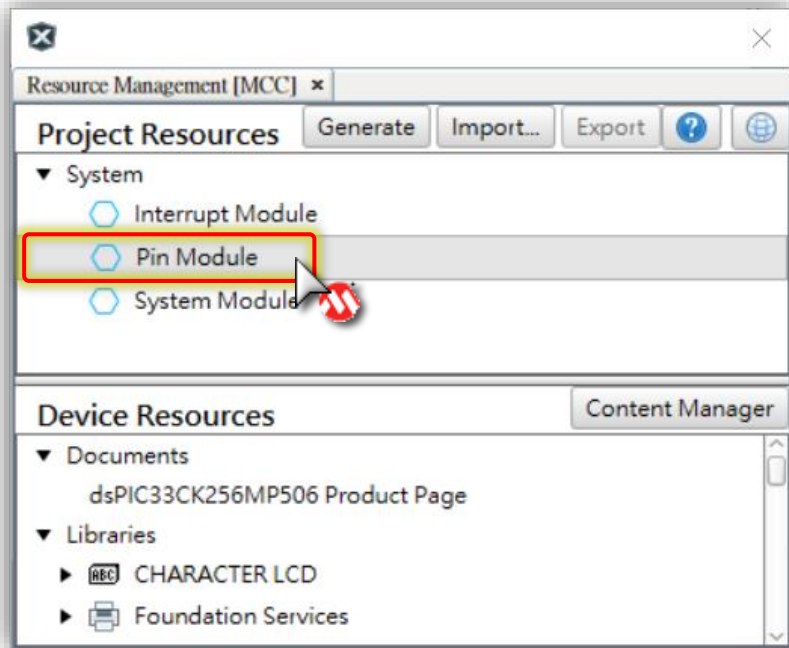
Package:	TQFP64	▼	Pin No:	13	22	23	27	50	51	24	32	36	37	52	53	3	4	5	6	60	59	58	55	54	44	43	42	39	38	31	30	21	12	11	8
			Port C ▼																Port D ▼																
Module	Function	Direction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Clock ▼	CLKI	input																																	
	CLKO	output																																	
	OSCI	input																																	
	OSCO	output																																	
	REFI1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	REFO1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
ICD ▼	PGCx	input																																	
	PGDx	input																																	
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
SPI1 ▼	SCK1OUT	in/out	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	SDI1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	SDO1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
	SS1OUT	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
TMR1	T1CK	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	

<

Lab5 OLED Display

Step 6

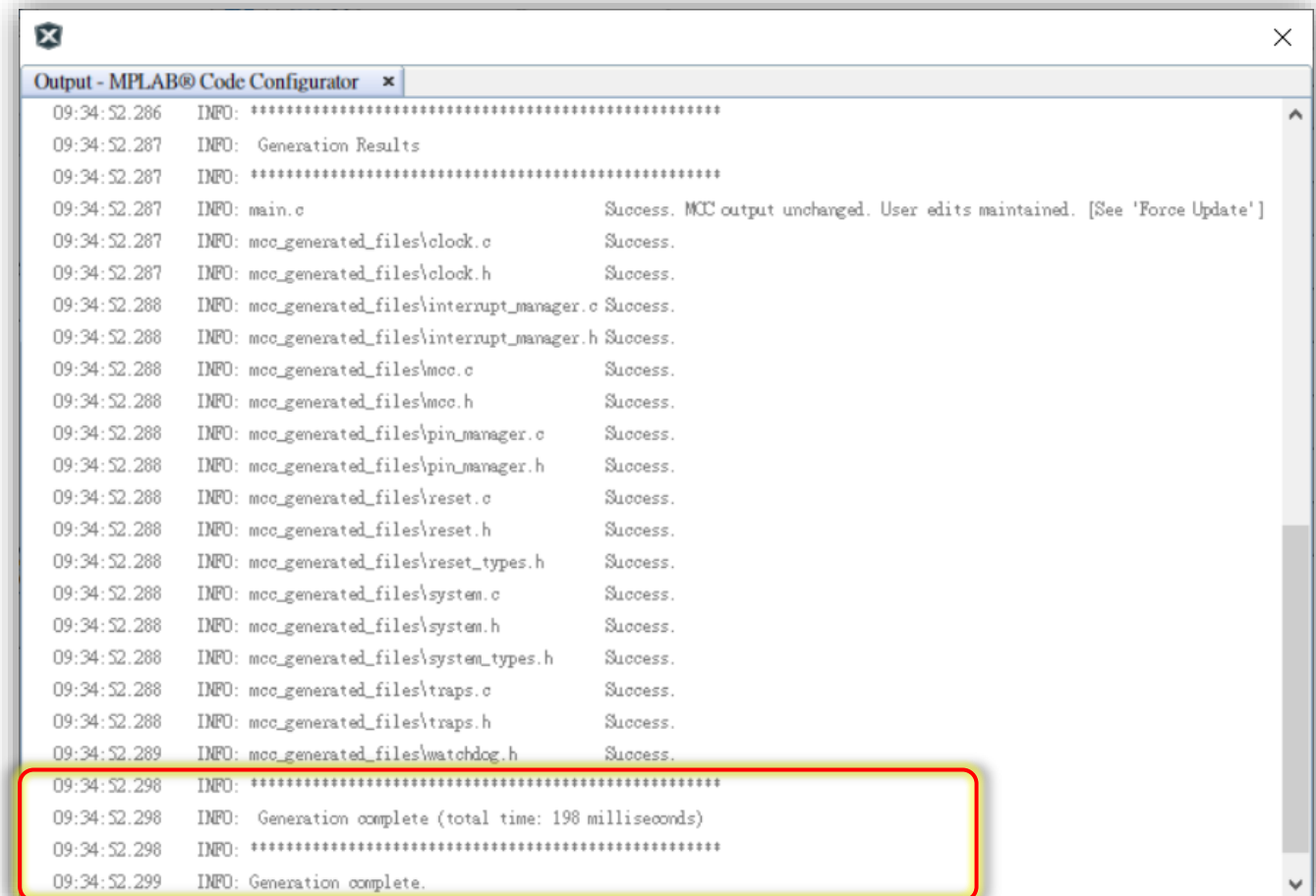
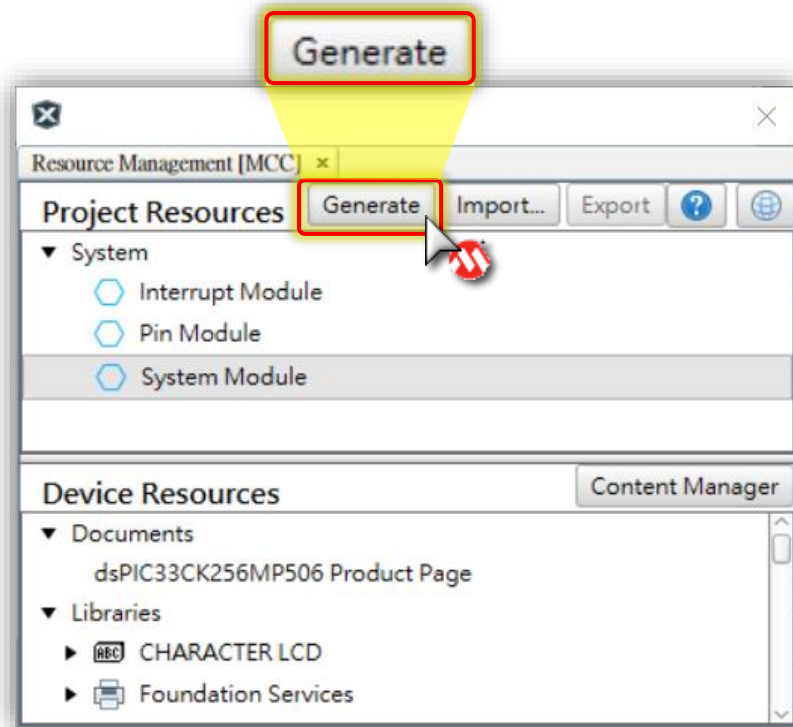
- Set Alias for RD0 、 RD1 、 RD2.
 - Pin Module : RD0 Custom Name > OLED_nCS , Start high.
 - Pin Module : RD1 Custom Name > OLED_nRST , Start high.
 - Pin Module : RD2 Custom Name > OLED_RS.



Lab5 OLED Display

Step 7

- Click "**Generate**" Button to generate your code.



OLED Functions of MCC

- The libraries base on MCC and ease to use. but we still need to assign related resources before use it.

- We Provide below functions for OLED:

"OLED.c", "OLED.h"

- `uint8_t OLED_Init( void );` /* Initial OLED Module */
- `void OLED_Clean( void );` /* Clear OLED */
- `void OLED_Update( void );` /* Update OLED Data */
- `void OLED_DrawLine(const uint16_t x1, const uint16_t y1, const uint16_t x2,
const uint16_t y2, uint8_t pixel);` /* Draw Pixel Line*/
- `void OLED_DrawBitmap(uint16_t x, uint16_t y, uint16_t w, uint16_t h,
const unsigned char *bitmap);` /* Draw Pixel Bitmap Picture */
- `void OLED_DrawFont( uint16_t x, uint16_t y, char chr );` /* Draw Single Character */
- `void OLED_DrawString( uint16_t x, uint16_t y, char *str );` /* Draw String */

Lab5 OLED Display

Code Example

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"
#include "mcc_generated_files/delay.h"
#include <stdio.h>
#include <string.h>
#include "OLED.h"
#include "Microchip_Splash.h"

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    OLED_Init();

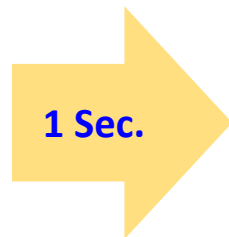
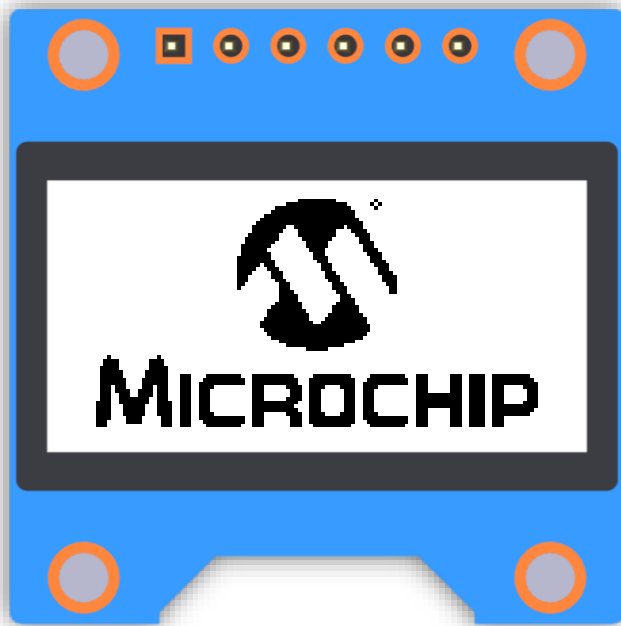
    OLED_Clean();
    OLED_DrawBitmap(0, 0, 128, 64, Microchip_Splash);
    OLED_Update();

    DELAY_milliseconds(1000);

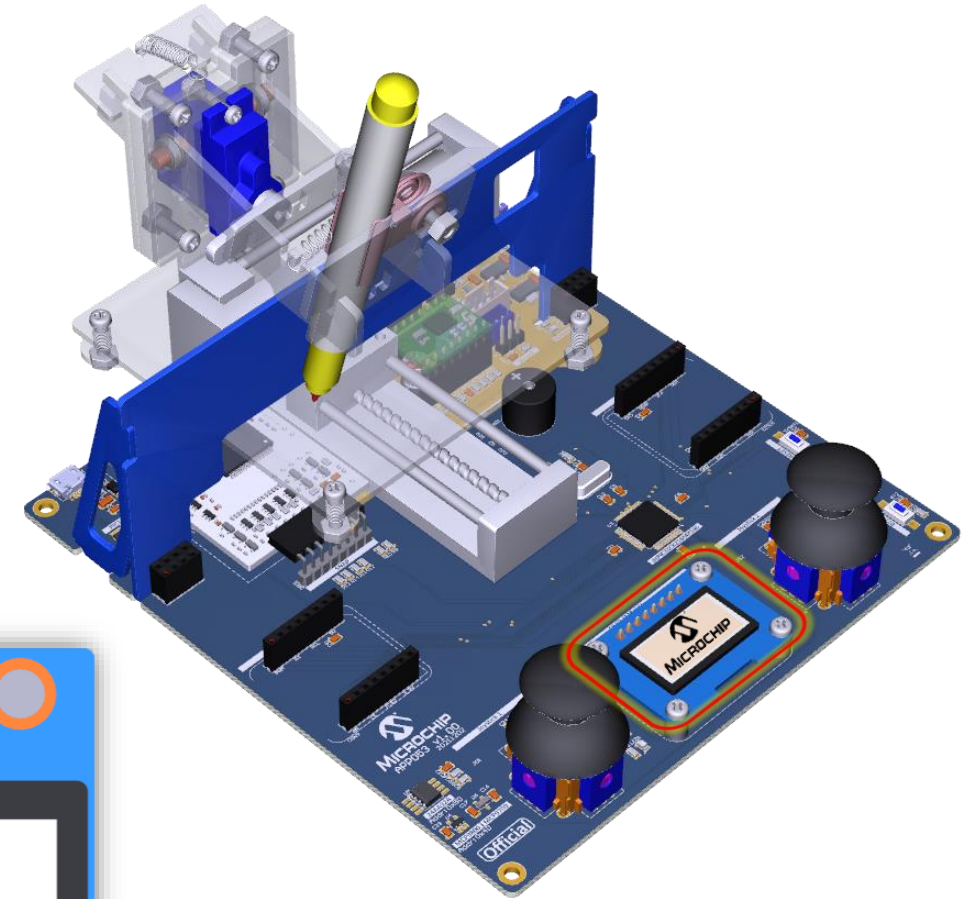
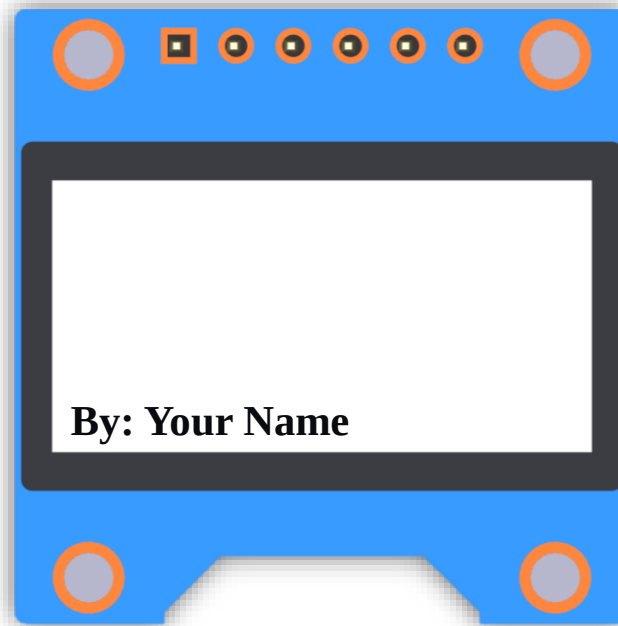
    OLED_Clean();
    OLED_DrawString(0, 56, "By:Your Name");
    OLED_Update();
    while (1)
    {
        ...
    }
    return 1;
}
```

Lab5 OLED Display

Result



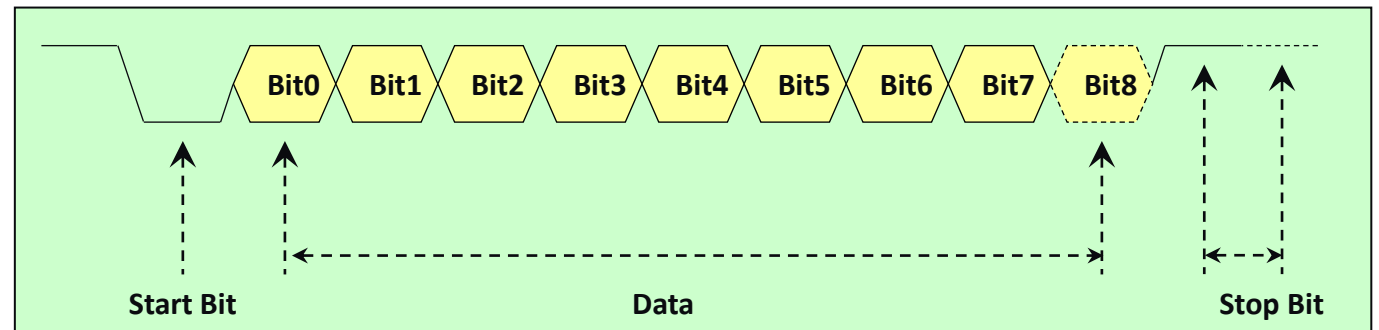
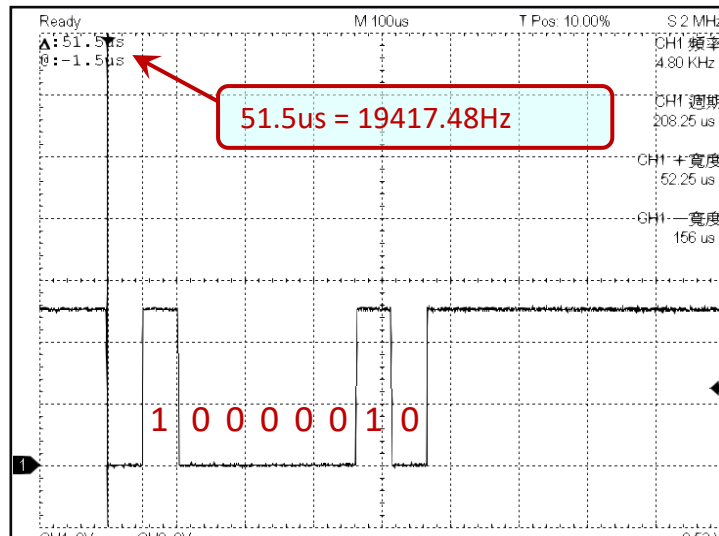
1 Sec.



UART Architecture

What's UART

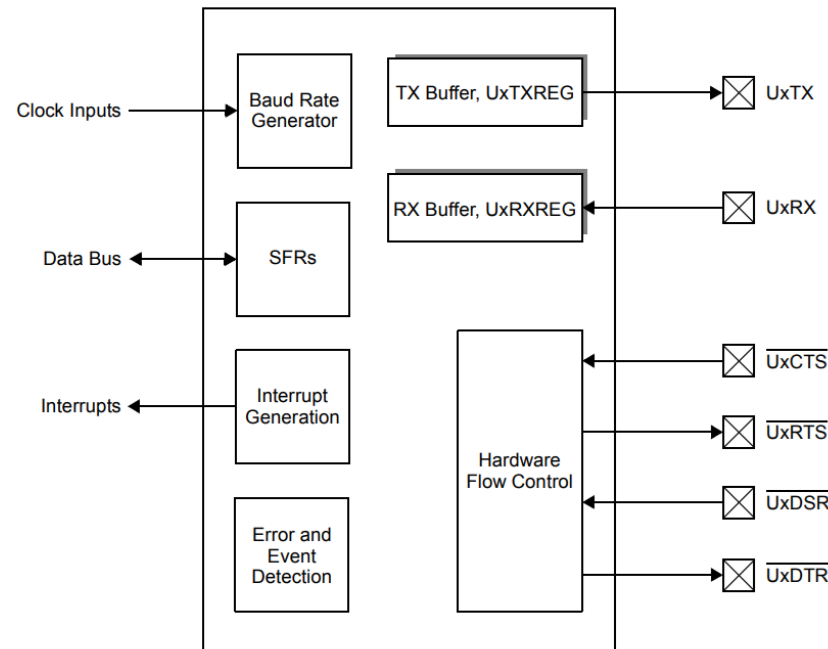
- UART : Universal Asynchronous Receiver Transmitter.
- The module data exchange through serial communication.
- The data formatting includes 1 start bit, 5 ~ 9 data bits and 1 ~ 2 stop bits.



UART Transmit Example : 'A' (0x41), 19,200 bps

dsPIC33CK's UART

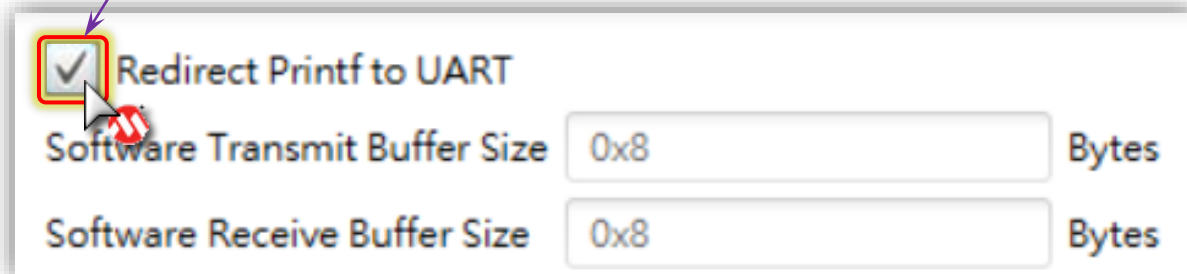
- UART module is full-duplex, asynchronous communication, Support **RS-232**, **RS-485**, **IrDA** and **LIN**.
- The module also supports **hardware flow control**, **Parity check**, **Frame Error**, **Overflow** **8 ~ 9 Data Bits**.
- **8-Deep FIFO** for **TX & RX**.
- Provide Loopback, Address Detect(RS-485), Auto Baud Detect(LIN bus).



STDOUT Redirect

- MCC provide STDOUT redirect function, It can redirect STDOUT to UART.
- You can use printf() to export data by UART, if redirect successfully.
- For Example :
 - **UART Transmit Example : printf("Hello World!!");**

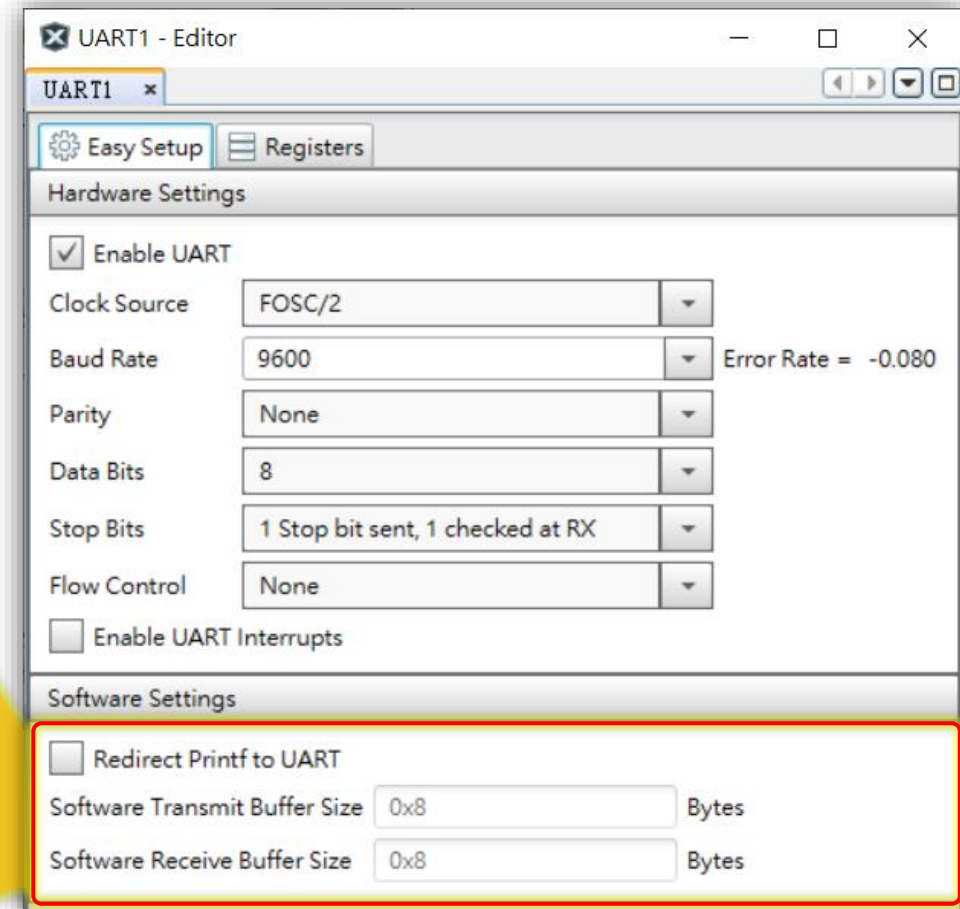
Redirect Printf to the UART



☒ Redirect Printf to UART

Software Transmit Buffer Size Bytes

Software Receive Buffer Size Bytes



UART1 - Editor

UART1 x

Easy Setup Registers

Hardware Settings

☒ Enable UART

Clock Source

Baud Rate Error Rate = -0.080

Parity

Data Bits

Stop Bits

Flow Control

☐ Enable UART Interrupts

Software Settings

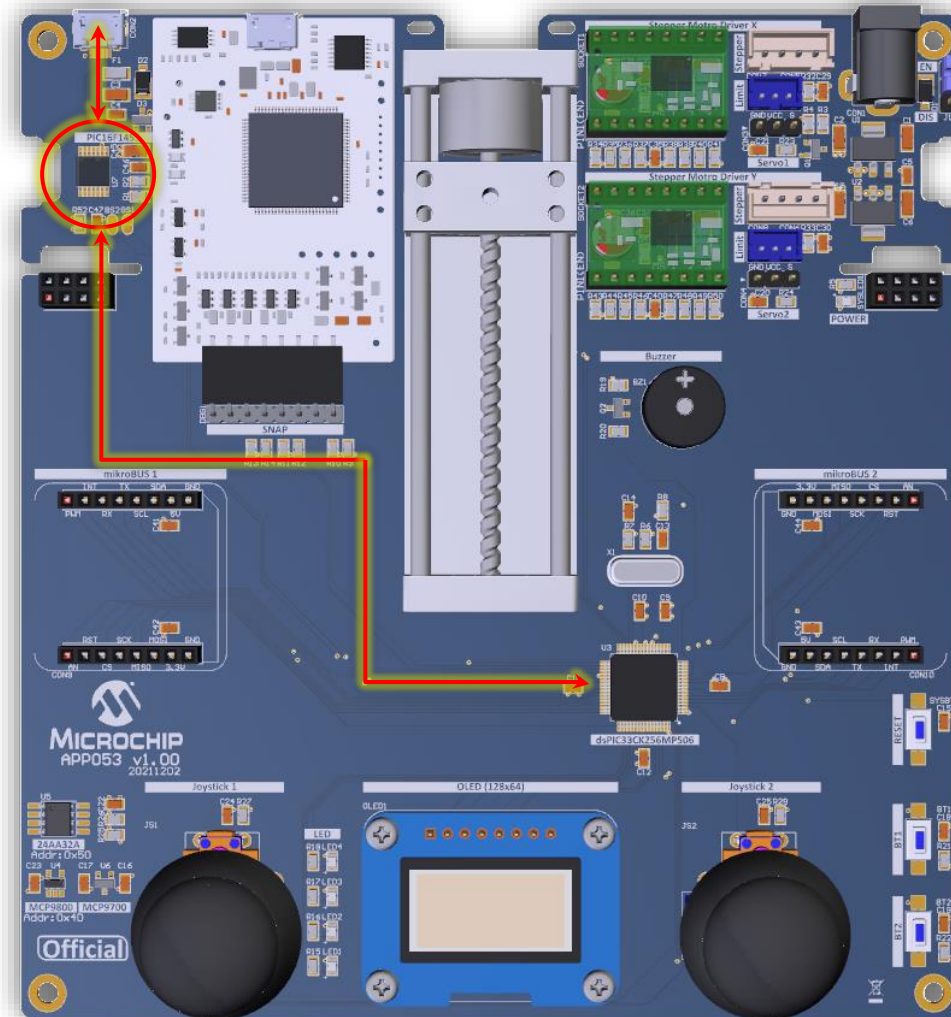
☐ Redirect Printf to UART

Software Transmit Buffer Size Bytes

Software Receive Buffer Size Bytes

USB CDC Emulator

- The "**PIC16F1455**" already preload USB Serial Emulator firmware (USB Virtual COM Port).



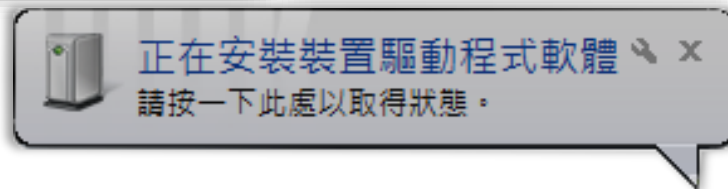
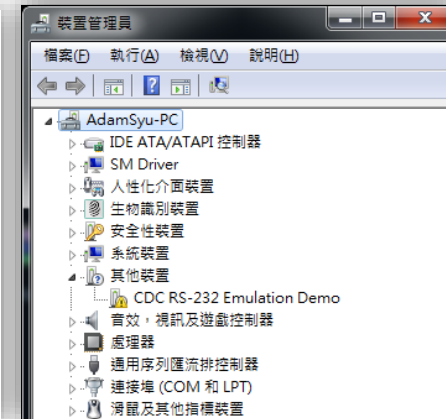
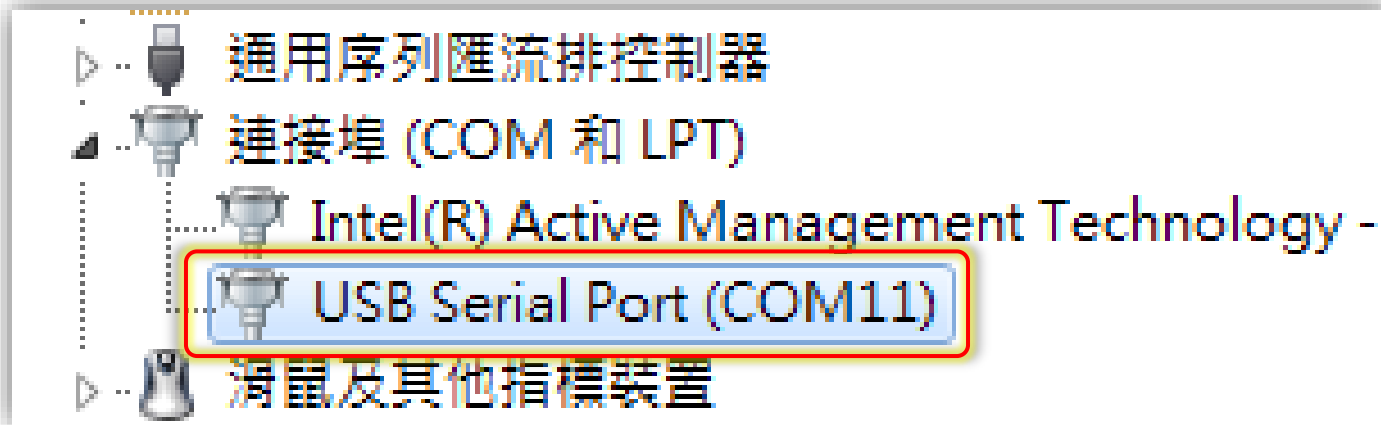
Lab Preparation

PC Terminal Software Setting

Lab Preparation

Step 1

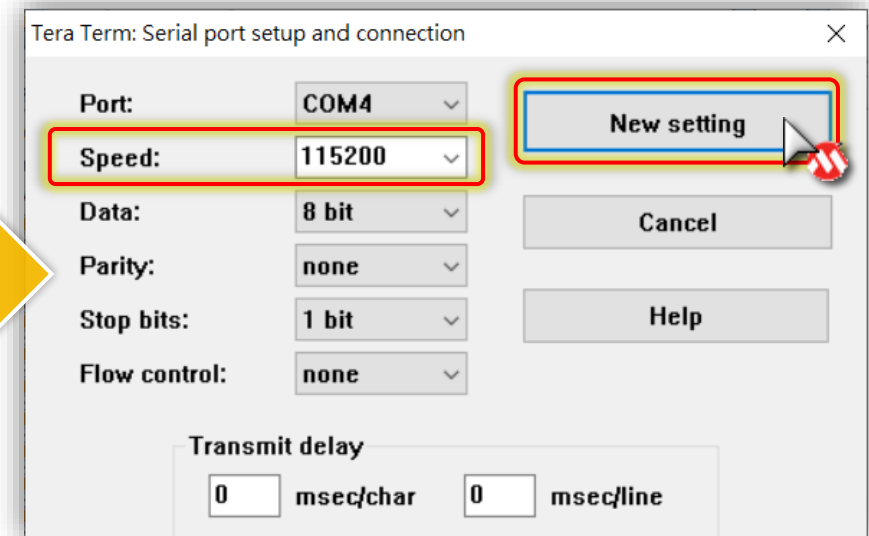
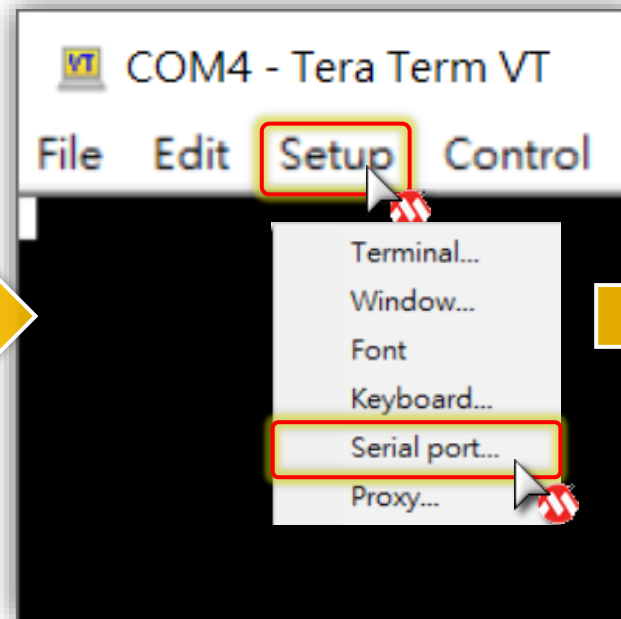
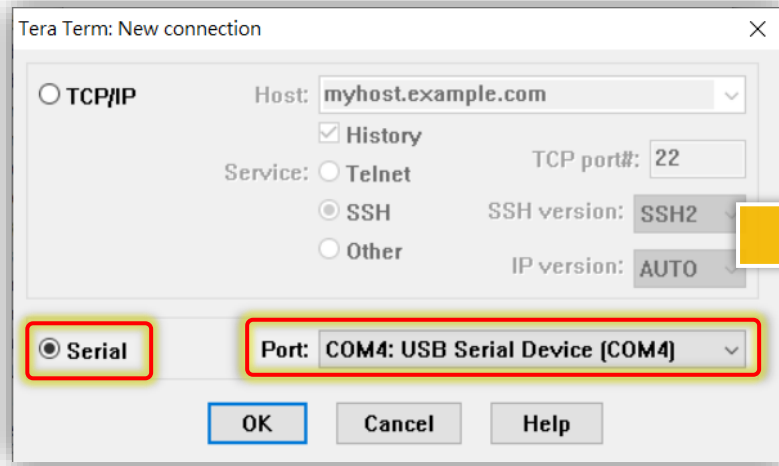
- Please confirm Windows Device Manager on your PC.
- For Window 7, you might need install driver before Labs.
 - Driver at : [\Appendix\Serial Emulator Driver\mchpcdc.inf](#).
- You will see a **USB Serial Port (COMx)** at Device Manager.



Lab Preparation

Step 2

- Find TeraTerm from your desktop or install it from.
 - <http://ttssh2.sourceforge.jp/index.html.en>.
- Select : Serial > Port : COMx : USB Serial Port (COMx).
- Check : Setup > Serial Port > **COMx**, **115200**, **8 bit**, **none**, **1 bit**, **none**.



Lab6 UART printf

Lab6 UART printf

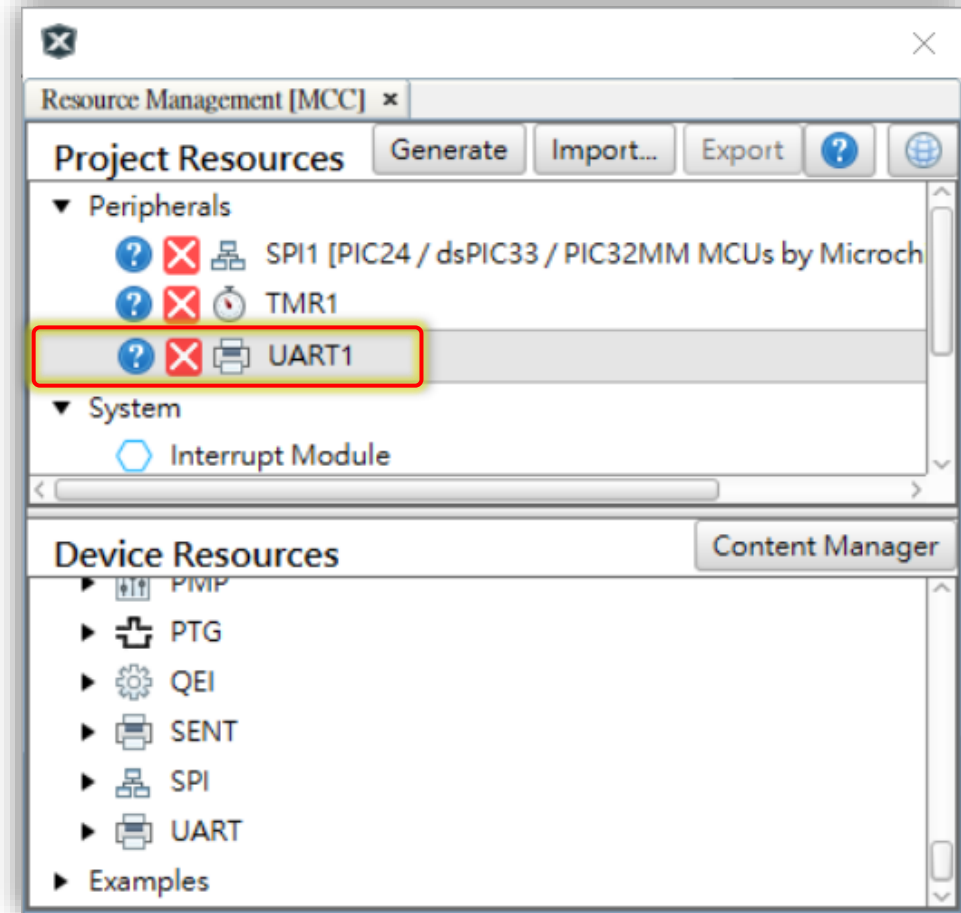
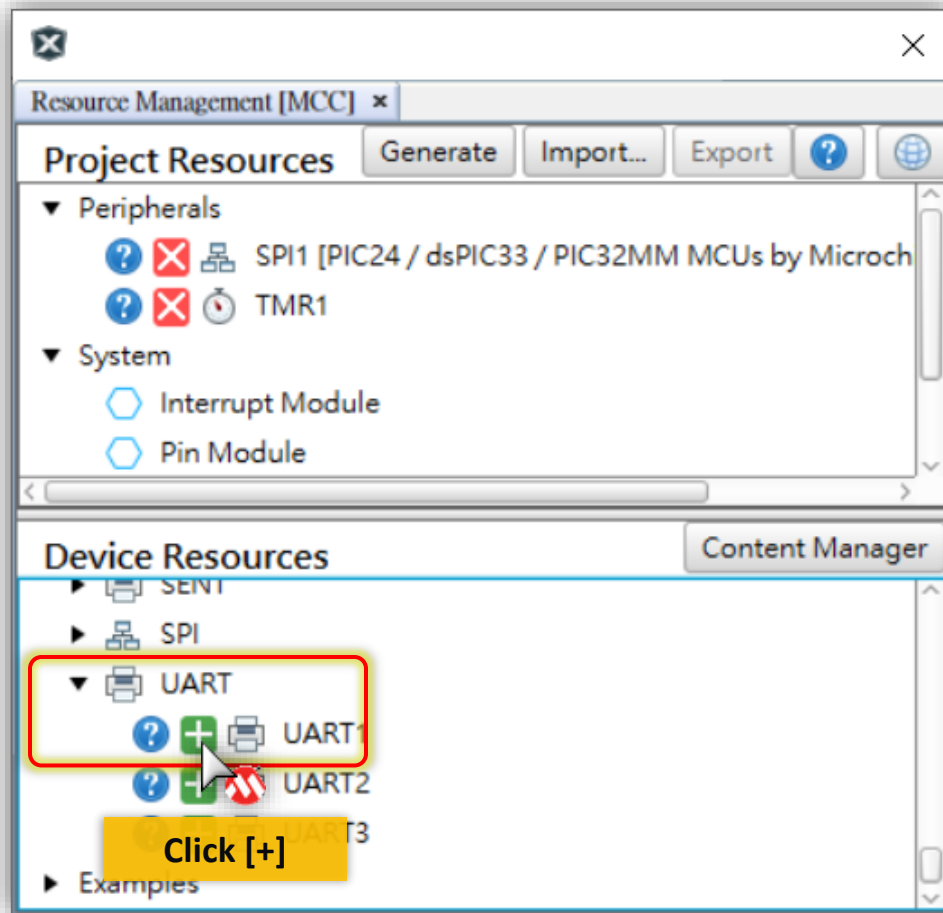
- Base on current project or Open `.\Lab6_xxx.X` to do exercises.
- Try to add UART1 module then output "Hello World!!".
- UART1 set to **115200**, **N**, **8**, **1**, **No flow control**.
- UART has connected by PCB layout.
 - dsPIC33CK (**UART1-U1TX**, **RD8**) <-> PIC16F1455 (**RX**, **RC5**).
 - dsPIC33CK (**UART1-U1RX**, **RD9**) <-> PIC16F1455 (**TX**, **RC4**).

Let's go!

Lab6 UART printf

Step 1

- Click [+] to add **UART** > **UART1** Module.



Lab6 UART printf

Step 2

- Set Baud Rate : 9600 > 115200.

UART1 - Editor

UART1

Easy Setup Registers

Hardware Settings

☒ Enable UART

Clock Source FOSC/2

Baud Rate 115200 Error Rate = -0.794

Parity None

Data Bits 8

Stop Bits 1 Stop bit sent, 1 checked at RX

Flow Control None

☐ Enable UART Interrupts

Software Settings

☐ Redirect Printf to UART

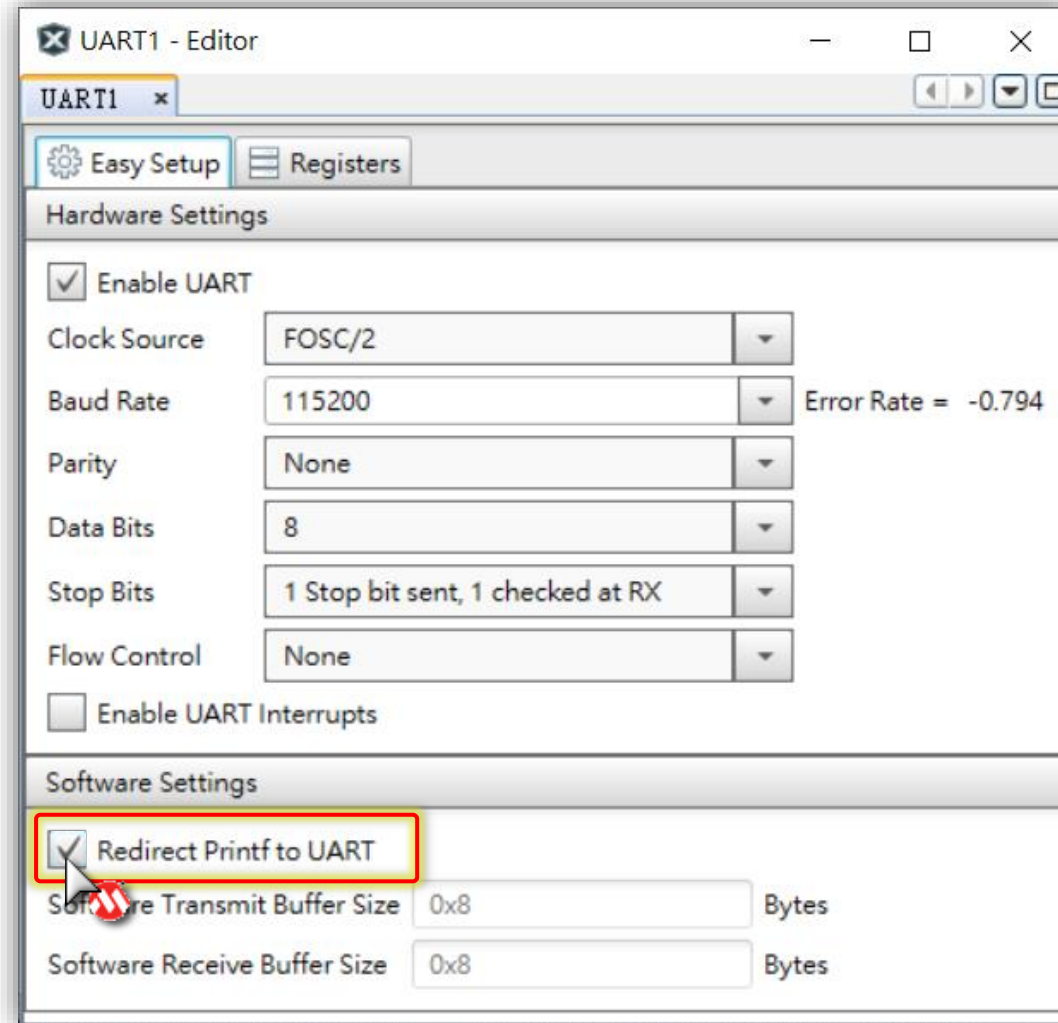
Software Transmit Buffer Size 0x8 Bytes

Software Receive Buffer Size 0x8 Bytes

Lab6 UART printf

Step 3

- Set **Enable Redirect Printf to UART**.



Lab6 UART printf

Step 4

- Assign UART1 TX and RX Pin.
 - Pin Manager : RD8 > U1TX, RD9 > U1RX.

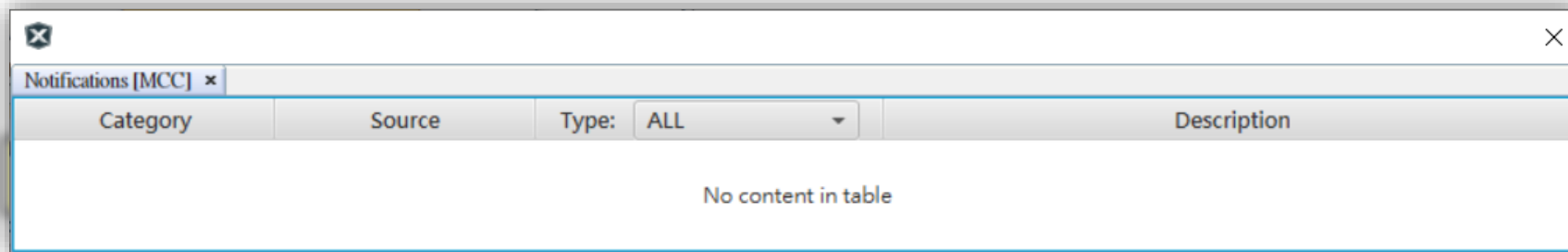
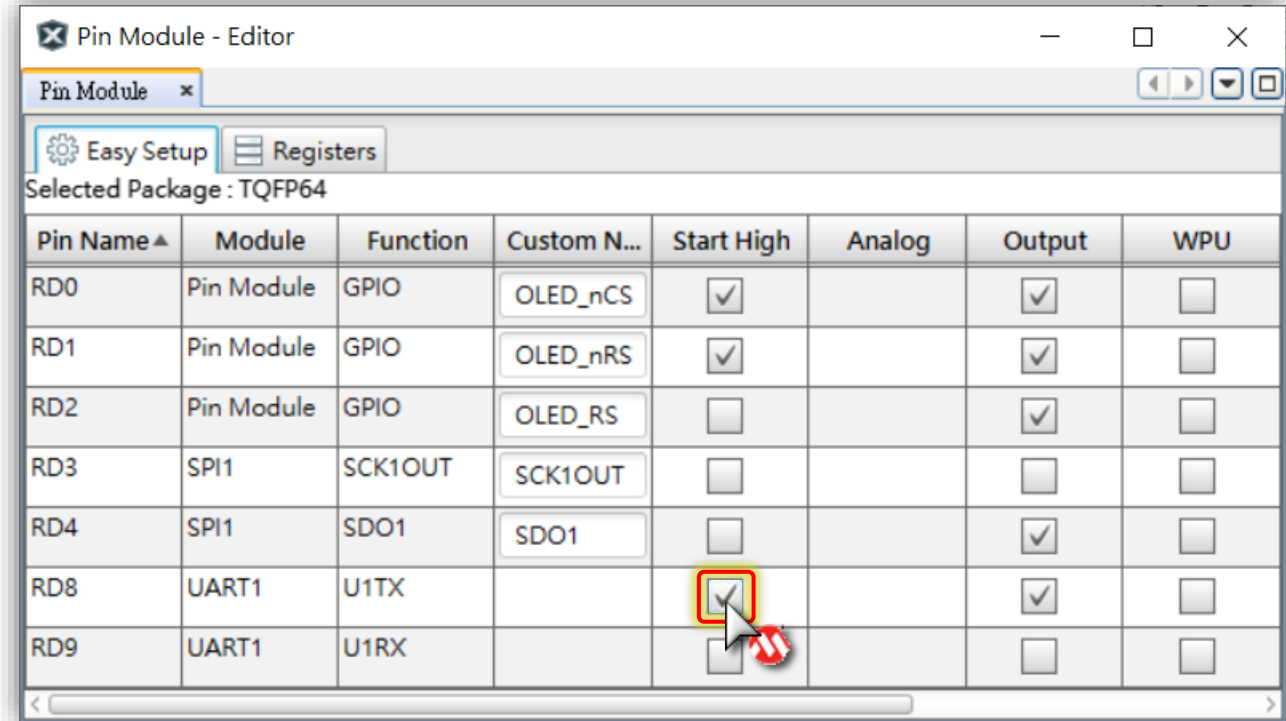
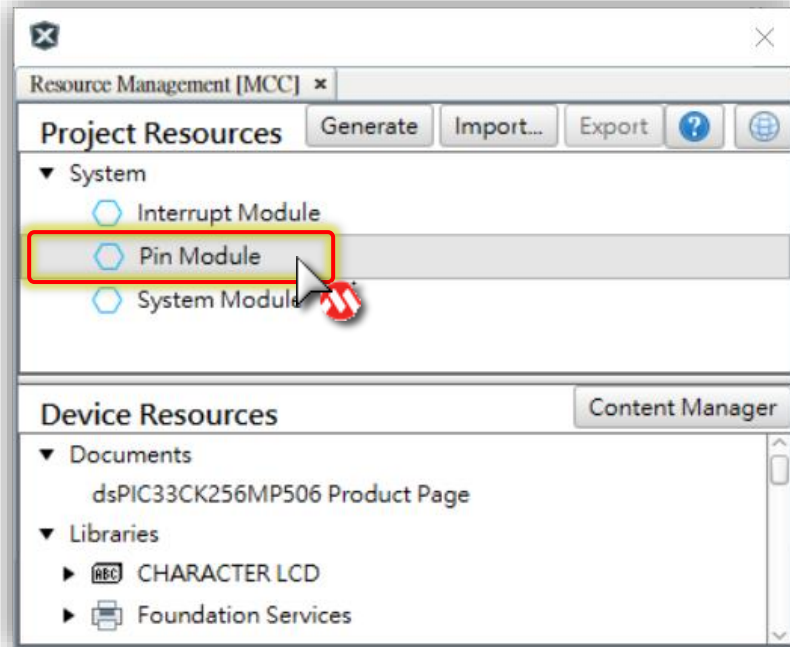
Pin Manager: Grid View

			Port C																Port D															
Module	Function	Direction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	REFO1	output																																
ICD	PGCx	input																																
	PGDx	input																																
Pin Module	GPIO	input																																
	GPIO	output																																
SPI1	SCK1OUT	in/out																																
	SDI1	input																																
	SDO1	output																																
	SS1OUT	output																																
TMR1	T1CK	input																																
UART1	U1CTS	input																																
	U1DSR	input																																
	U1DTR	output																																
	U1RTS	output																																
	U1RX	input																																
	U1TX	output																																

Lab6 UART printf

Step 5

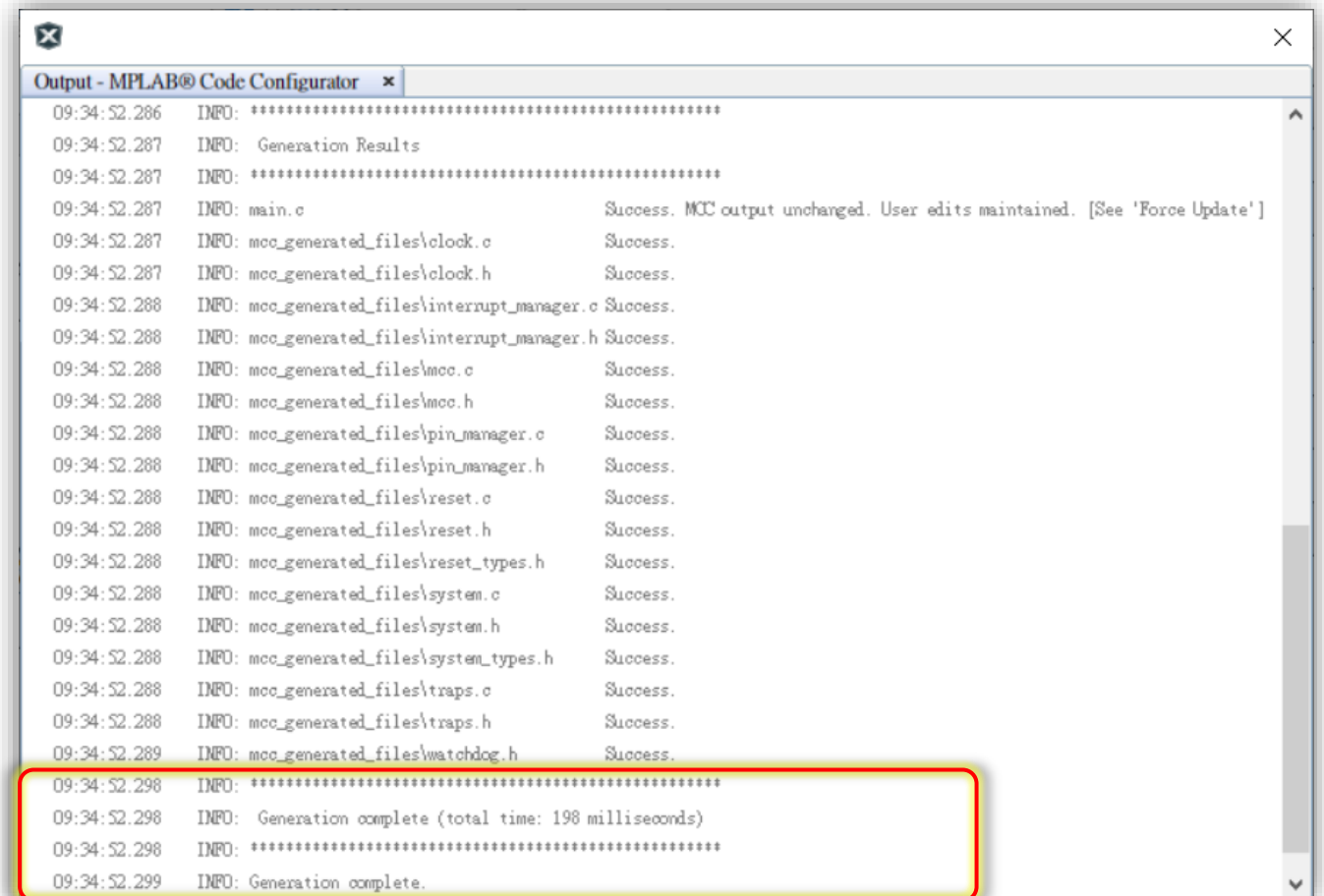
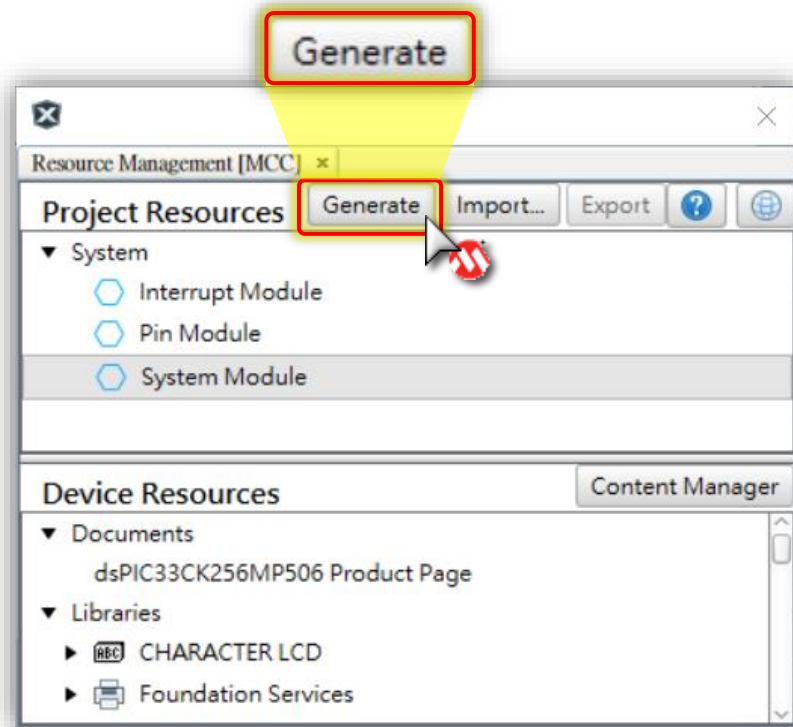
- Set **RD8** Start **High**.



Lab6 UART printf

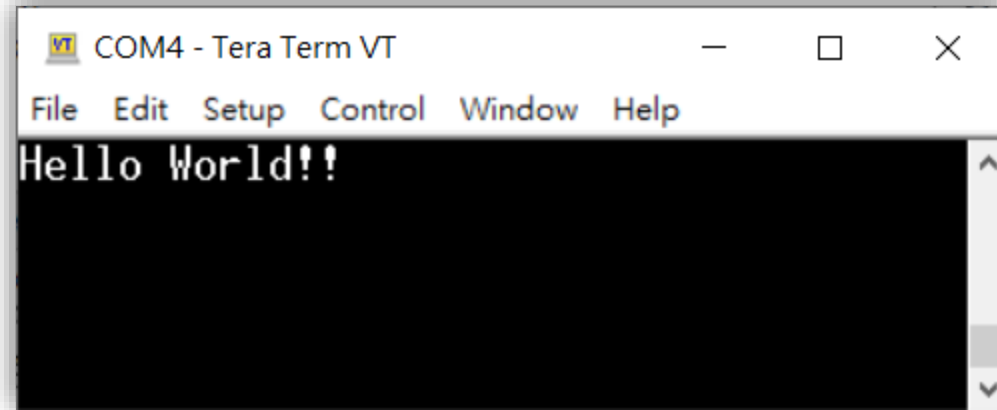
Step 6

- Click "**Generate**" Button to generate your code.



VT100 Console Control

- Use VT100 command to control UART output as



- `printf("\033[2J");` // Clear screen
- `printf("\033[?25l");` // Hide cursor
- `printf("\033[y;xH");` // Set cursor to (x,y)

Let's go!

Lab6 UART printf

Code Example

```
...
#include "Microchip_Splash.h"

void DisplayTask( void );

uint8_t StringBuffer[100];
volatile uint8_t TMR1Flag;

int main(void)
{
    ...
}

void TMR1_CallBack(void)
{
    TMR1Flag = 1;
    LED1_Toggle();
    LED2_Toggle();
}

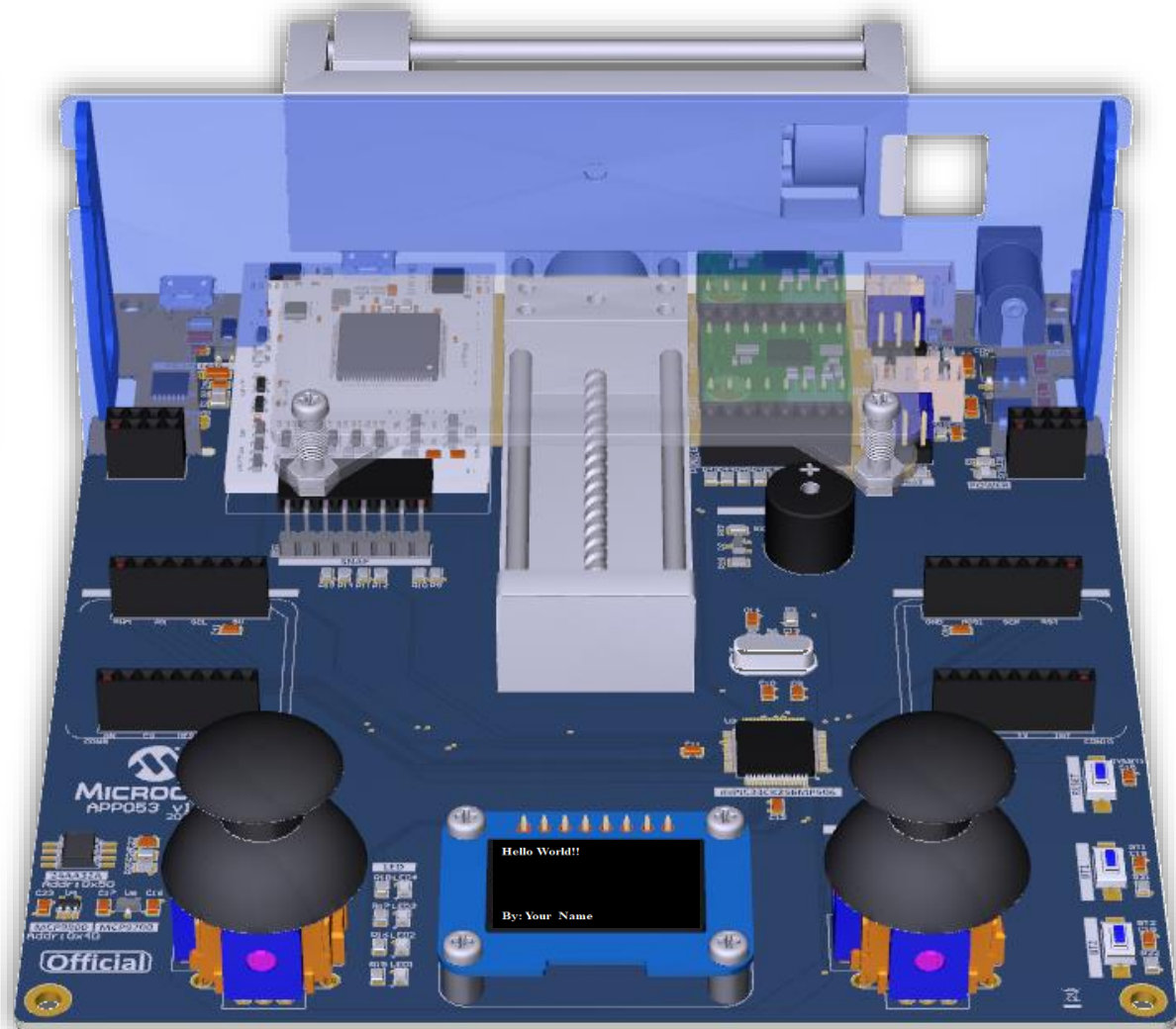
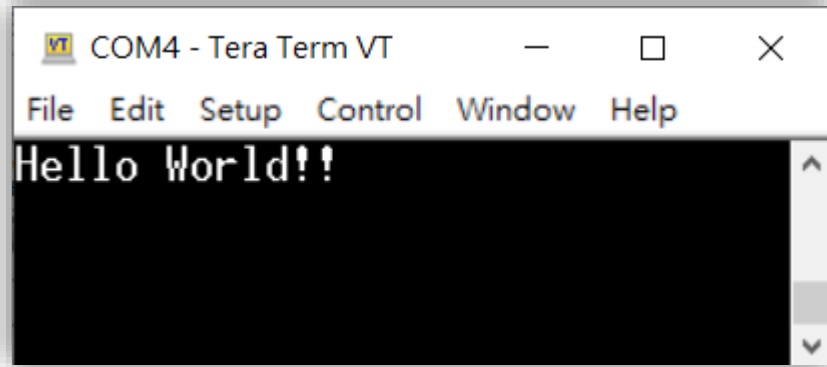
void DisplayTask( void )
{
    OLED_Clean();
    sprintf((char *) StringBuffer, "Hello World!!!");
    printf("\033[1;1H%s", StringBuffer);
    OLED_DrawString(0, 0, (char *) StringBuffer);
    OLED_DrawString(0, 56, "By:Your Name");
    OLED_Update();
}
```

```
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    ...
    OLED_Update();

    printf("\033[2J"); // Clear screen
    printf("\033[?25l"); // Hide cursor
    while (1)
    {
        if (TMR1Flag)
        {
            TMR1Flag=0;
            LED1_Toggle();
            LED2_Toggle();
            DisplayTask();
        }
        ...
    }
    return 1;
}
```

Lab6 UART printf

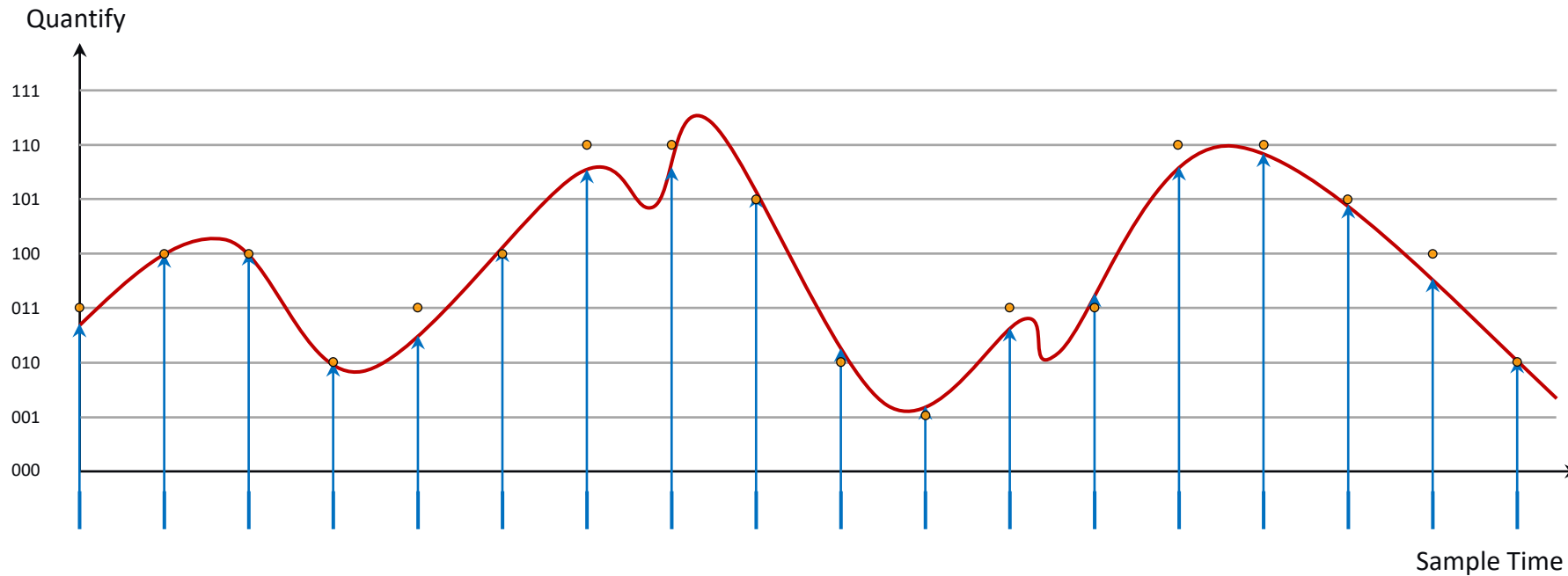
Result



12 Bits High Speed ADC Architecture

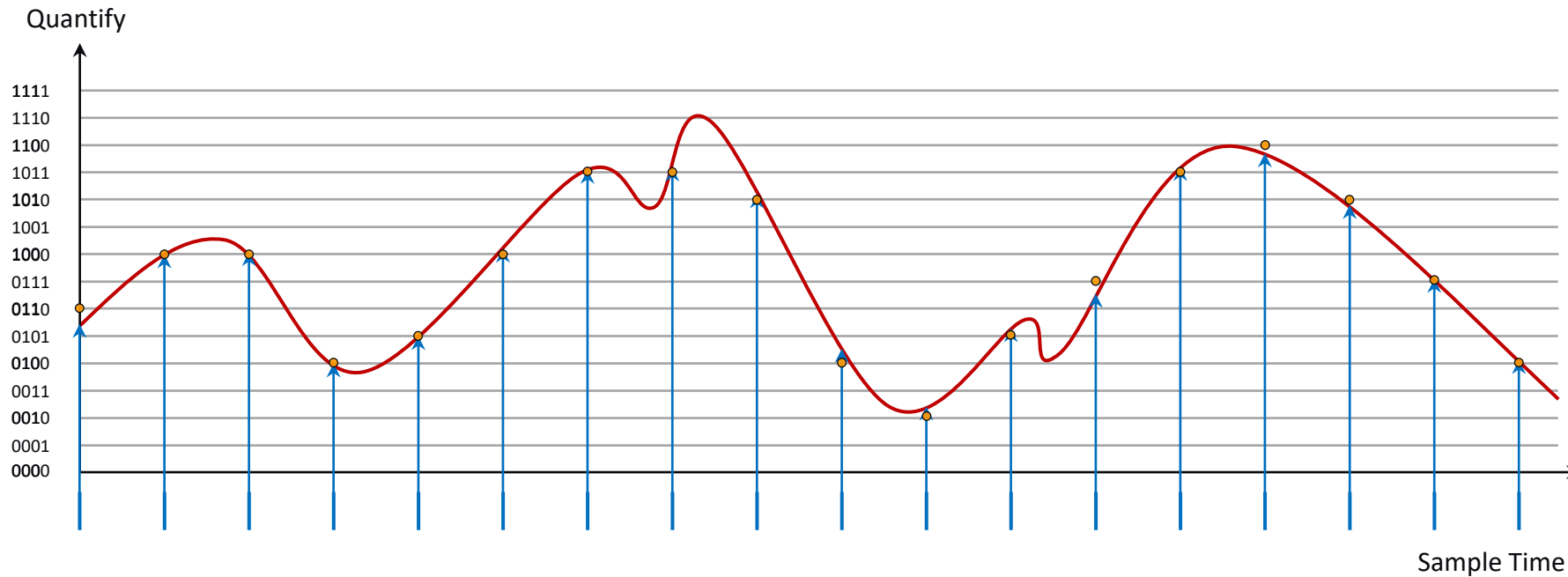
What's ADC ?

- The Analog-to-Digital Converter(ADC) converts a signal from the time/amplitude continuous domain into.
a time/amplitude discrete domain.



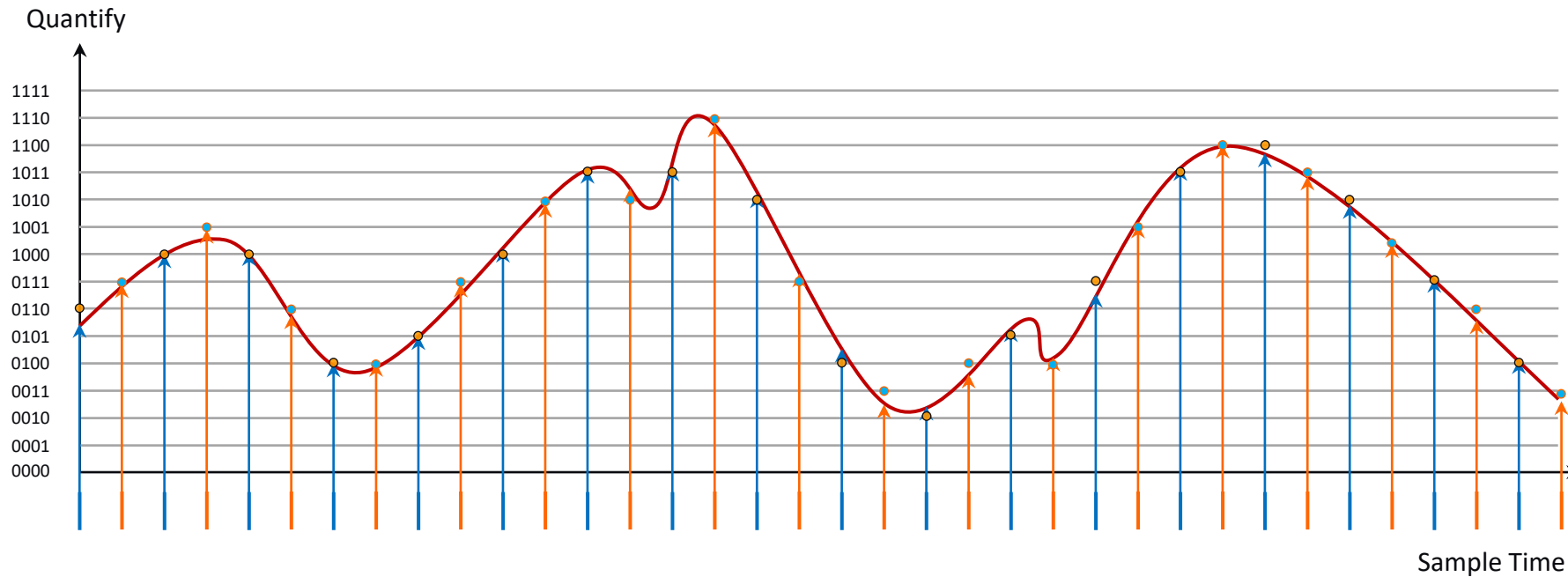
Resolution

- The resolution of the converter indicates the number of discrete values. The resolution determines the magnitude of the quantization error. (wiki)



Sample Rate

- The analog signal is required to define the rate at which new digital values are sampled from the analog signal.
- This faithful reproduction is only possible if the sampling rate is higher than twice the highest frequency of the signal. (wiki)

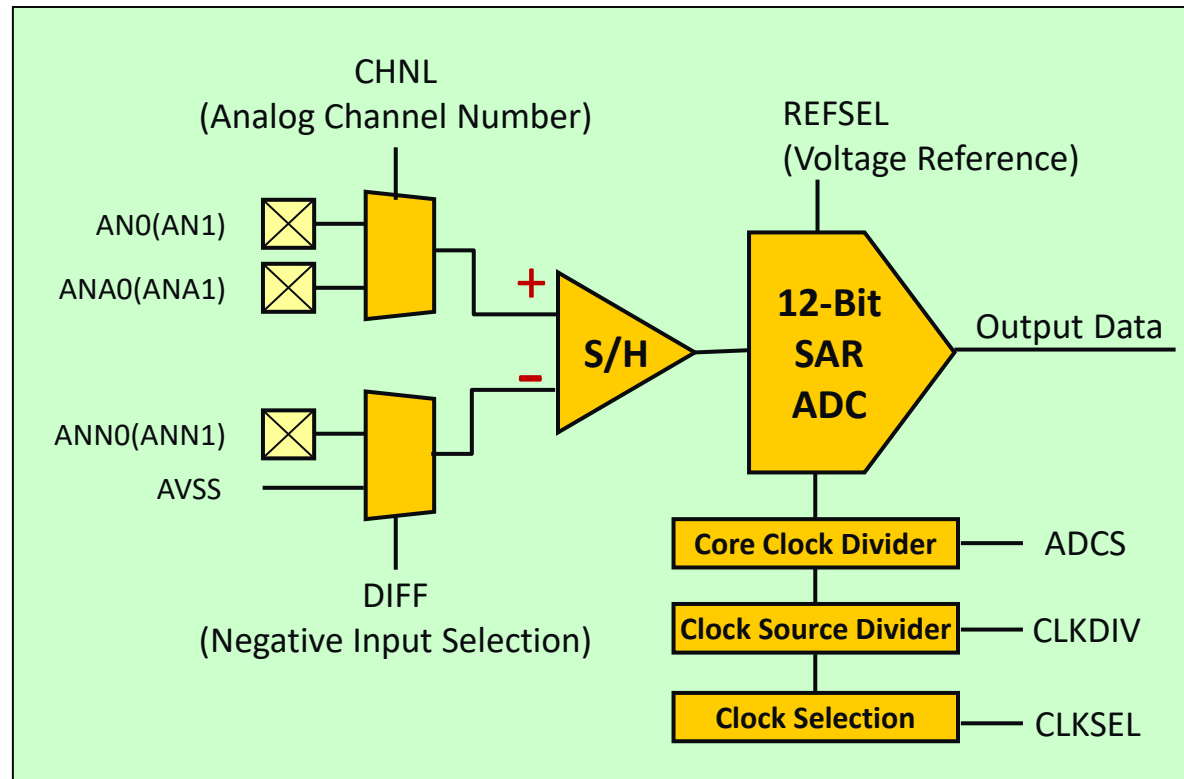


dsPIC33CK ADC Features

- The High-Speed, 12-Bit Multiple SARs Analog-to-Digital Converter (ADC) includes the following features:
 - Three ADC Cores: **Two Dedicated Cores** and **One Shared (common) Core**
 - User-Configurable **Resolution of up to 12 Bits, Up to 3.5 Msps Conversion Rate** per Channel at 12-Bit Resolution
 - Up to **24 Analog Input Channels**, with a Separate 16-Bit Conversion Result Register for each Input
 - Conversion Result can be Formatted as Unsigned or Signed Data
 - Simultaneous Sampling of up to Three Analog Inputs, Channel Scan Capability
 - Multiple Conversion Trigger Options for each Core, including:
 - PWM triggers from CPU cores
 - MCCP/SCCP modules triggers
 - CLC modules triggers
 - External pin trigger event (ADTRG31)
 - **Software trigger**
 - Four Integrated Digital Comparators with Dedicated Interrupts
 - Four Oversampling Filters with Dedicated Interrupts

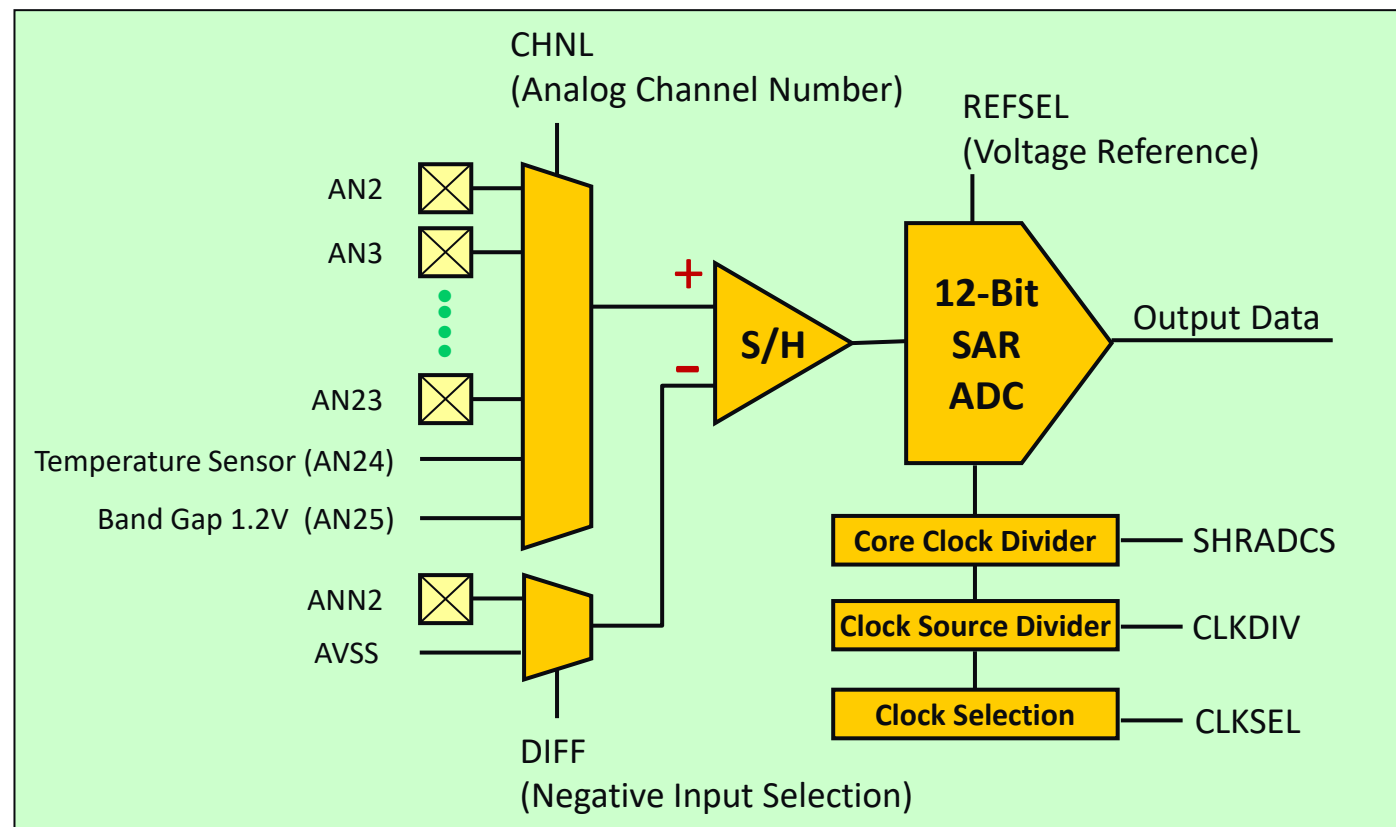
ADC Dedicate Core Channel Selection

- One analog input AN0(AN1) and one alternate input ANA0(ANA1) to connect as Analog Positive input of S/H.
- ANN0(ANN1) or AVSS as Analog Negative input of the S/H.
- The differential input voltage is **Positive – Negative**.



ADC Shared Core Channel Selection

- 24 analog channel inputs to connect as Analog Positive input of S/H.
- ANN2 or AVSS as Analog Negative input of the S/H.
- The differential input voltage is **Positive – Negative**.



Lab7 ADC Multiple Channel Periodically

Lab7 ADC Multiple Channel Periodically

- Base on current project or Open `.\Lab7_xxx.X` to do exercises.
- Try to add ADC1 module and conversion Joystick1 and Joystick2 voltage to shows ADC result on UART.
- Use Timer1 to software trigger of ADC conversion periodically.

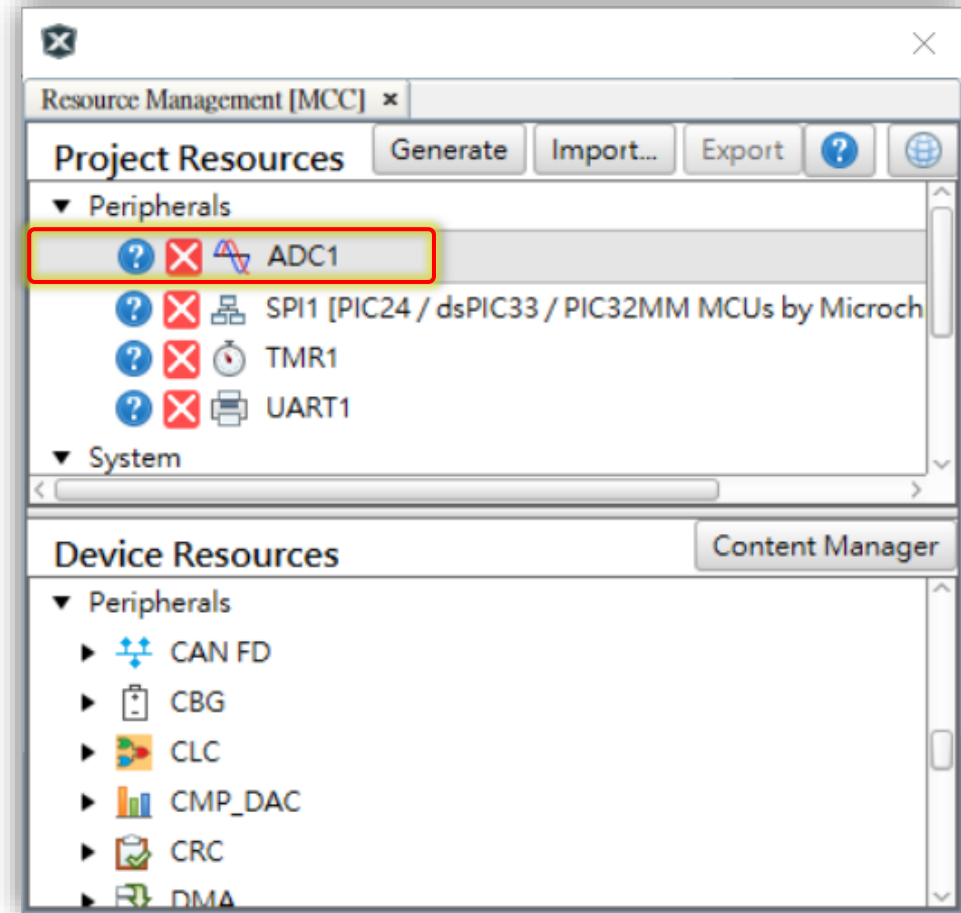
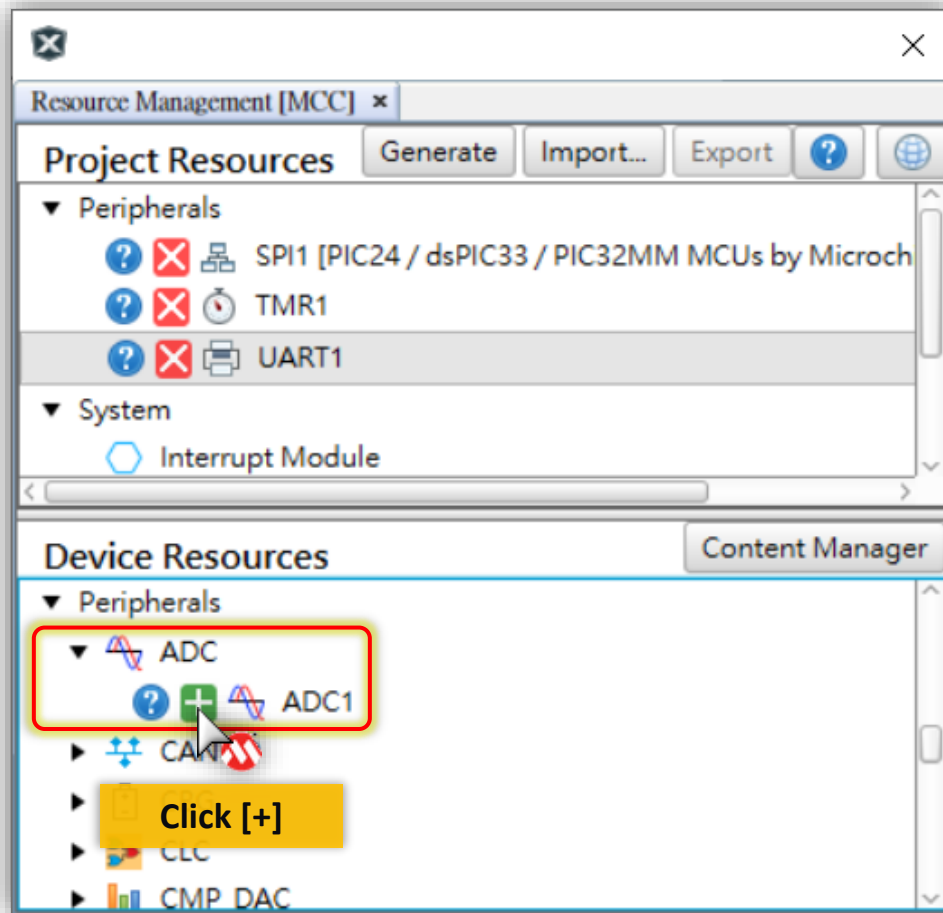
	Joystick1		Joystick2	
	X	Y	X	Y
PORT/Channel	RA4/AN4	RA1/ANA1	RA3/AN3	RA0/AN0
Custom Name	VR1	VR2	VR3	VR4
ADC Core	Share	Core1	Share	Core0
Trigger Source	Common Software Trigger			
Interrupt	Enable			

Let's go!

Lab7 ADC Multiple Channel Periodically

Step 1

- Click [+] to add **ADC** > **ADC1** Module.



Lab7 ADC Multiple Channel Periodically

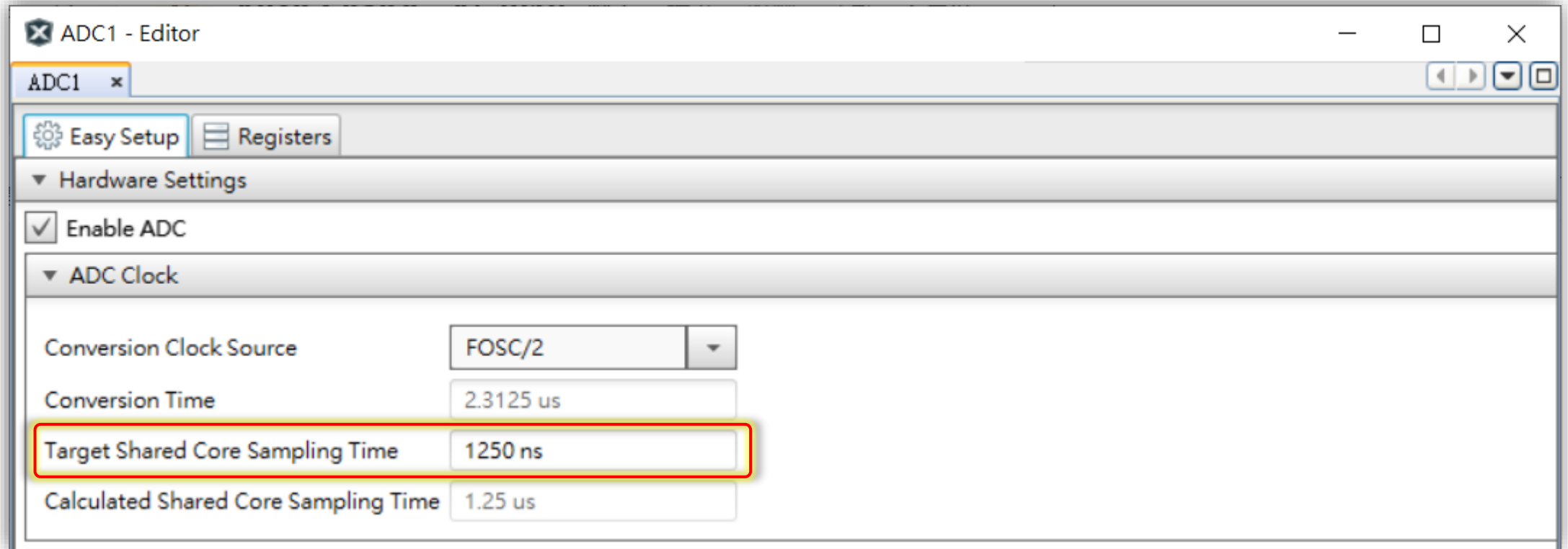
Notice

- For the **dedicated core**, the throughput includes **4 TADCORE sampling time** and **13 TADCORE conversion time**.
- For the **shared core**, the throughput includes **10 TADCORE sampling time** and **13 TADCORE conversion time**.
 - $ADC\ Clock\ Source = F_{osc} \div 2 \div 1(CLKDIV) = 32MHz \div 2 \div 1 = 16MHz$
 - $Dedicated\ Core\ Clock\ Source = ADC\ Clock\ Source \div 2(ADCS) = 16MHz \div 2 = 8MHz$
 - $Shared\ Core\ Clock\ Source = ADC\ Clock\ Source \div 2(SHRADCS) = 16MHz \div 2 = 8MHz$
 - $Dedicated\ Core\ T_{ADCORE} = 1 \div Dedicated\ Core\ Clock = (1 \div 8MHz) = 125ns$
 - $Shared\ Core\ T_{ADCORE} = 1 \div Shared\ Core\ Clock = (1 \div 8MHz) = 125n$
 - **Dedicated core sampling time : $4T_{ADCORE} = (125ns \times 4) = 500ns$**
 - **Shared core sampling time : $10T_{ADCORE} = (125ns \times 10) = 1,250ns$**
 - **sampling time set to maximum(500ns, 1250ns).**

Lab7 ADC Multiple Channel Periodically

Step 2

- Set Target Shared Core Sampling Time : 250ns > 1,250ns.



The screenshot shows the 'ADC1 - Editor' window with the 'Easy Setup' tab selected. Under 'Hardware Settings', 'Enable ADC' is checked. Under 'ADC Clock', the 'Conversion Clock Source' is set to 'FOSC/2'. The 'Conversion Time' is 2.3125 us. The 'Target Shared Core Sampling Time' is 1250 ns, which is highlighted with a red box. The 'Calculated Shared Core Sampling Time' is 1.25 us.

Parameter	Value
Conversion Clock Source	FOSC/2
Conversion Time	2.3125 us
Target Shared Core Sampling Time	1250 ns
Calculated Shared Core Sampling Time	1.25 us

Lab7 ADC Multiple Channel Periodically

Step 3

- Set **Enable AN4 Shared Core**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16	RC7		None	None	☐
Shared	☐	AN17	RC6		None	None	☐
Shared	☐	AN18	RD10		None	None	☐
Shared	☐	AN19	RD11		None	None	☐
Shared	☐	AN2	RB7		None	None	☐
Shared	☐	AN3	RA3		None	None	☐
Shared	☒	AN4	RA4	channel_AN4	None	None	☐
Shared	☐	AN5	RB0		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 4

- Set **Trigger Source** : **None** > **Common Software Trigger**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16 ▼	RC7		None ▼	None ▼	☐
Shared	☐	AN17 ▼	RC6		None ▼	None ▼	☐
Shared	☐	AN18 ▼	RD10		None ▼	None ▼	☐
Shared	☐	AN19 ▼	RD11		None ▼	None ▼	☐
Shared	☐	AN2 ▼	RB7		None ▼	None ▼	☐
Shared	☐	AN3 ▼	RA3		None ▼	None ▼	☐
Shared	☒	AN4 ▼	RA4	channel_AN4	Common Soft... ▼	None ▼	☐
Shared	☐	AN5 ▼	RB0		None ▼	None ▼	☐

Lab7 ADC Multiple Channel Periodically

Step 5

- Set **Enable Interrupt**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16	RC7		None	None	☐
Shared	☐	AN17	RC6		None	None	☐
Shared	☐	AN18	RD10		None	None	☐
Shared	☐	AN19	RD11		None	None	☐
Shared	☐	AN2	RB7		None	None	☐
Shared	☐	AN3	RA3		None	None	☐
Shared	☒	AN4	RA4	channel_AN4	Common Soft...	None	☒
Shared	☐	AN5	RB0		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 6

- Set Core Channel : **AN1 > ANA1**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☐	AN0	RA0		None	None	☐
Core1	☐	ANA1	RA1		None	None	☐
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 7

- Set **Enable ANA1 Dedicate Core1**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☐	AN0	RA0		None	None	☐
Core1	☒	ANA1	RA1	channel_ANA1	None	None	☐
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 8

- Set **Trigger Source** : **None** > **Common Software Trigger**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☐	AN0	RA0		None	None	☐
Core1	☒	ANA1	RA1	channel_ANA1	Common Soft...	None	☐
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 9

- Set **Enable Interrupt**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☐	AN0	RA0		None	None	☐
Core1	☒	ANA1	RA1	channel_ANA1	Common Soft...	None	☒
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 10

- Set **Enable AN3 Shared Core**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16 ▼	RC7		None ▼	None ▼	☐
Shared	☐	AN17 ▼	RC6		None ▼	None ▼	☐
Shared	☐	AN18 ▼	RD10		None ▼	None ▼	☐
Shared	☐	AN19 ▼	RD11		None ▼	None ▼	☐
Shared	☐	AN2 ▼	RB7		None ▼	None ▼	☐
Shared	☒	AN3 ▼	RA3	channel_AN3	None ▼	None ▼	☐
Shared	☒	AN4 ▼	RA4	channel_AN4	Common Soft... ▼	None ▼	☒
Shared	☐	AN5 ▼	RB0		None ▼	None ▼	☐

Lab7 ADC Multiple Channel Periodically

Step 11

- Set **Trigger Source** : **None** > **Common Software Trigger**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16 ▼	RC7		None ▼	None ▼	☐
Shared	☐	AN17 ▼	RC6		None ▼	None ▼	☐
Shared	☐	AN18 ▼	RD10		None ▼	None ▼	☐
Shared	☐	AN19 ▼	RD11		None ▼	None ▼	☐
Shared	☐	AN2 ▼	RB7		None ▼	None ▼	☐
Shared	☒	AN3 ▼	RA3	channel_AN3	Common Soft... ▼	None ▼	☐
Shared	☒	AN4 ▼	RA4	channel_AN4	Common Soft... ▼	None ▼	☒
Shared	☐	AN5 ▼	RB0		None ▼	None ▼	☐

Lab7 ADC Multiple Channel Periodically

Step 12

- Set **Enable Interrupt**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Shared	☐	AN16 ▼	RC7		None ▼	None ▼	☐
Shared	☐	AN17 ▼	RC6		None ▼	None ▼	☐
Shared	☐	AN18 ▼	RD10		None ▼	None ▼	☐
Shared	☐	AN19 ▼	RD11		None ▼	None ▼	☐
Shared	☐	AN2 ▼	RB7		None ▼	None ▼	☐
Shared	☒	AN3 ▼	RA3	channel_AN3	Common Soft... ▼	None ▼	☒
Shared	☒	AN4 ▼	RA4	channel_AN4	Common Soft... ▼	None ▼	☒
Shared	☐	AN5 ▼	RB0		None ▼	None ▼	☐

Lab7 ADC Multiple Channel Periodically

Step 13

- Set **Enable AN0 Dedicate Core0**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☒	AN0	RA0	channel_AN0	None	None	☐
Core1	☒	ANA1	RA1	channel_ANA1	Common Soft...	None	☒
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 14

- Set **Trigger Source** : **None** > **Common Software Trigger**.

▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☒	AN0	RA0	channel_AN0	Common Soft...	None	☐
Core1	☒	ANA1	RA1	channel_ANA1	Common Soft...	None	☒
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 15

- Set **Enable Interrupt**.

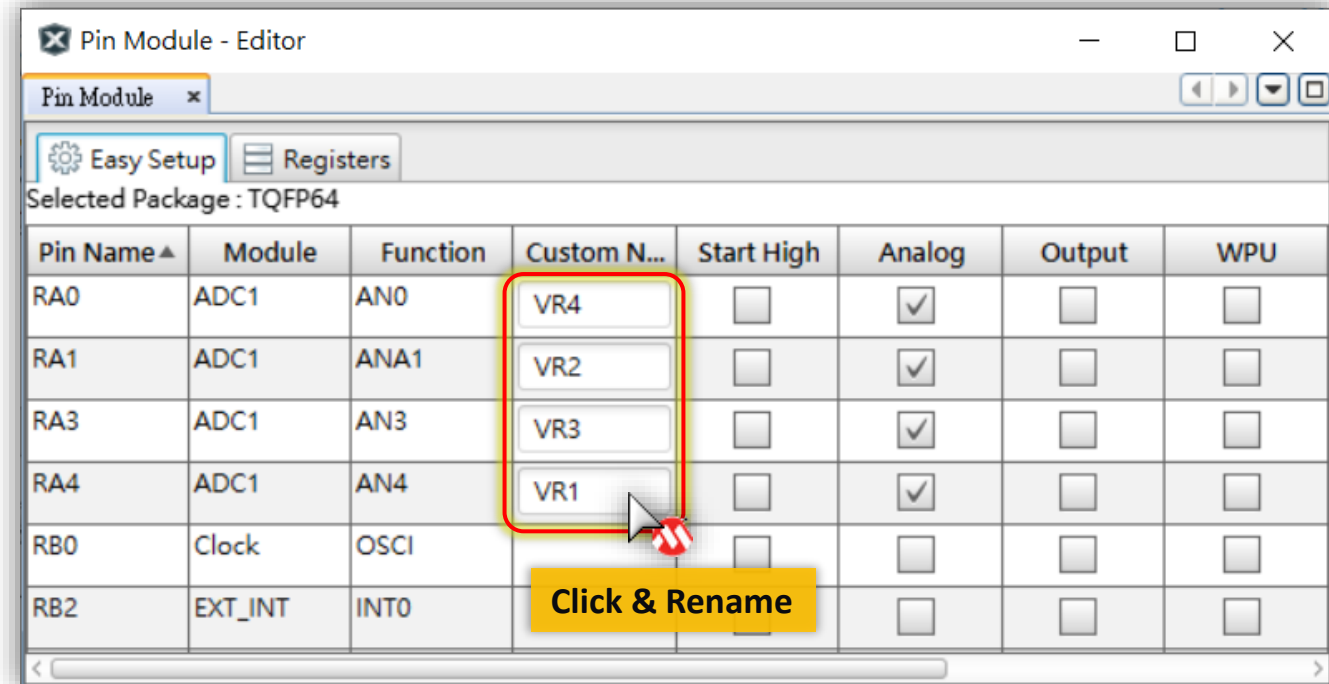
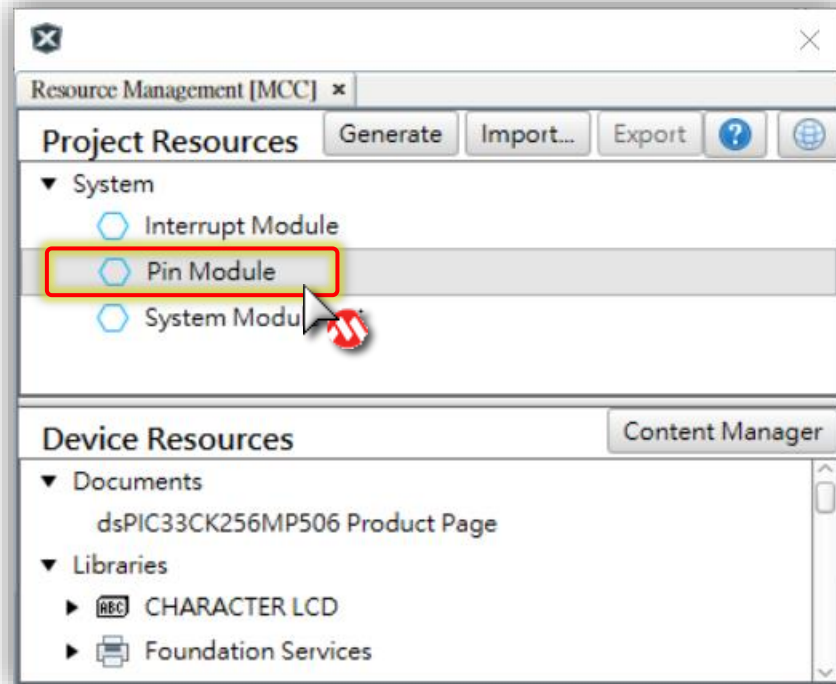
▼ Selected Channels

Core	Enable	Core Channel	Pin Name	Custom Name	Trigger Source	Compare	Interrupt
Core0	☒	AN0	RA0	channel_AN0	Common Soft...	None	☒
Core1	☒	ANA1	RA1	channel_ANA1	Common Soft...	None	☒
Shared	☐	AN10	RB8		None	None	☐
Shared	☐	AN11	RB9		None	None	☐
Shared	☐	AN12	RC0		None	None	☐
Shared	☐	AN13	RC1		None	None	☐
Shared	☐	AN14	RC2		None	None	☐
Shared	☐	AN15	RC3		None	None	☐

Lab7 ADC Multiple Channel Periodically

Step 16

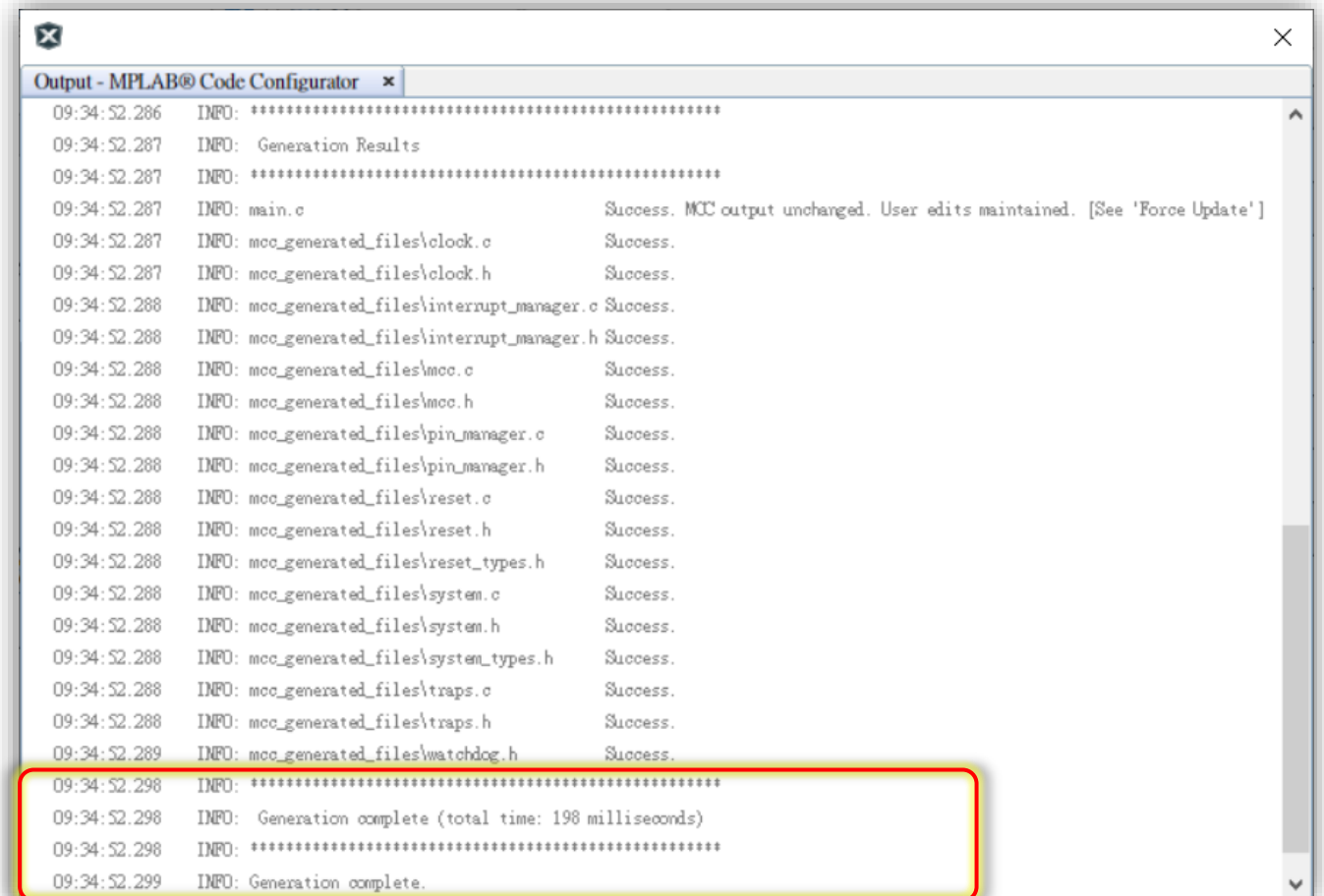
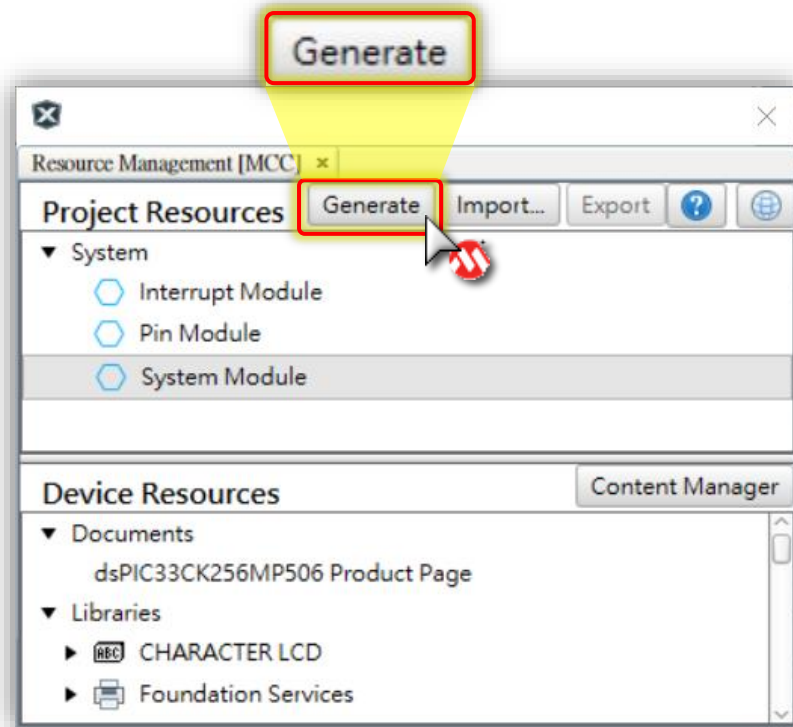
- Set Alias for RD0, RD1, RD2.
 - Pin Module : RA4(AN4) Custom Name > VR1, RA1 (ANA1) Custom Name > VR2.
 - Pin Module : RA3 (AN3) Custom Name > VR3, RA0 (AN0) Custom Name > VR4.



Lab7 ADC Multiple Channel Periodically

Step 17

- Click "**Generate**" Button to generate your code.



ADC Functions of MCC

- MCC Provide below functions for ADC (Interrupt Enabled):

Prototype definition : "adc1.h"

- void ADC1_Initialize(void);
- void ADC1_Core0PowerEnable () ;
- void ADC1_Core1PowerEnable () ;
- void ADC1_SharedCorePowerEnable () ;
- void ADC1_SetCustomNameInterruptHandler(void* handler);
- **ADC1_CustomName_CallBack(uint16_t adcVal);**
- void ADC1_Setchannel_AN24InterruptHandler(void* handler);
- void ADC1_Setchannel_AN25InterruptHandler(void* handler);
- ADC1_channel_AN24_CallBack(uint16_t adcVal);
- ADC1_channel_AN25_CallBack(uint16_t adcVal);

Lab7 ADC Multiple Channel Periodically

Code Example

```
...
#include "Microchip_Splash.h"
#include "mcc_generated_files/adc1.h"

void DisplayTask( void );

volatile uint8_t TMR1Flag;
volatile uint16_t ADC1_VR1_Buffer = 0;
volatile uint16_t ADC1_VR2_Buffer = 0;
volatile uint16_t ADC1_VR3_Buffer = 0;
volatile uint16_t ADC1_VR4_Buffer = 0;

int main(void)
{
    ...
}

void TMR1_CallBack(void)
{
    TMR1Flag = 1;
}

void ADC1_VR1_CallBack (uint16_t adcVal)
{
    ADC1_VR1_Buffer = adcVal;
}

void ADC1_VR2_CallBack (uint16_t adcVal)
{
    ADC1_VR2_Buffer = adcVal;
}
```

```
void ADC1_VR3_CallBack (uint16_t adcVal)
{
    ADC1_VR3_Buffer = adcVal;
}

void ADC1_VR4_CallBack (uint16_t adcVal)
{
    ADC1_VR4_Buffer = adcVal;
}

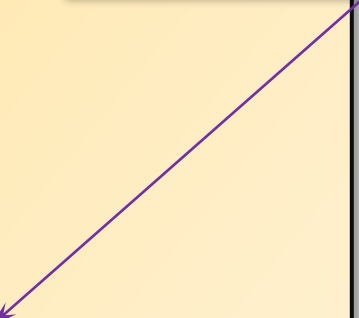
void DisplayTask( void )
{
    OLED_Clean();
    sprintf((char *) StringBuffer, "Hello World!!");
    printf("\033[1;1H%s", StringBuffer);
    OLED_DrawString(0, 0, (char *) StringBuffer);
    sprintf((char *) StringBuffer, "VR1:%4d", ADC1_VR1_Buffer);
    printf("\033[2;1H%s", StringBuffer);
    OLED_DrawString(0, 8, (char *) StringBuffer);
    sprintf((char *) StringBuffer, "VR2:%4d", ADC1_VR2_Buffer);
    printf("\033[3;1H%s", StringBuffer);
    OLED_DrawString(60, 8, (char *) StringBuffer);
    sprintf((char *) StringBuffer, "VR3:%4d", ADC1_VR3_Buffer);
    printf("\033[4;1H%s", StringBuffer);
    OLED_DrawString(0, 16, (char *) StringBuffer);
    sprintf((char *) StringBuffer, "VR4:%4d", ADC1_VR4_Buffer);
    printf("\033[5;1H%s", StringBuffer);
    OLED_DrawString(60, 16, (char *) StringBuffer);
    OLED_DrawString(0, 56, "By:Your Name");
    OLED_Update();
}
```

Lab7 ADC Multiple Channel Periodically

Code Example

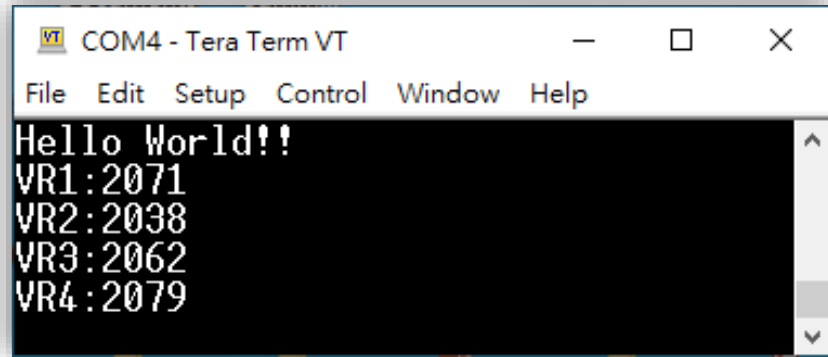
```
...  
  
int main(void)  
{  
    // initialize the device  
    SYSTEM_Initialize();  
    ...  
    printf("\033[2J"); // Clear screen  
    printf("\033[?25l"); // Hide cursor  
    while (1)  
    {  
        if (TMR1Flag)  
        {  
            TMR1Flag = 0;  
            LED1_Toggle();  
            LED2_Toggle();  
            ADC1_SoftwareTriggerEnable();  
            DisplayTask();  
        }  
  
        if (BT1_GetValue())  
            LED3_SetLow();  
        else  
            LED3_SetHigh();  
        ...  
    }  
    return 1;  
}
```

**Timer1 for ADC
Software Trigger periodically**

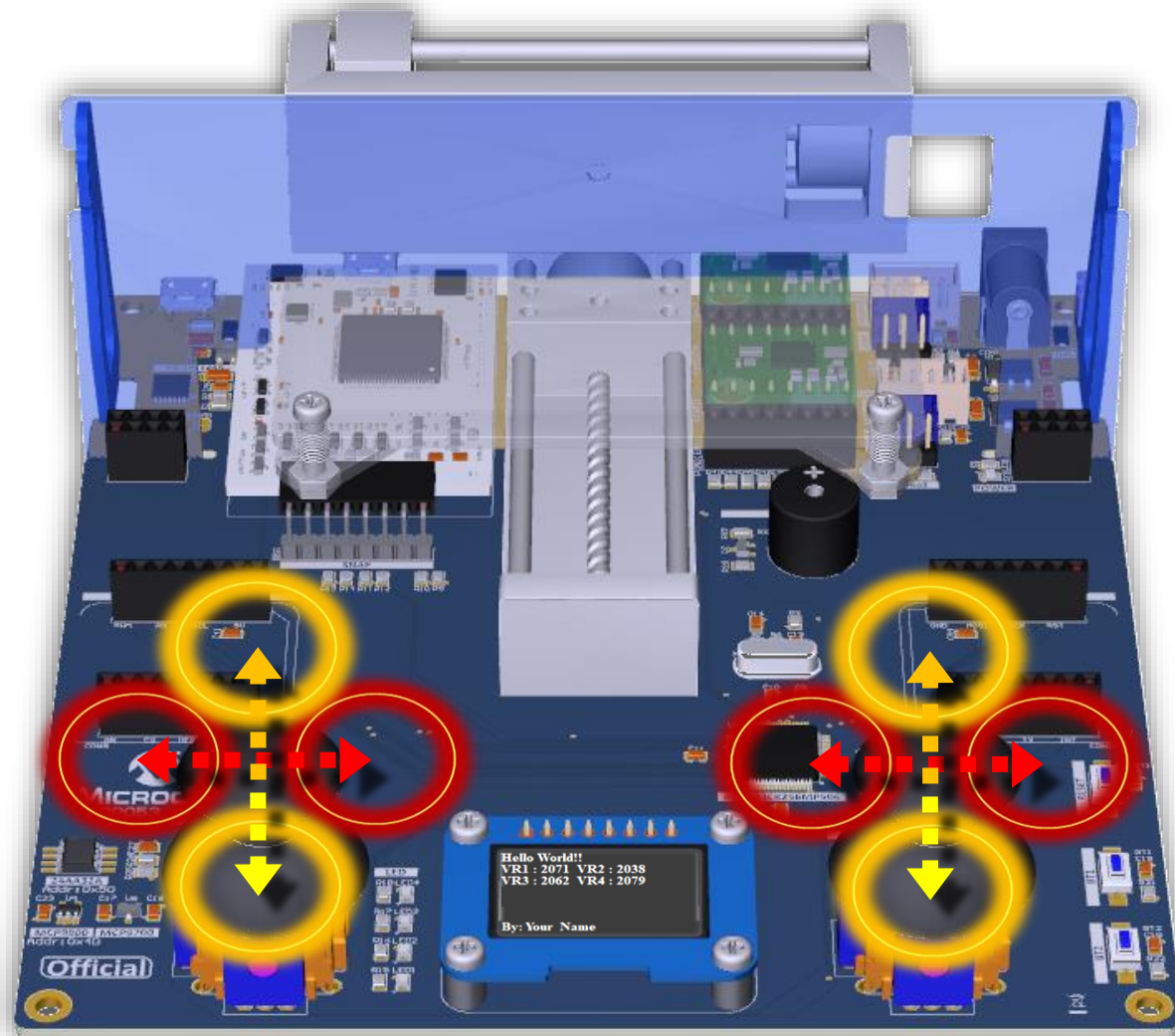


Lab7 ADC Multiple Channel Periodically

Result



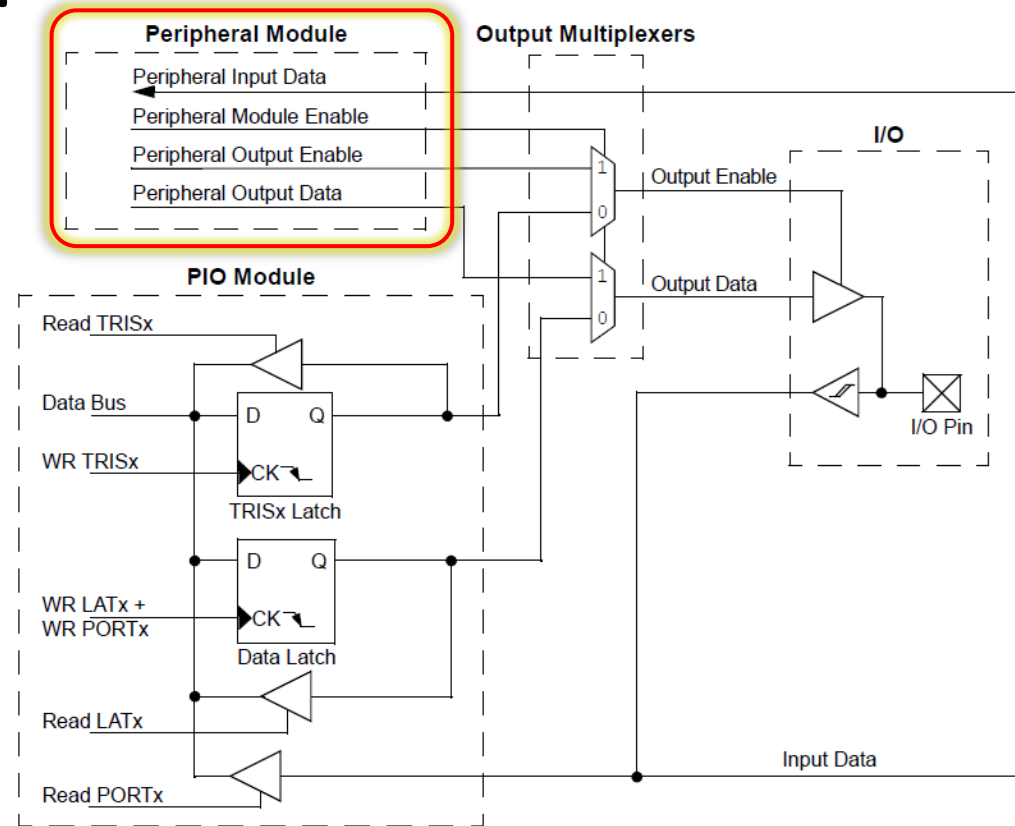
```
VT COM4 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1:2071
VR2:2038
VR3:2062
VR4:2079
```



External Interrupt Architecture

EXT_INT Module

- Interrupts are signals provided to internal peripheral modules or from external pins of the MCU.
- dsPIC33CK256MP506 provides **4 Ext Interrupt Pins**.
- Most external Interrupt pins have flexibility of **Peripheral Select Pin(PPS)**, except the **INT0** has **dedicated pin**.



Lab8 Multi-Ext Interrupt

Lab8 Multi-Ext Interrupt

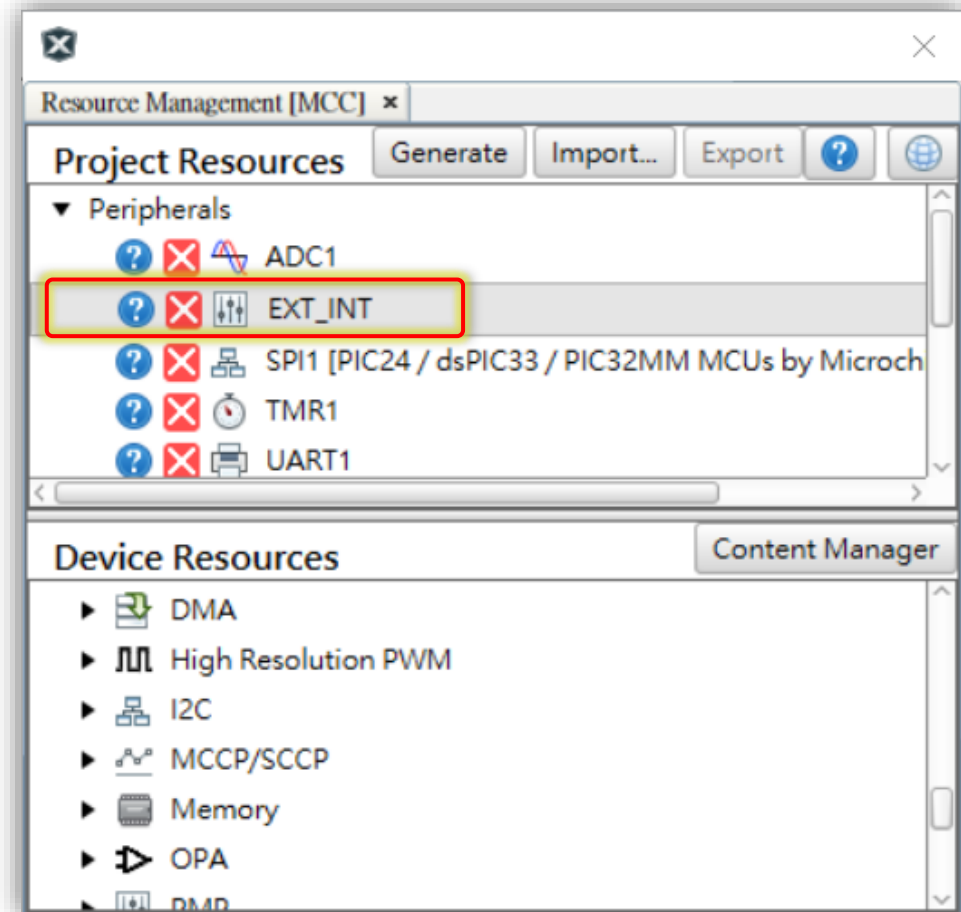
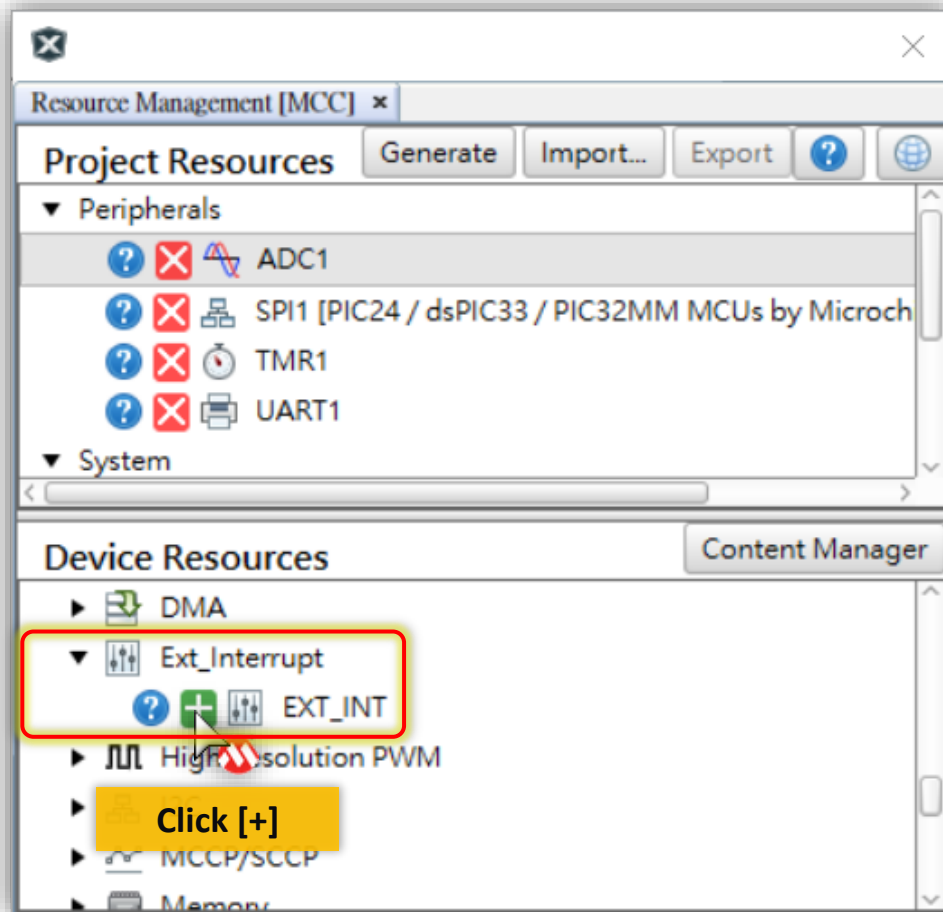
- Base on current project or Open `.\Lab8_xxx.X` to do exercises.
- To add EXT_INT to your project and enable channel **INT0** and **INT1**.
- Set EXT_INT Edge Detect to **Negative Edge**.
- Try use Joystick' button to toggle the LED.
 - Joystick1 Button(**INT0**, **RB2**) to toggle **LED3**.
 - Joystick2 Button(**INT1**, **RC12**) to toggle **LED4**.

Let's go!

Lab8 Multi-Ext Interrupt

Step 1

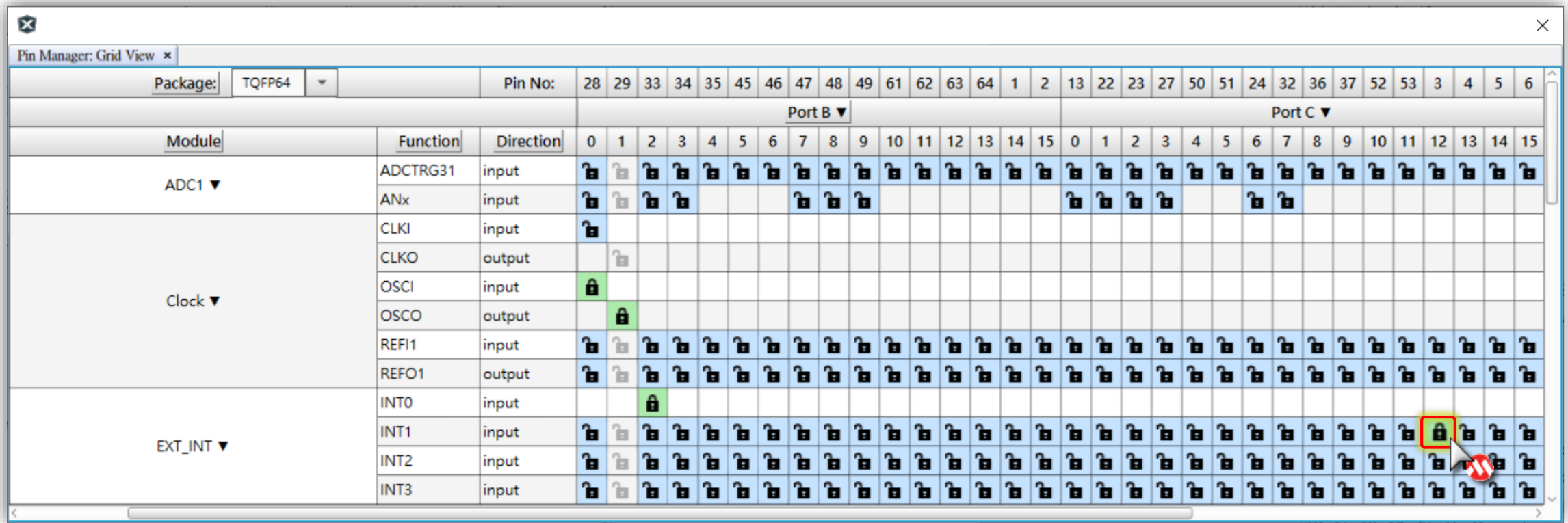
- Click [+] to add **Ext_Interrupt** > **EXT_INT** module.



Lab8 Multi-Ext Interrupt

Step 2

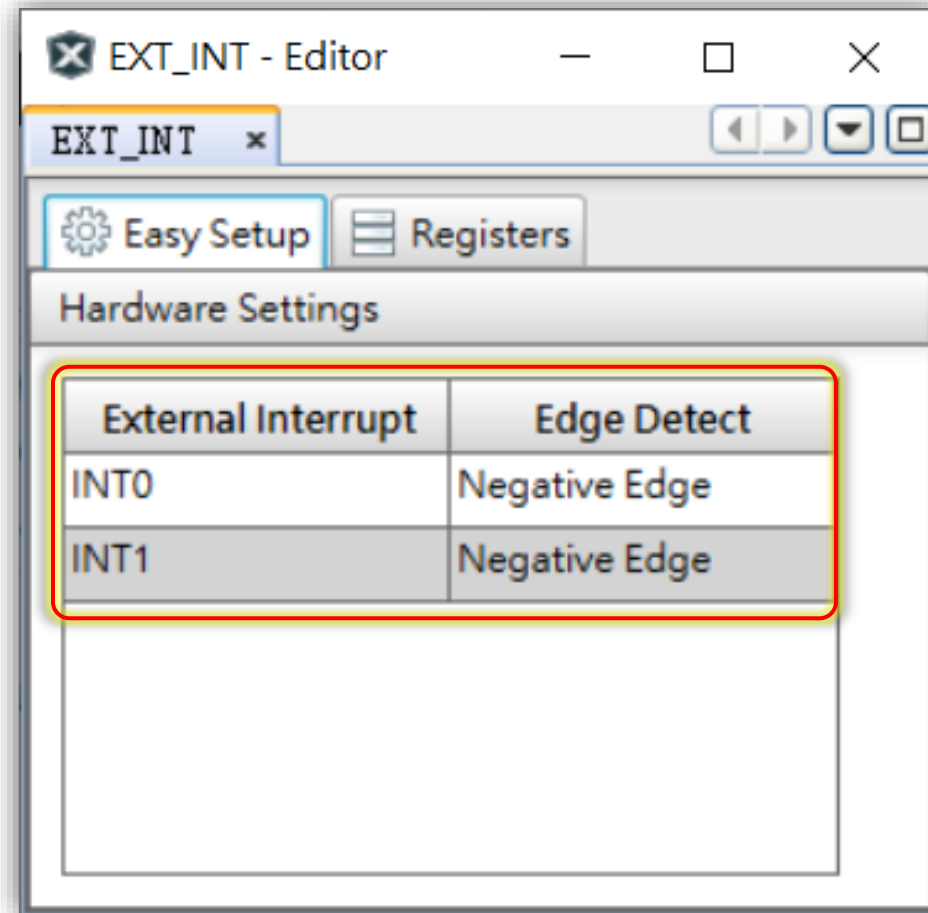
- **Assign INT1 Pin.**
 - **Pin Manager** : **RC12 > INT1.**



Lab8 Multi-Ext Interrupt

Step 3

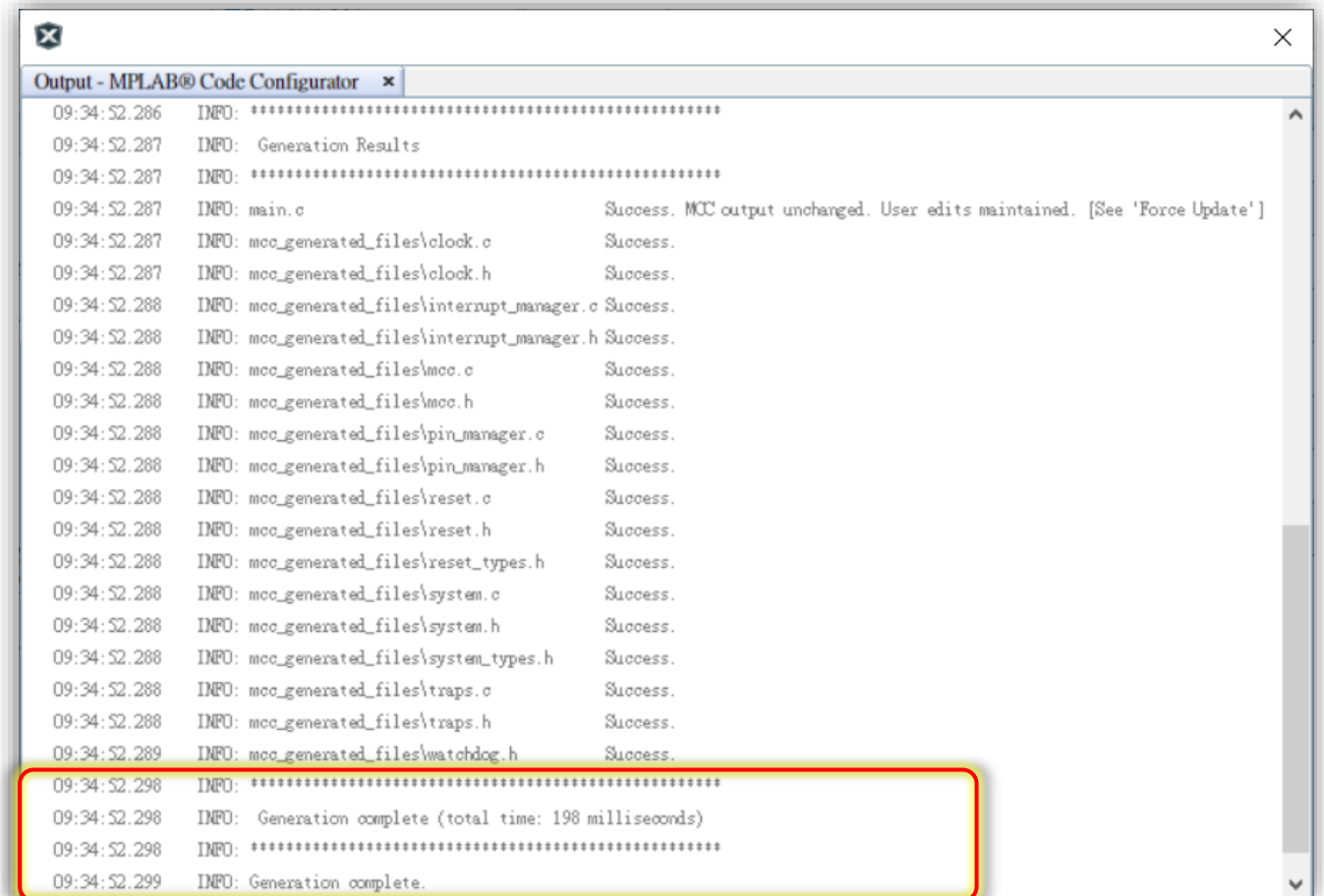
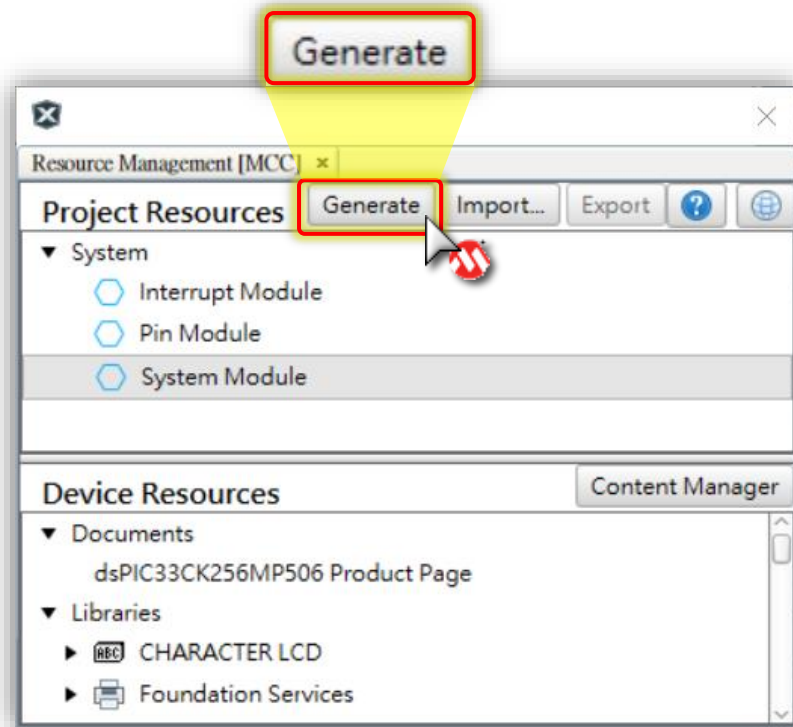
- Set both **INT0** and **INT1** Edge Detect : **Positive Edge > Negative Edge**.



Lab8 Multi-Ext Interrupt

Step 4

- Click "**Generate**" Button to generate your code.



EXT_INT Functions of MCC

- MCC Provide below functions for Ext_Interrupt:

Prototype definition : "ext_int.h"

- void EXT_INT_Initialize (void);
- void EXT_INT_n_CallBack(void);

Lab8 Multi-Ext Interrupt

Code Example

```
...
#include "mcc_generated_files/adc1.h"
#include "mcc_generated_files/ext_int.h"

int main(void)
{
    ...
}

void TMR1_CallBack(void)
{
    TMR1Flag = 1;
}

void DisplayTask( void )
{
    ...
}

void EX_INT0_CallBack(void)
{
    LED3_Toggle();
}

void EX_INT1_CallBack(void)
{
    LED4_Toggle();
}
```

```
...

int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

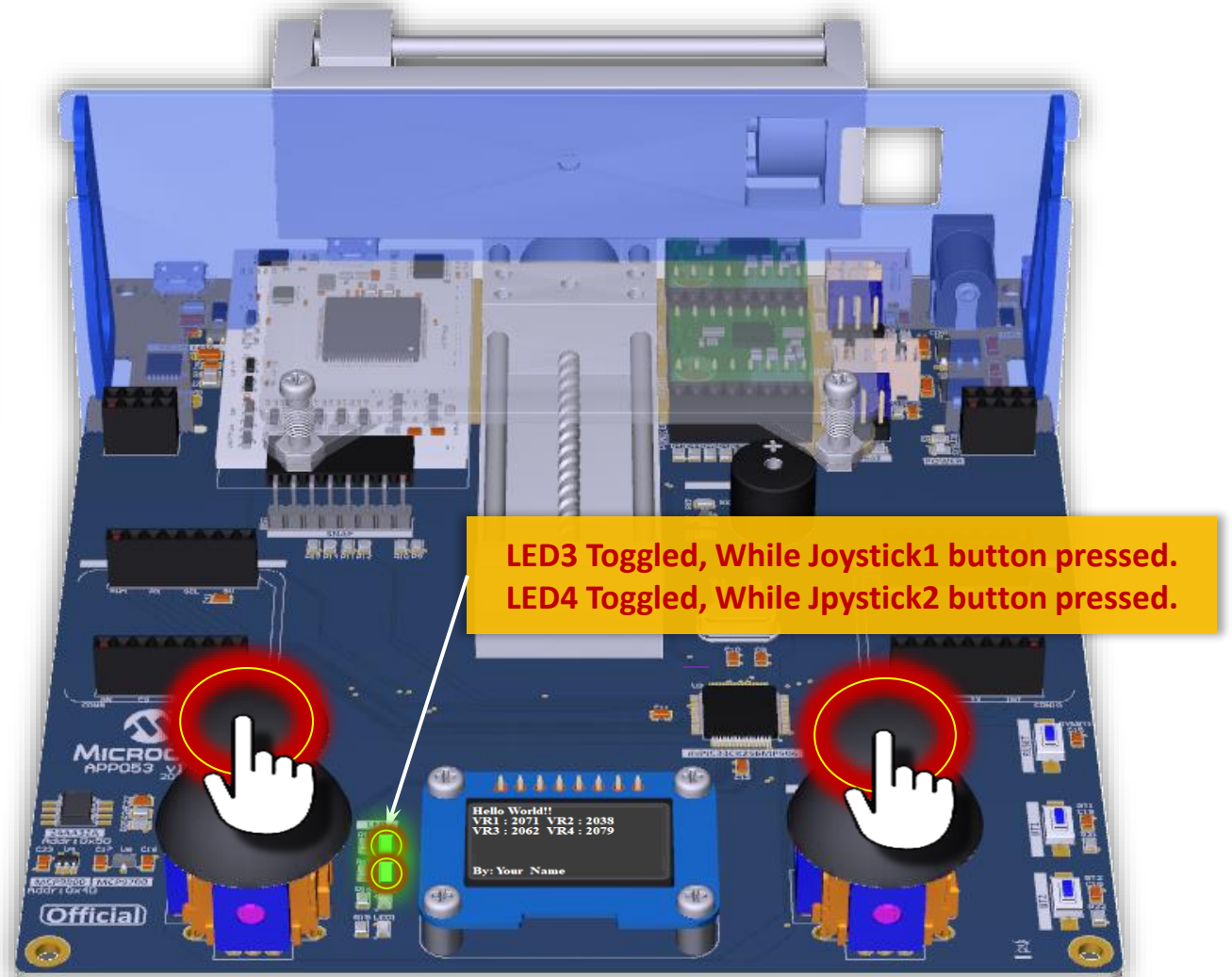
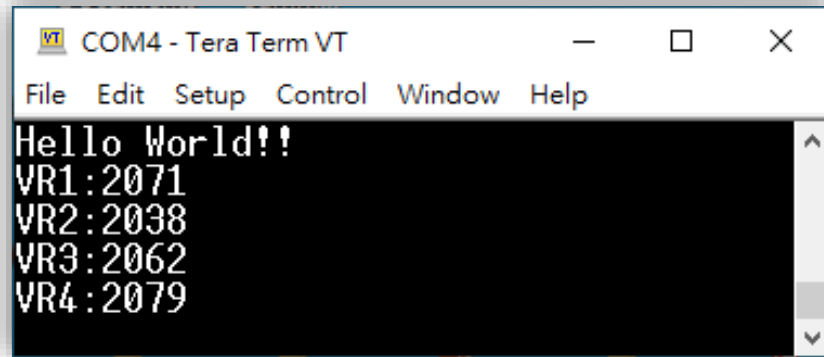
    ...
    printf("\033[2J"); // Clear screen
    printf("\033[?25l"); // Hide cursor
    while (1)
    {
        if (TMR1Flag)
        {
            ...
            DisplayTask();
        }

        if (BT1_GetValue())
            LED3_SetLow();
        else
            LED3_SetHigh();

        if (BT2_GetValue())
            LED4_SetLow();
        else
            LED4_SetHigh();
    }
    return 1;
}
```


Lab8 Multi-Ext Interrupt

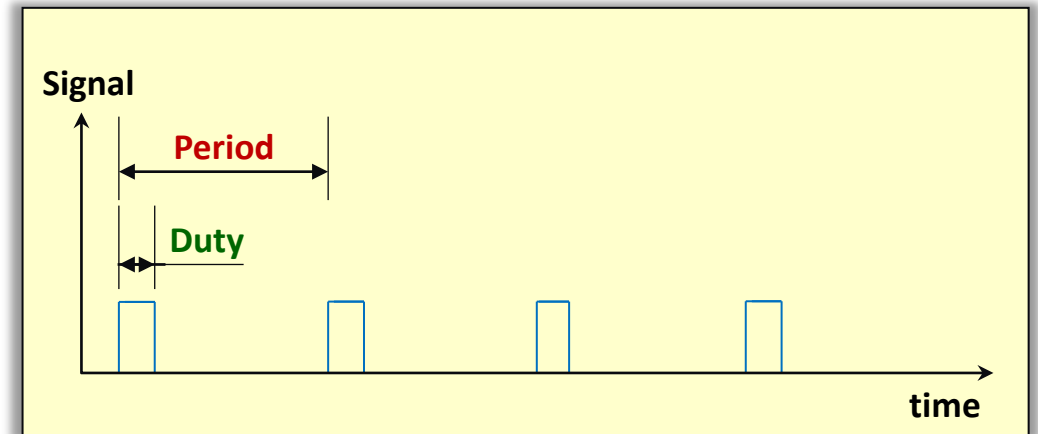
Result



High Resolution PWM Architecture

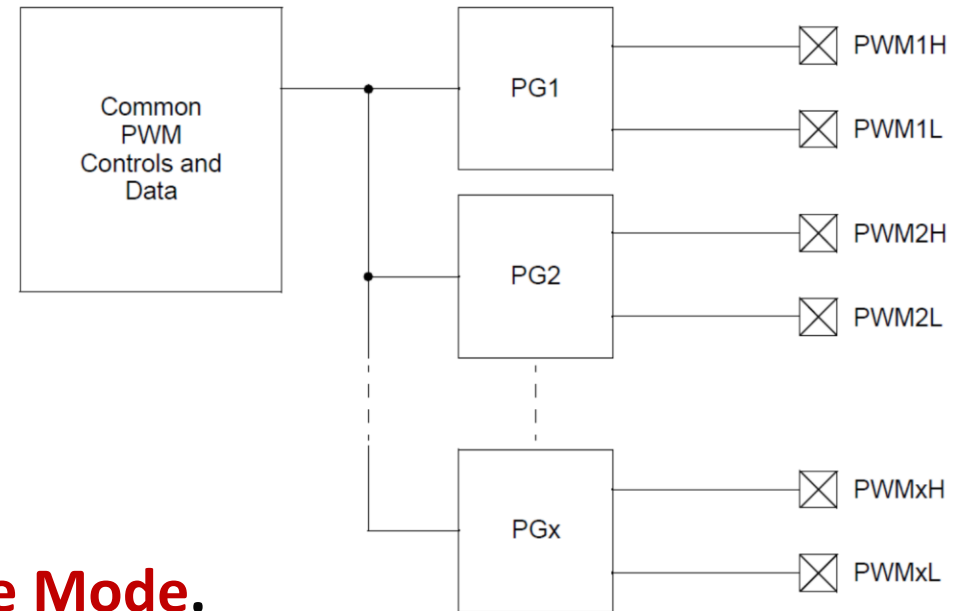
What's PWM

- The Pulse-width modulation (PWM) Waveform is a modulation technique used to encode a message into a pulsing signal.
- Its main use is to allow the control of the power supplied to electrical devices, ex. motor control, ballast, LED, H-bridge, power converters, and other types of power control applications.
- There are two important terms
 - **Period :**
The single square wave duration.
 - **Duty :**
The proportion of active time to the regular interval.



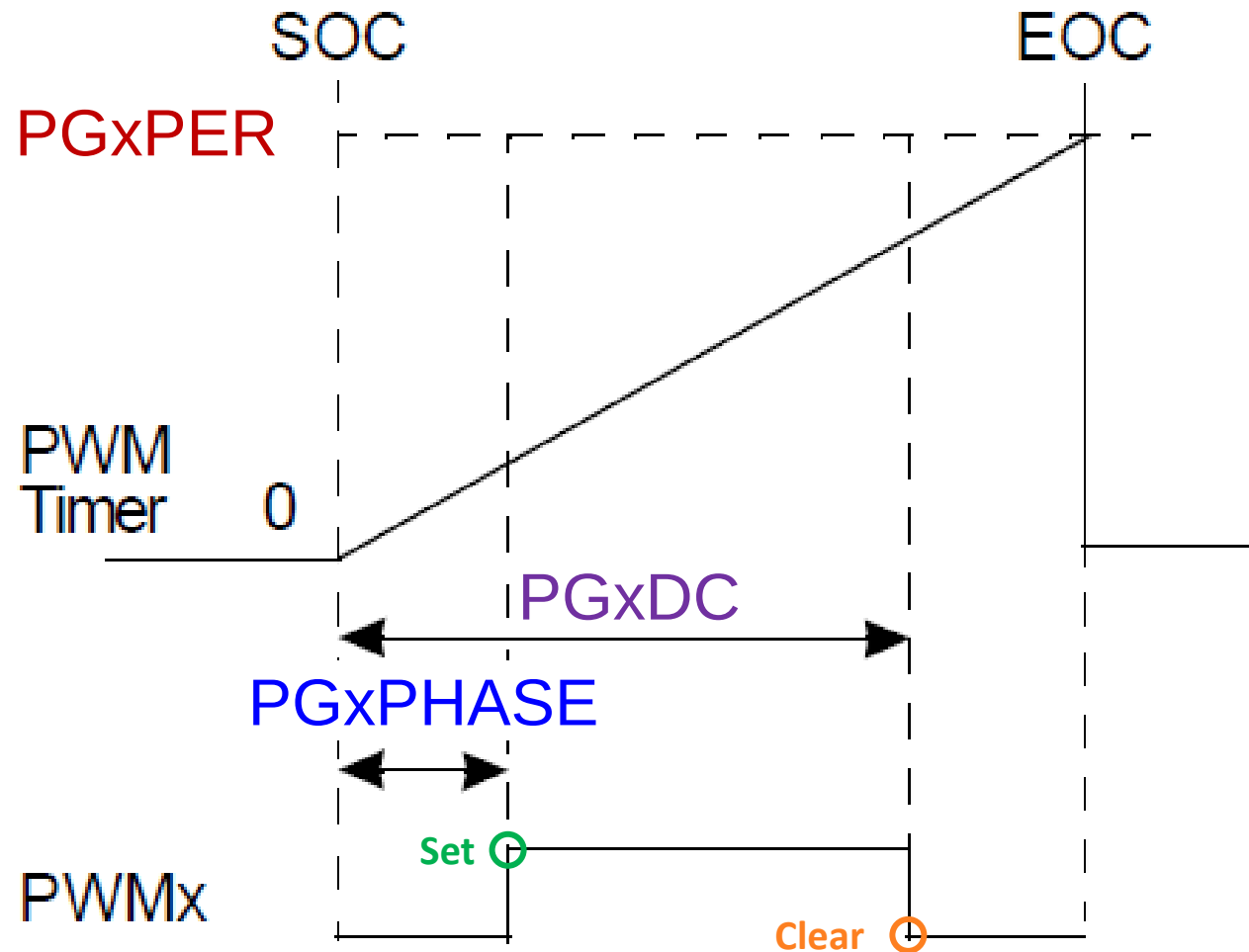
High Resolution PWM

- This flexible module provides features to support many types of Motor Control (MC) and Power Control (PC).
- dsPIC33CK256MP506 include one PWM module provide maximum eight pairs of PWM outputs available.
- All PWM has dedicated pins except PWM4, PWM4 could has flexibility of Peripheral Select Pin(PPS).
- Support lot of operating modes:
 - **Independent Edge mode.**
 - Variable Phase PWM mode.
 - Center-Aligned mode.
 - Double Update Center-Aligned mode.
 - Dual Edge Center-Aligned mode.
 - Dual PWM mode.
- At this section, we focus on **Independent Edge Mode.**



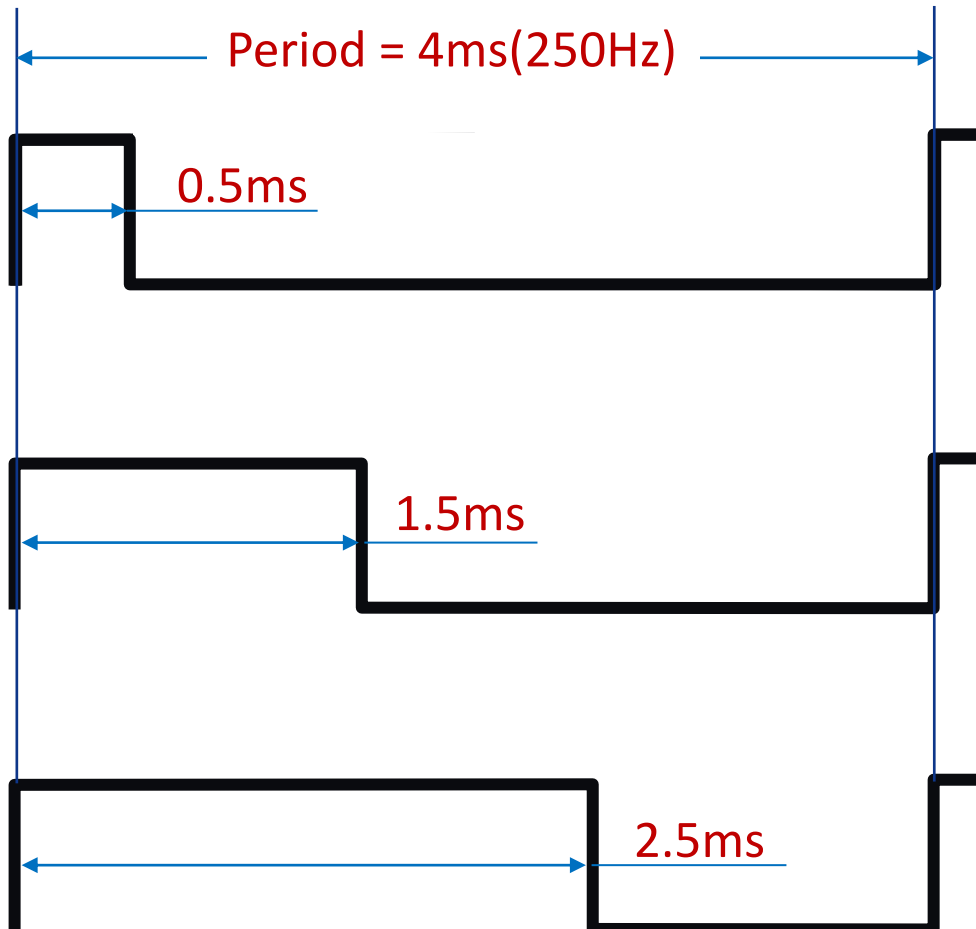
Independent Edge Mode of High Resolution PWM

- For **Independent Edge Mode**, the period time (T) has controlled by the **PGxPER** register.
- PWMx to "Set" after shift of **PGxPHASE** and "Clear" after **PGxDC**.



Servo Motor angle control (SG90)

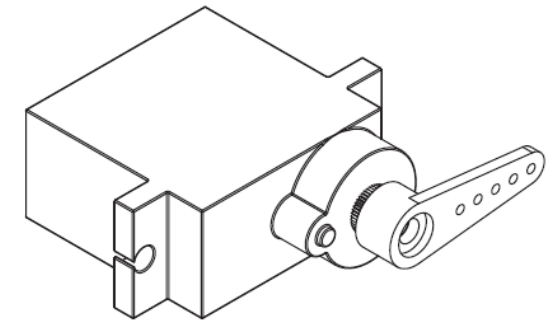
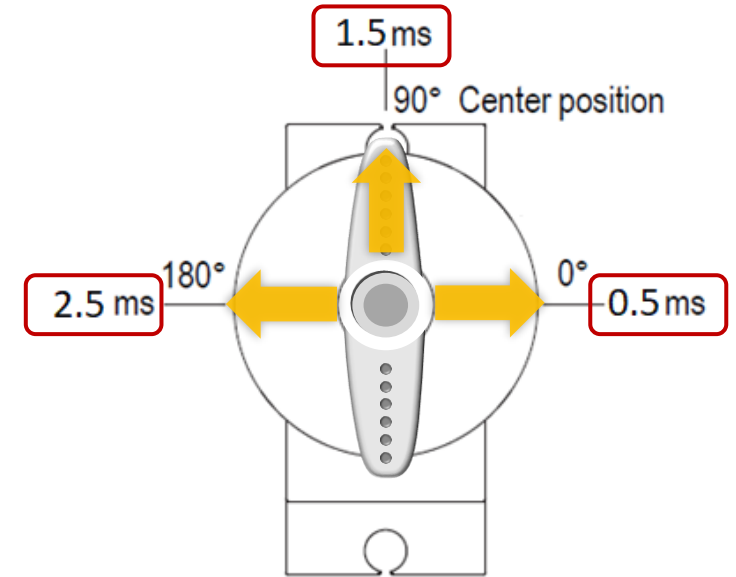
$$\text{Duty Cycle(\%)} : \frac{\text{Servo Motor Angle Duty}}{\text{Period}} \times 100\%$$



$$\frac{0.5ms}{4ms} = 12.5\% \quad (0^\circ)$$

$$\frac{1.5ms}{4ms} = 37.5\% \quad (90^\circ)$$

$$\frac{2.5ms}{4ms} = 62.5\% \quad (180^\circ)$$



- **Brown** - GND
- **Red** - VCC
- **Orange** - Signal

Lab9 PWM Output

Lab9 PWM Output

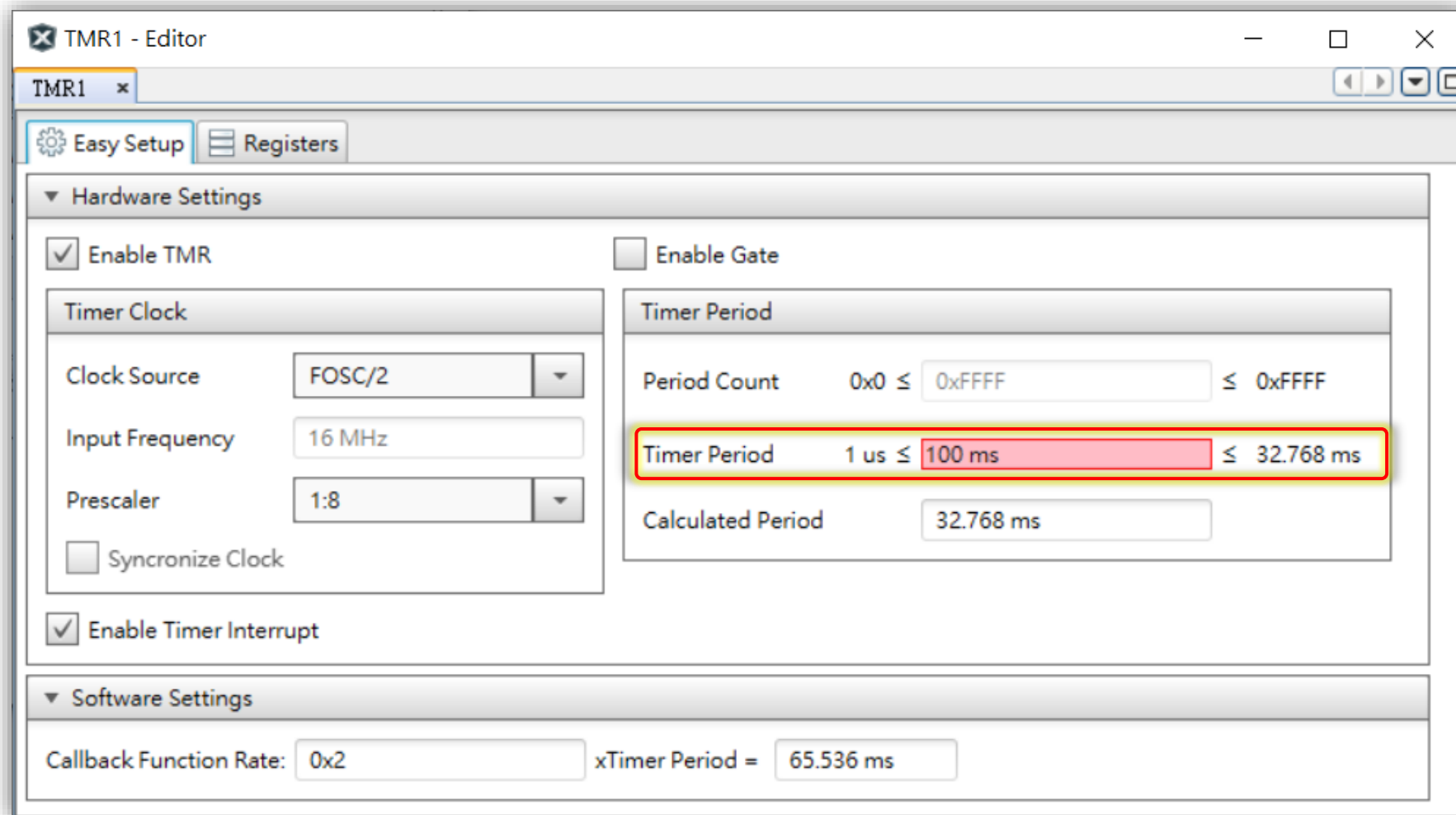
- Base on current project or open .\Lab9_xxx.X to do exercises.
- Try enable and initial **High Resolution PWM Generator 1** and **PWM Generator 2** to **Independent Edge Mode**, then given proper duty of PWM to control the **rotation angle** of **Servo Motor**.
- Modify **Timer1(TMR1)** period from 200ms to 50ms (**25ms** x 2).
- **High Resolution PWM-PG1** and **PWM-PG2** module clock source select to **16MHz**, configure PWM frequency to **250Hz(4ms period)** and Duty cycle to **12.5%** as initial.

Let's go!

Lab9 PWM Output

Step 1

- Set Prescaler : **1:64** > **1:8**.



The screenshot shows the 'TMR1 - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing the following configuration:

- ☒ Enable TMR
- ☐ Enable Gate
- Timer Clock**
 - Clock Source: FOSC/2
 - Input Frequency: 16 MHz
 - Prescaler: 1:8
 - ☐ Synchronize Clock
- ☒ Enable Timer Interrupt

The 'Timer Period' section is also expanded, showing:

- Period Count: 0x0 ≤ 0xFFFF ≤ 0xFFFF
- Timer Period: 1 us ≤ 100 ms ≤ 32.768 ms** (highlighted with a red box)
- Calculated Period: 32.768 ms

The 'Software Settings' section is also expanded, showing:

- Callback Function Rate: 0x2
- xTimer Period = 65.536 ms

Lab9 PWM Output

Step 2

- Set **Time Period** : **100ms > 25ms**.

TMR1 - Editor

TMR1 x

Easy Setup Registers

▼ Hardware Settings

☒ Enable TMR ☐ Enable Gate

Timer Clock

Clock Source: FOSC/2

Input Frequency: 16 MHz

Prescaler: 1:8

☐ Synchronize Clock

Timer Period

Period Count: $0x0 \leq 0xC34F \leq 0xFFFF$

Timer Period: $1 \mu s \leq 25 \text{ ms} \leq 32.768 \text{ ms}$

Calculated Period: 25 ms

☒ Enable Timer Interrupt

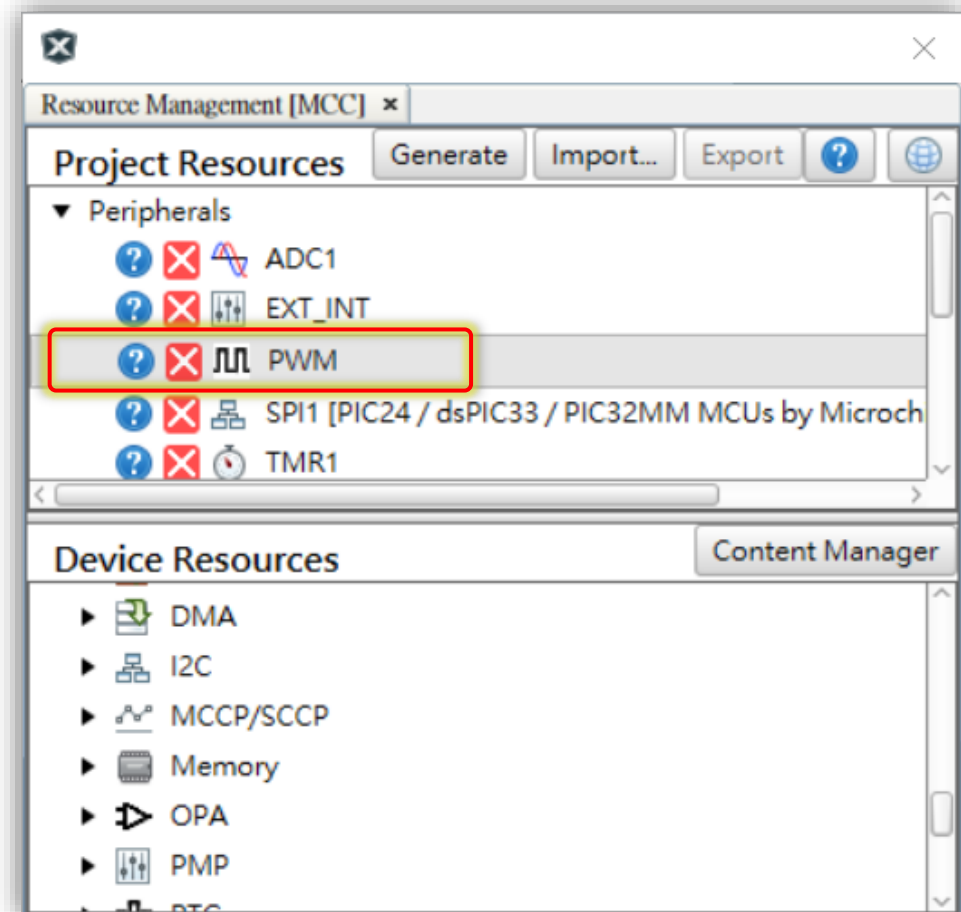
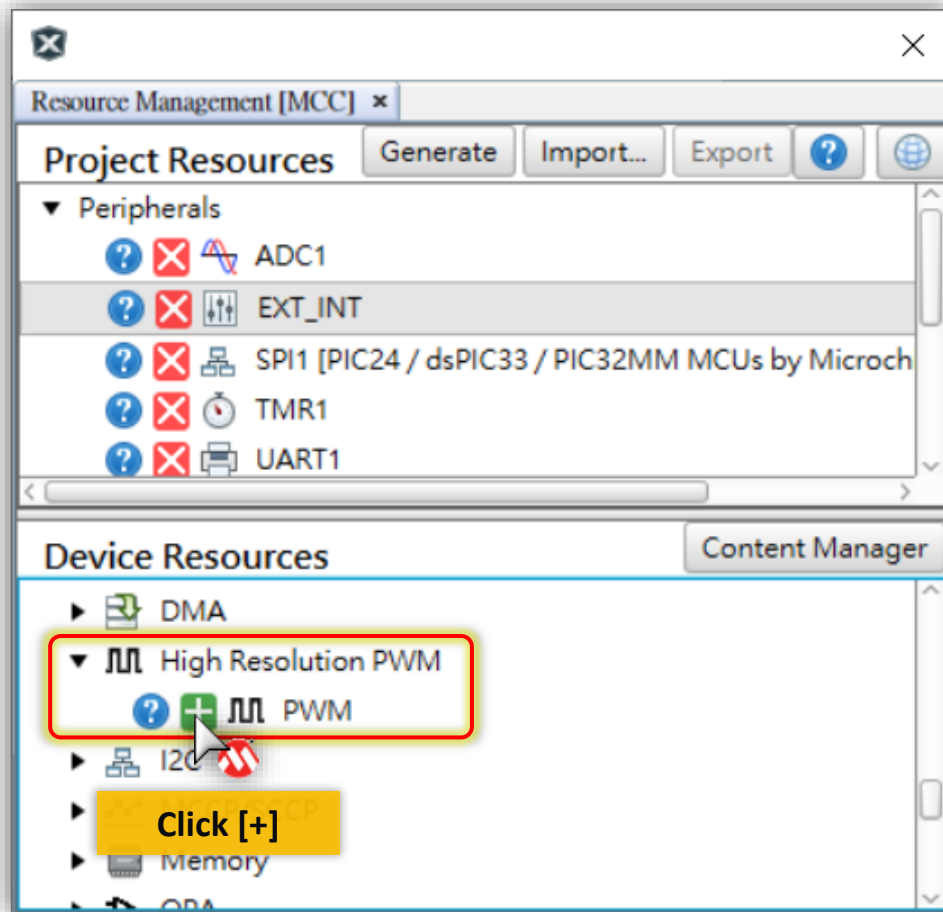
▼ Software Settings

Callback Function Rate: 0x2 xTimer Period = 50 ms

Lab9 PWM Output

Step 3

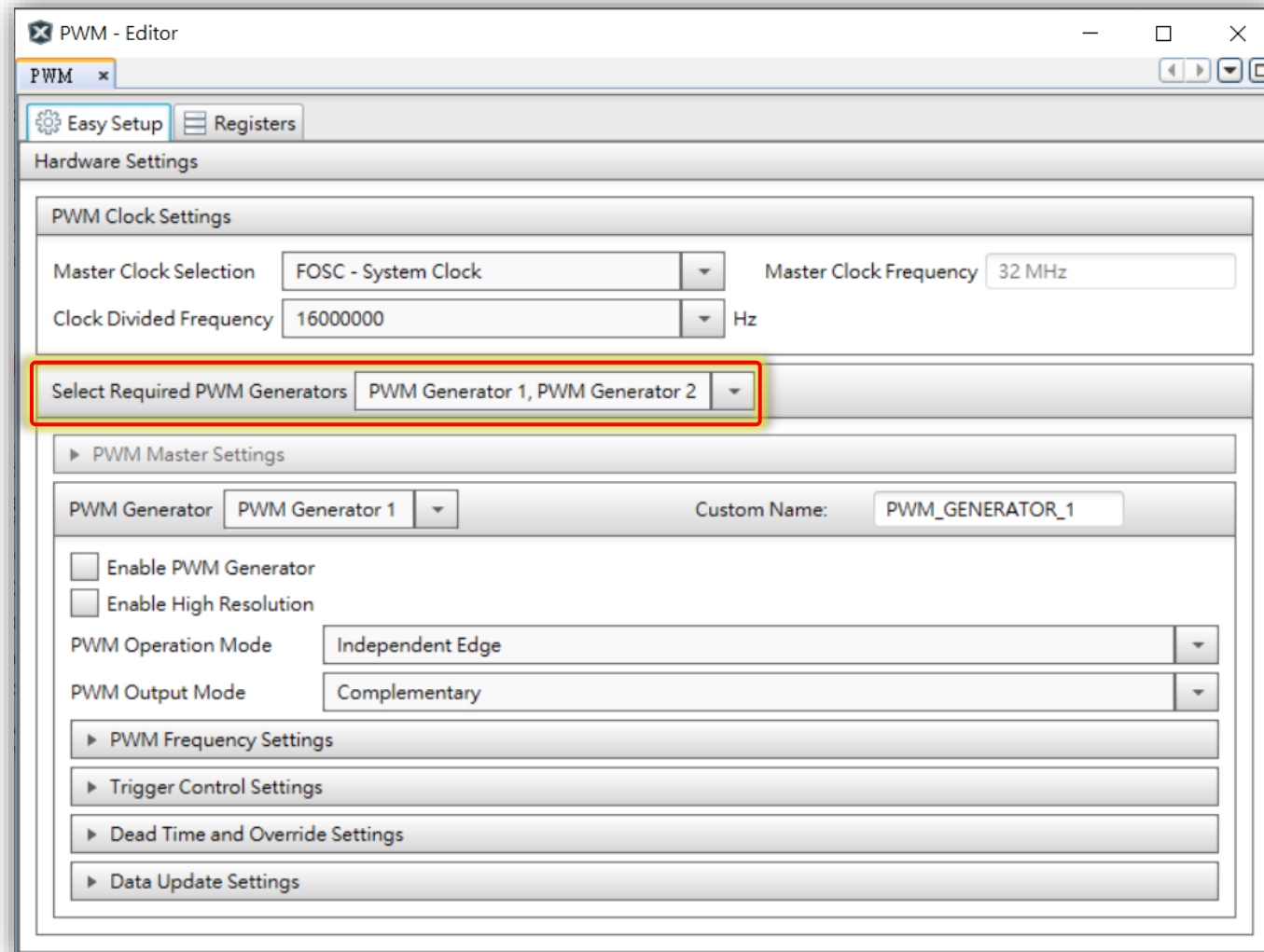
- Click [+] to add **High Resolution PWM** > **PWM** module.



Lab9 PWM Output

Step 4

- Set **PWM Generators** : **PWM Generator 1** and **PWM Generator 2**.

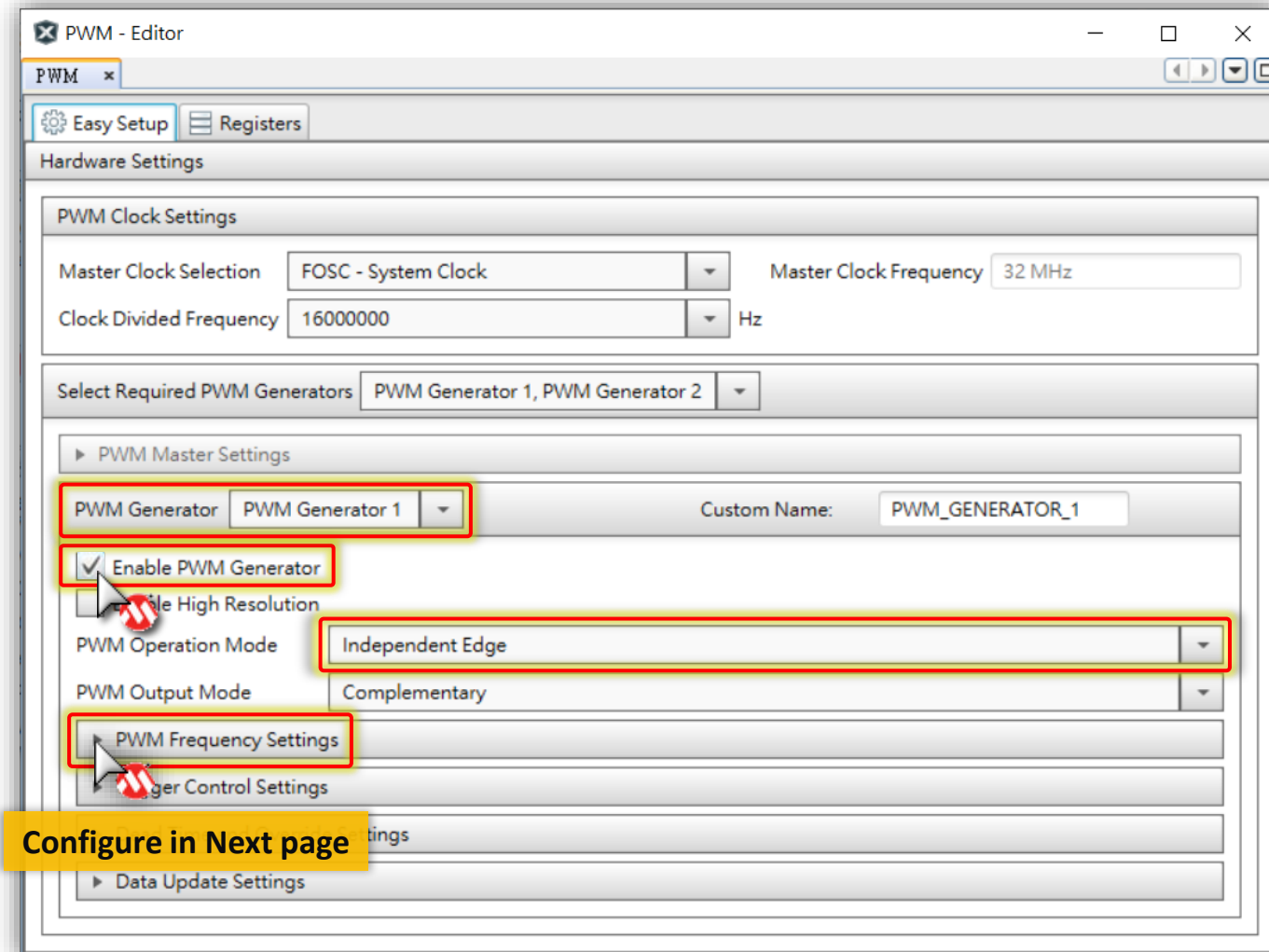


The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing 'PWM Clock Settings' with 'Master Clock Selection' set to 'FOSC - System Clock' and 'Master Clock Frequency' at '32 MHz'. The 'Clock Divided Frequency' is set to '16000000 Hz'. A red rectangle highlights the 'Select Required PWM Generators' dropdown menu, which is currently set to 'PWM Generator 1, PWM Generator 2'. Below this, the 'PWM Master Settings' section is expanded, showing 'PWM Generator' set to 'PWM Generator 1' and 'Custom Name' as 'PWM_GENERATOR_1'. The 'Enable PWM Generator' and 'Enable High Resolution' checkboxes are unchecked. The 'PWM Operation Mode' is set to 'Independent Edge' and the 'PWM Output Mode' is set to 'Complementary'. At the bottom, there are expandable sections for 'PWM Frequency Settings', 'Trigger Control Settings', 'Dead Time and Override Settings', and 'Data Update Settings'.

Lab9 PWM Output

Step 4

- Set **Enable PWM Generator 1** and Confirm **PWM Mode : Independent Edge**.



Lab9 PWM Output

Step 5

- Set **PWM1 Clock Selection** : **32000000 > 16000000**.

The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'PWM Master Settings' section is expanded, showing 'PWM Generator 1' selected in the 'PWM Generator' dropdown. The 'Custom Name' is 'PWM_GENERATOR_1'. The 'Enable PWM Generator' checkbox is checked, and 'Enable High Resolution' is unchecked. The 'PWM Operation Mode' is 'Independent Edge' and the 'PWM Output Mode' is 'Complementary'. The 'PWM Frequency Settings' section is expanded, showing 'PWM Input Clock Selection' set to '16000000 Hz'. Below this, the 'Period' section is expanded, showing 'Use Master Period' unchecked. The 'Requested Frequency' is '244.14 Hz', the 'Calculated Frequency' is '941.17647 kHz', the 'Requested Period' is '1.0625 us', and the 'Calculated Period' is '1.0625 us'. The 'Duty Cycle' section is expanded, showing 'Use Master Duty Cycle' unchecked. The 'PWM Duty Cycle' is '0 %'.

Parameter	Value
PWM Generator	PWM Generator 1
Custom Name	PWM_GENERATOR_1
Enable PWM Generator	Checked
Enable High Resolution	Unchecked
PWM Operation Mode	Independent Edge
PWM Output Mode	Complementary
PWM Input Clock Selection	16000000 Hz
Requested Frequency	244.14 Hz
Calculated Frequency	941.17647 kHz
Requested Period	1.0625 us
Calculated Period	1.0625 us
PWM Duty Cycle	0 %

Lab9 PWM Output

Step 6

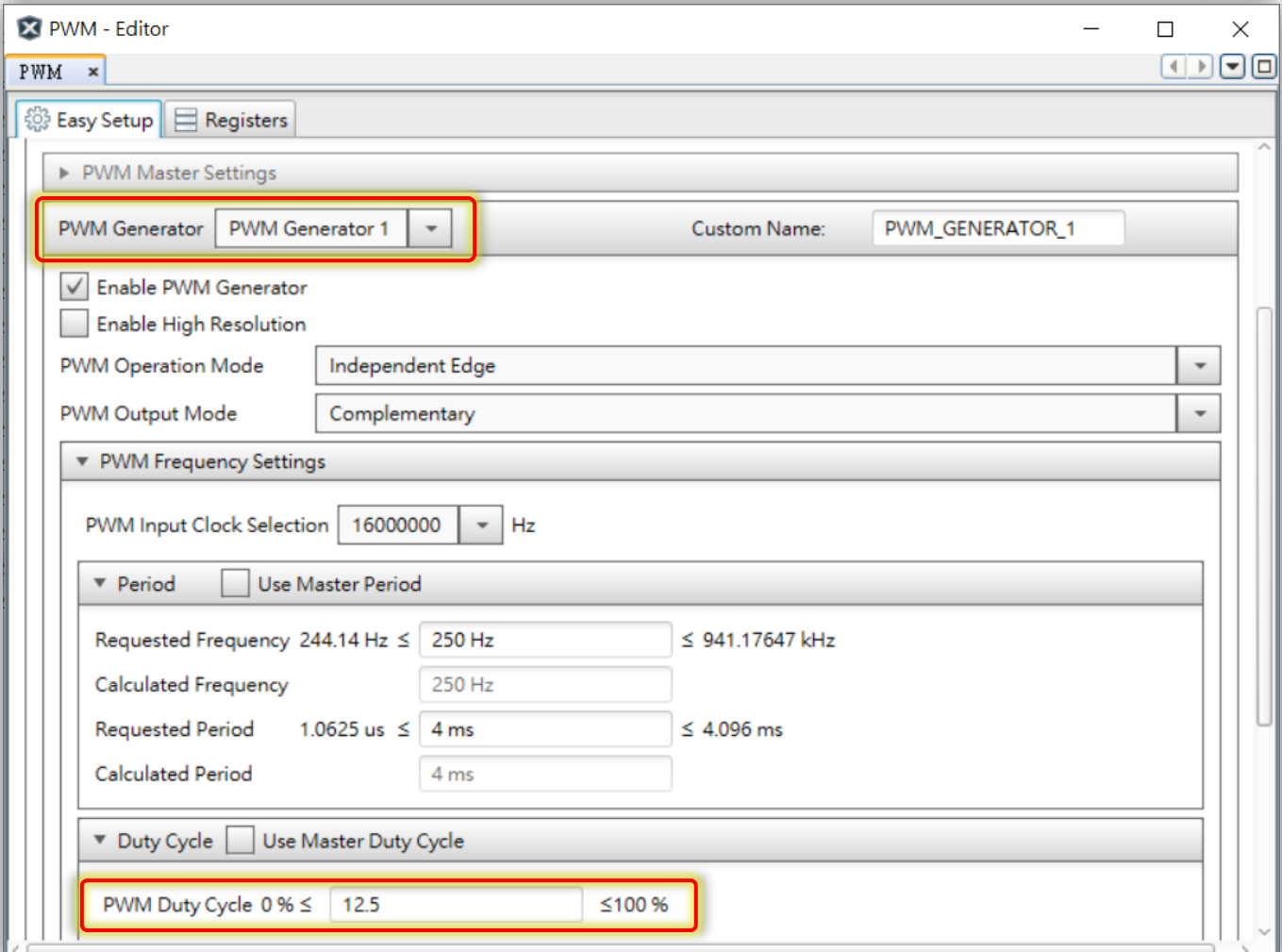
- Set **PWM1 Frequency** : **941.17647kHz** > 250Hz.

The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'PWM Master Settings' section includes a dropdown for 'PWM Generator' set to 'PWM Generator 1', a 'Custom Name' field with 'PWM_GENERATOR_1', and checkboxes for 'Enable PWM Generator' (checked) and 'Enable High Resolution' (unchecked). The 'PWM Operation Mode' is set to 'Independent Edge' and the 'PWM Output Mode' is set to 'Complementary'. The 'PWM Frequency Settings' section shows 'PWM Input Clock Selection' as '16000000 Hz'. Under the 'Period' section, 'Use Master Period' is unchecked, and the 'Requested Frequency' is set to '244.14 Hz ≤ 250 Hz ≤ 941.17647 kHz'. The 'Calculated Frequency' is '250 Hz', the 'Requested Period' is '1.0625 us ≤ 4 ms ≤ 4.096 ms', and the 'Calculated Period' is '4 ms'. Under the 'Duty Cycle' section, 'Use Master Duty Cycle' is unchecked, and the 'PWM Duty Cycle' is set to '0 % ≤ 0 ≤ 100 %'.

Lab9 PWM Output

Step 7

- Set PWM1 Duty Cycle : 0 > 12.5.



The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'PWM Master Settings' section is expanded, and the 'PWM Generator' dropdown is highlighted with a red box. The 'Custom Name' field is set to 'PWM_GENERATOR_1'. Below this, the 'Enable PWM Generator' checkbox is checked, and the 'Enable High Resolution' checkbox is unchecked. The 'PWM Operation Mode' is set to 'Independent Edge', and the 'PWM Output Mode' is set to 'Complementary'. The 'PWM Frequency Settings' section is expanded, showing the 'PWM Input Clock Selection' set to '16000000 Hz'. The 'Period' section is expanded, showing the 'Use Master Period' checkbox is unchecked. The 'Requested Frequency' is set to '244.14 Hz', and the 'Calculated Frequency' is '250 Hz'. The 'Requested Period' is '1.0625 us', and the 'Calculated Period' is '4 ms'. The 'Duty Cycle' section is expanded, showing the 'Use Master Duty Cycle' checkbox is unchecked. The 'PWM Duty Cycle' is set to '0 %', and the 'Calculated Period' is '4 ms'. The 'PWM Duty Cycle' field is highlighted with a red box, showing a value of 12.5.

PWM - Editor

PWM x

Easy Setup Registers

PWM Master Settings

PWM Generator PWM Generator 1 Custom Name: PWM_GENERATOR_1

☒ Enable PWM Generator
☐ Enable High Resolution

PWM Operation Mode Independent Edge

PWM Output Mode Complementary

PWM Frequency Settings

PWM Input Clock Selection 16000000 Hz

Period ☐ Use Master Period

Requested Frequency 244.14 Hz ≤ 250 Hz ≤ 941.17647 kHz

Calculated Frequency 250 Hz

Requested Period 1.0625 us ≤ 4 ms ≤ 4.096 ms

Calculated Period 4 ms

Duty Cycle ☐ Use Master Duty Cycle

PWM Duty Cycle 0 % ≤ 12.5 ≤ 100 %

Lab9 PWM Output

Step 8

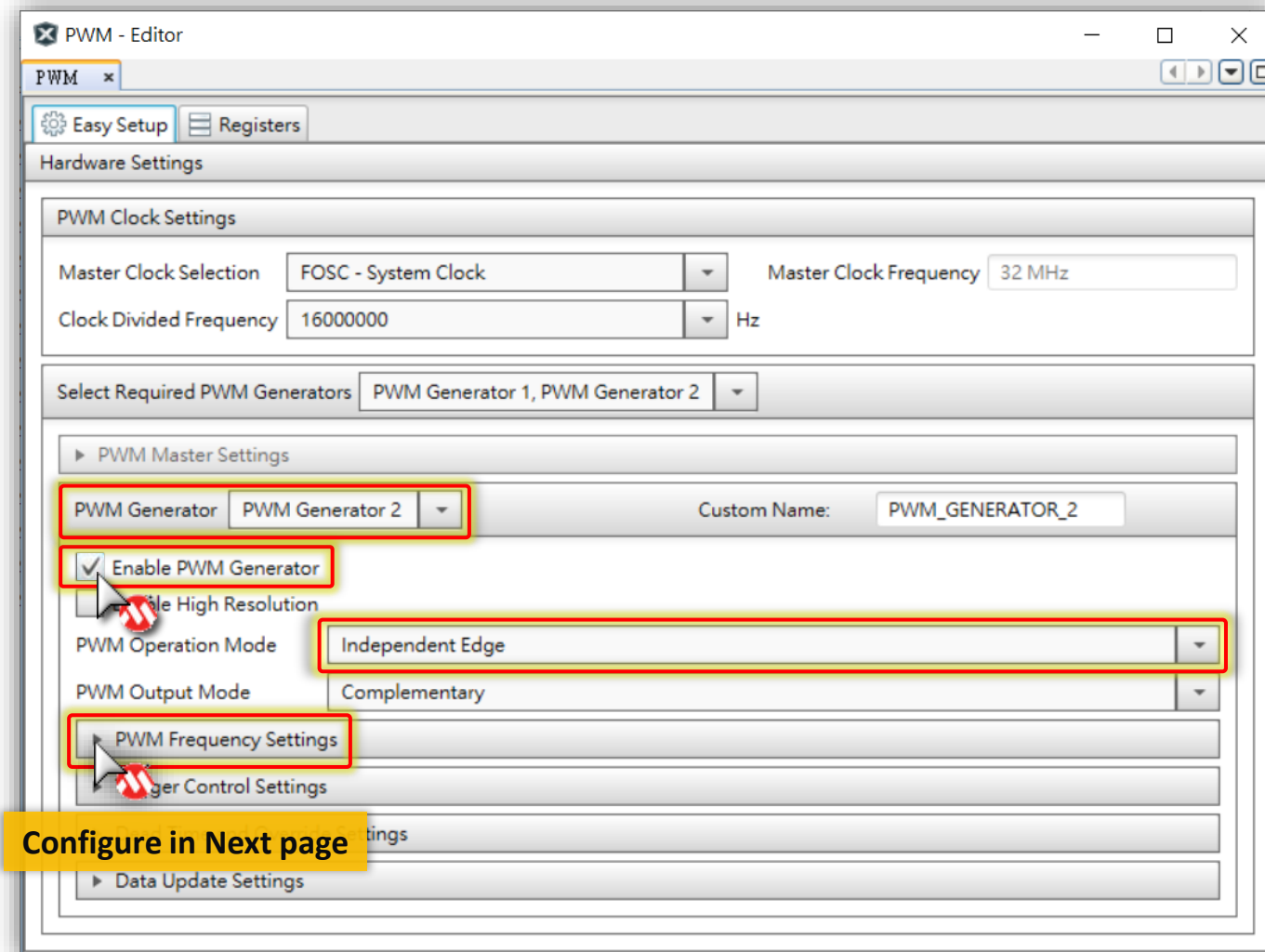
- Set **PWM Generator** : **PWM Generator 1 > PWM Generator 2.**

The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing 'PWM Clock Settings' with 'Master Clock Selection' set to 'FOSC - System Clock' and 'Master Clock Frequency' at '32 MHz'. The 'Clock Divided Frequency' is set to '16000000 Hz'. Below this, 'Select Required PWM Generators' is set to 'PWM Generator 1, PWM Generator 2'. The 'PWM Master Settings' section is expanded, and the 'PWM Generator' dropdown is highlighted with a red box, showing 'PWM Generator 2' selected. The 'Custom Name' is 'PWM_GENERATOR_2'. Other settings include 'Enable PWM Generator' (unchecked), 'Enable High Resolution' (unchecked), 'PWM Operation Mode' set to 'Independent Edge', and 'PWM Output Mode' set to 'Complementary'. At the bottom, there are expandable sections for 'PWM Frequency Settings', 'Trigger Control Settings', 'Dead Time and Override Settings', and 'Data Update Settings'.

Lab9 PWM Output

Step 9

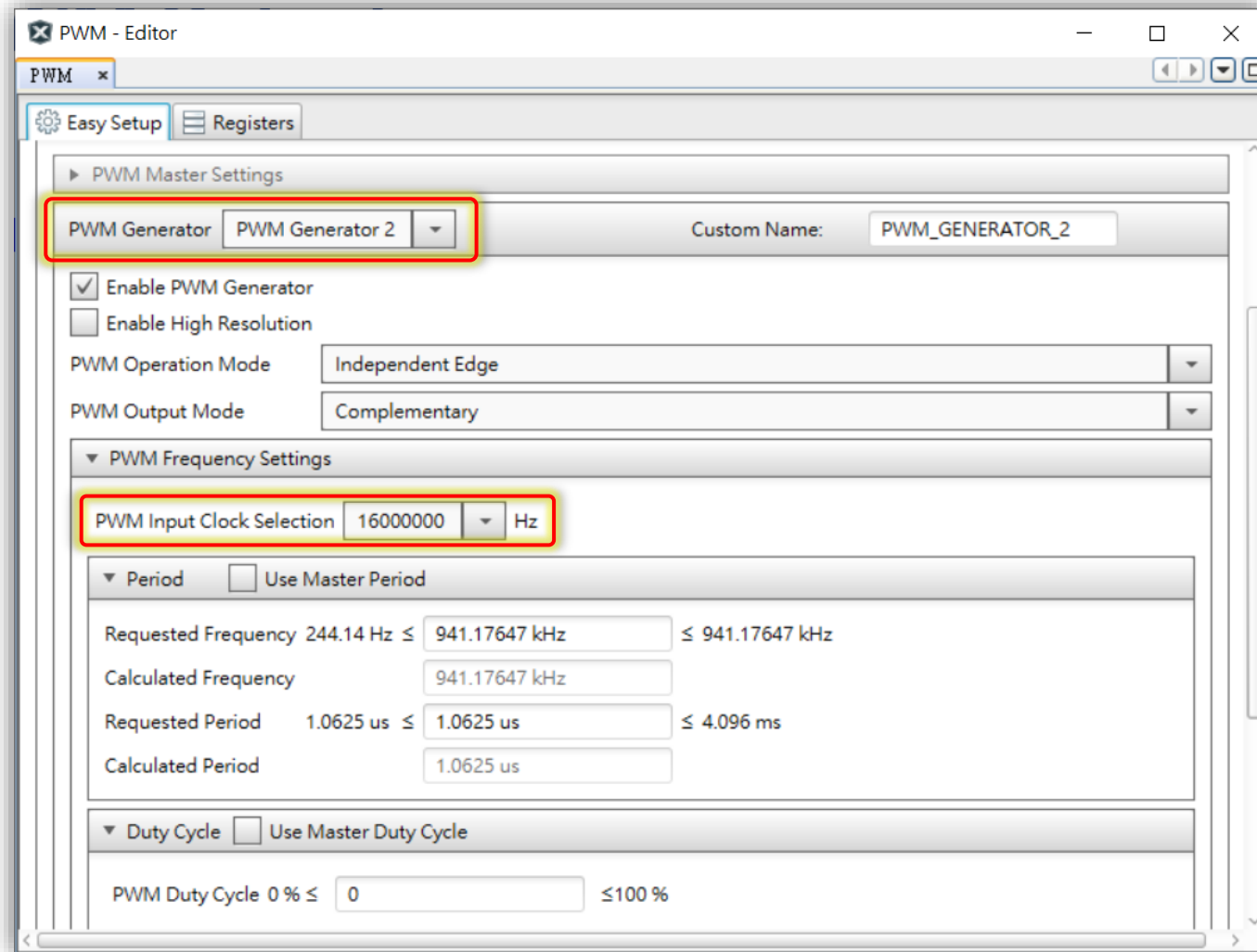
- Set **Enable PWM Generator 2** and Confirm **PWM Mode : Independent Edge**.



Lab9 PWM Output

Step 10

- Set **PWM2 Clock Selection** : **32000000 > 16000000**.



The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'PWM Master Settings' section includes a dropdown for 'PWM Generator' set to 'PWM Generator 2', a 'Custom Name' field with 'PWM_GENERATOR_2', and checkboxes for 'Enable PWM Generator' (checked) and 'Enable High Resolution' (unchecked). The 'PWM Operation Mode' is set to 'Independent Edge' and the 'PWM Output Mode' is set to 'Complementary'. The 'PWM Frequency Settings' section shows 'PWM Input Clock Selection' set to '16000000 Hz'. Below this, the 'Period' section shows 'Use Master Period' unchecked, with 'Requested Frequency' and 'Calculated Frequency' both at 941.17647 kHz, and 'Requested Period' and 'Calculated Period' both at 1.0625 us. The 'Duty Cycle' section shows 'Use Master Duty Cycle' unchecked, with 'PWM Duty Cycle' set to 0%.

PWM - Editor

PWM x

Easy Setup Registers

PWM Master Settings

PWM Generator PWM Generator 2 Custom Name: PWM_GENERATOR_2

☒ Enable PWM Generator
☐ Enable High Resolution

PWM Operation Mode Independent Edge

PWM Output Mode Complementary

PWM Frequency Settings

PWM Input Clock Selection 16000000 Hz

Period ☐ Use Master Period

Requested Frequency 244.14 Hz ≤ 941.17647 kHz ≤ 941.17647 kHz

Calculated Frequency 941.17647 kHz

Requested Period 1.0625 us ≤ 1.0625 us ≤ 4.096 ms

Calculated Period 1.0625 us

Duty Cycle ☐ Use Master Duty Cycle

PWM Duty Cycle 0 % ≤ 0 ≤ 100 %

Lab9 PWM Output

Step 11

- Set **PWM2 Frequency** : **941.17647kHz** > 250Hz.

The screenshot shows the 'PWM - Editor' window with the 'Easy Setup' tab selected. The 'PWM Master Settings' section is expanded, showing 'PWM Generator 2' selected in the 'PWM Generator' dropdown. The 'Custom Name' is 'PWM_GENERATOR_2'. The 'Enable PWM Generator' checkbox is checked. The 'PWM Operation Mode' is 'Independent Edge' and the 'PWM Output Mode' is 'Complementary'. The 'PWM Frequency Settings' section is expanded, showing 'PWM Input Clock Selection' as '16000000 Hz'. The 'Period' section is expanded, showing 'Requested Frequency' as '244.14 Hz ≤ 250 Hz ≤ 941.17647 kHz', which is highlighted with a red box. The 'Calculated Frequency' is '250 Hz'. The 'Requested Period' is '1.0625 us ≤ 4 ms ≤ 4.096 ms' and the 'Calculated Period' is '4 ms'. The 'Duty Cycle' section is expanded, showing 'PWM Duty Cycle 0 % ≤ 0 ≤ 100 %'.

Lab9 PWM Output

Step 12

- Set **PWM2 Duty Cycle** : **0 > 12.5**.

PWM - Editor

PWM x

Easy Setup Registers

PWM Master Settings

PWM Generator PWM Generator 2 Custom Name: PWM_GENERATOR_2

☒ Enable PWM Generator
☐ Enable High Resolution

PWM Operation Mode Independent Edge

PWM Output Mode Complementary

PWM Frequency Settings

PWM Input Clock Selection 16000000 Hz

Period ☐ Use Master Period

Requested Frequency 244.14 Hz ≤ 250 Hz ≤ 941.17647 kHz

Calculated Frequency 250 Hz

Requested Period 1.0625 us ≤ 4 ms ≤ 4.096 ms

Calculated Period 4 ms

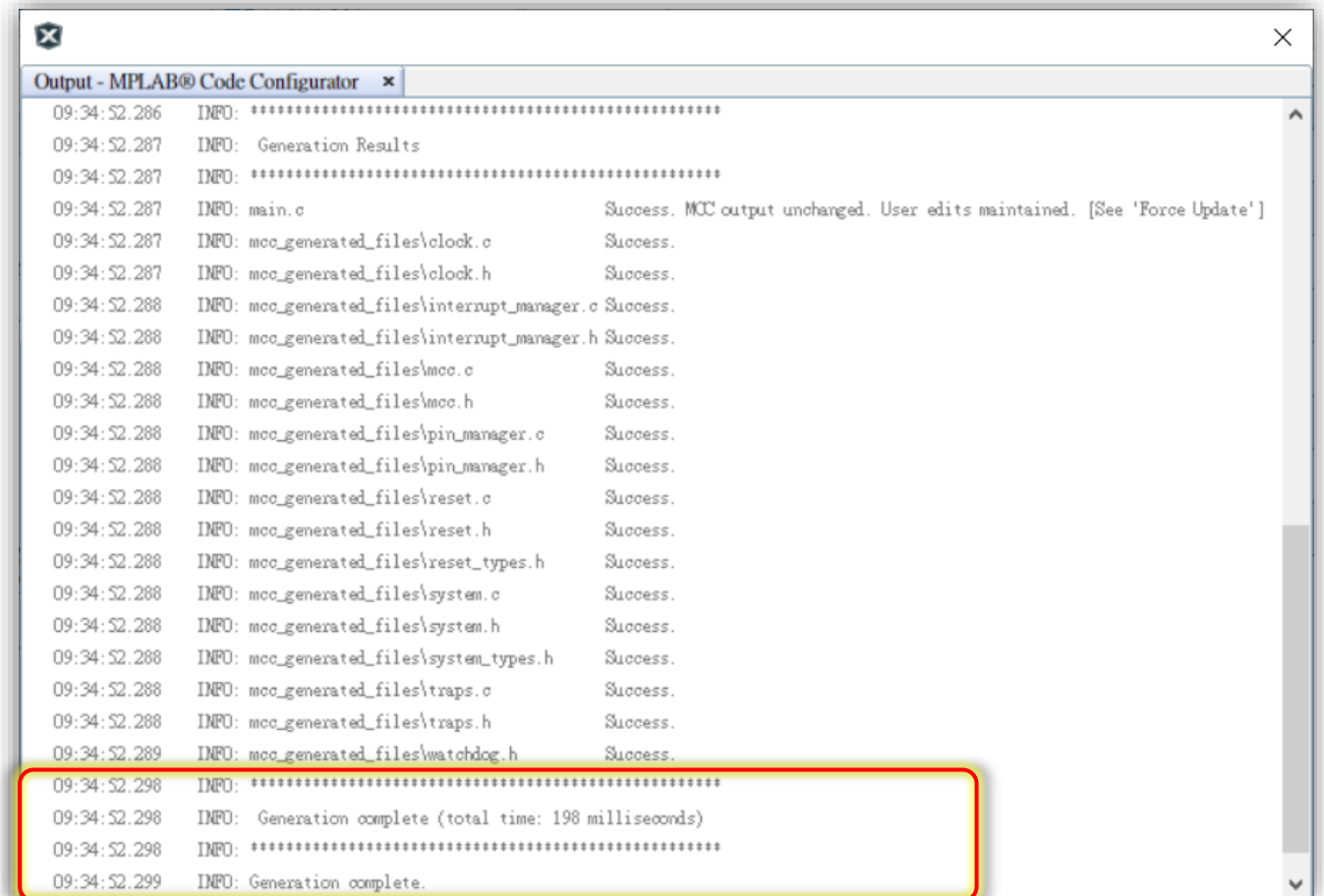
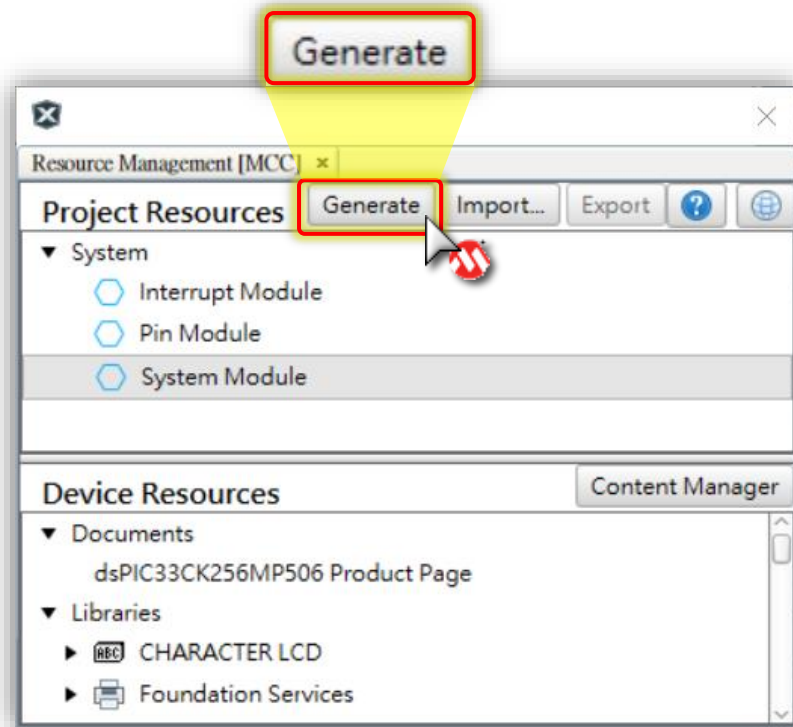
Duty Cycle ☐ Use Master Duty Cycle

PWM Duty Cycle 0 % ≤ 12.5 ≤ 100 %

Lab9 PWM Output

Step 13

- Click "**Generate**" Button to generate your code.



High Resolution PWM Functions of MCC

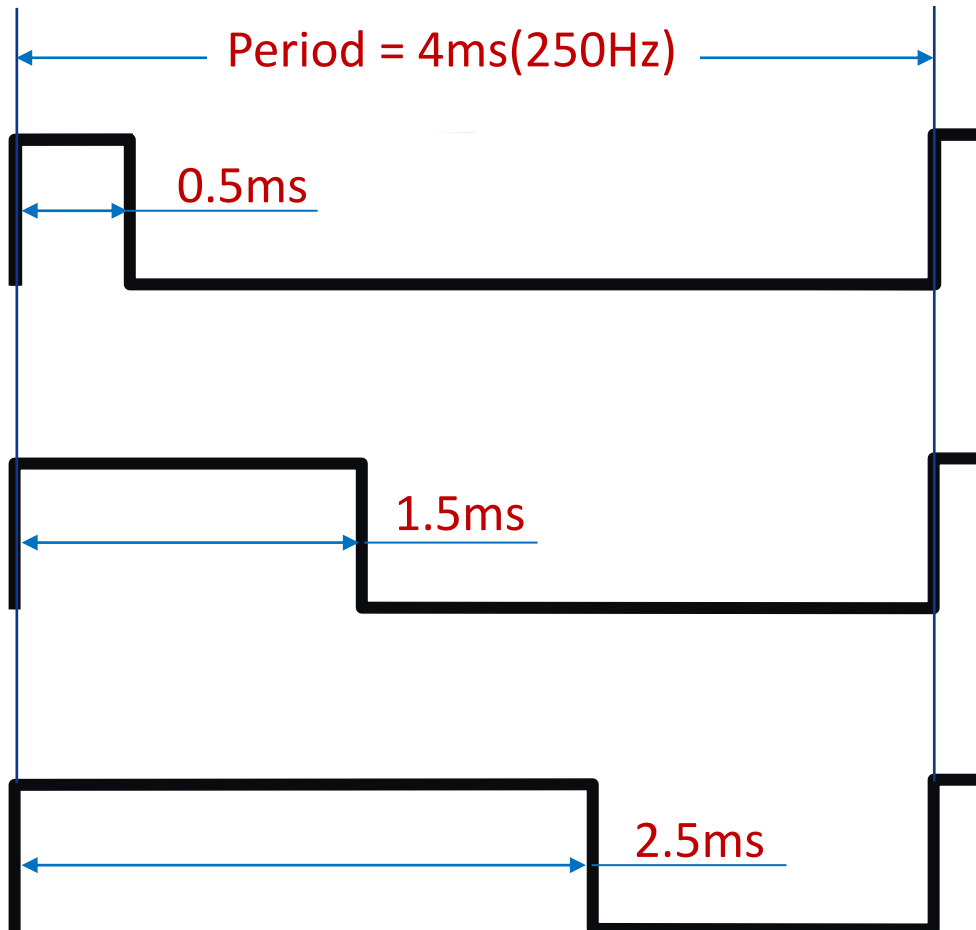
- MCC Provide below functions for High Resolution PWM:

Prototype definition : "pwm.h"

- void PWM_Initialize (void);
- void PWM_GeneratorEnable(PWM_GENERATOR genNum);
- void PWM_GeneratorDisable(PWM_GENERATOR genNum);
- void PWM_Enable(void);
- void PWM_Disable(void);
- PWM_MasterPeriodSet(uint16_t masterPeriod);
- void PWM_MasterDutyCycleSet(uint16_t masterDutyCycle);
- void PWM_MasterPhaseSet(uint16_t masterPhase);
- void PWM_PeriodSet(PWM_GENERATOR genNum,uint16_t period);
- void PWM_DutyCycleSet (PWM_GENERATOR genNum,uint16_t dutyCycle);
- void PWM_PhaseSet(PWM_GENERATOR genNum,uint16_t phase);
- void PWM_Generatorn_Callback(void);
- void PWM_SoftwareUpdateRequest(PWM_GENERATOR genNum)

Duty Cycle to Count Calculation

$$\text{Frequency} = \left(\frac{\text{Clock Source}}{\text{PWM Period}} \right) \Rightarrow \text{PWM Period} = \left(\frac{16\text{M(Hz)}}{250\text{(Hz)}} \right) = 64000(\text{Counts})$$



Duty Count

$$64,000 \times 12.5\% = \mathbf{8000} \quad (0^\circ)$$

$$64,000 \times 37.5\% = 24000 \quad (90^\circ)$$

$$64,000 \times 62.5\% = \mathbf{40000} \quad (180^\circ)$$

Lab9 PWM Output

Code Example

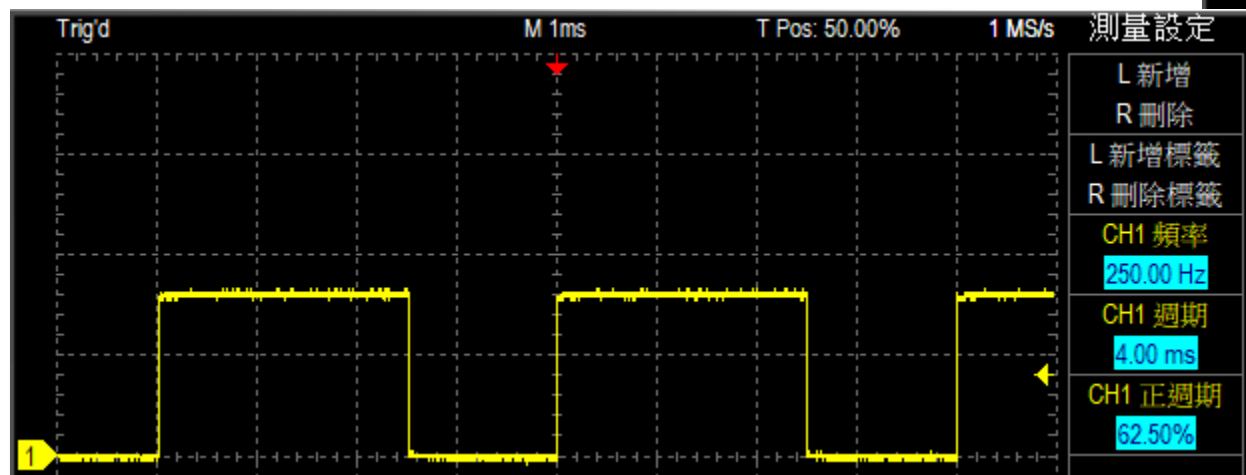
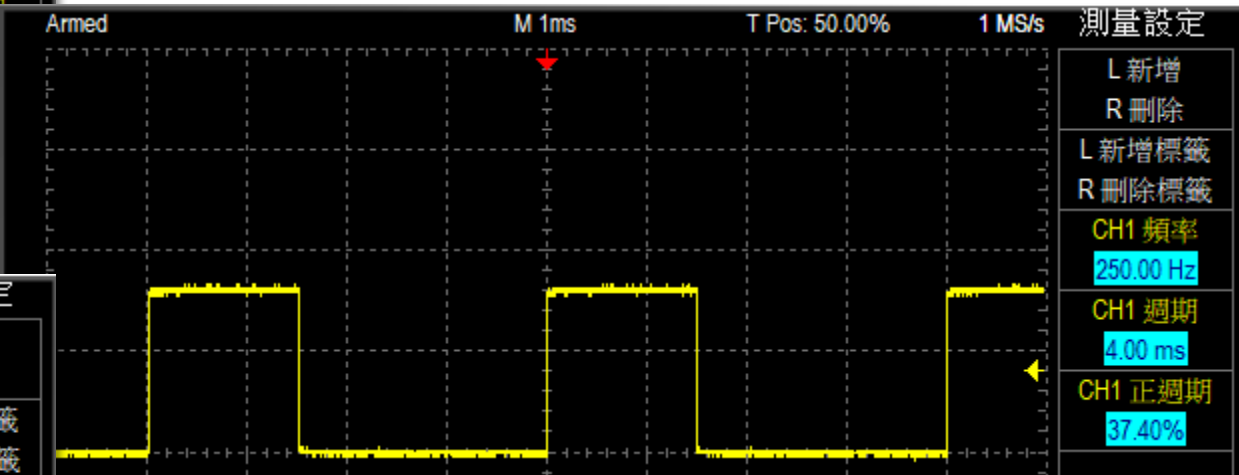
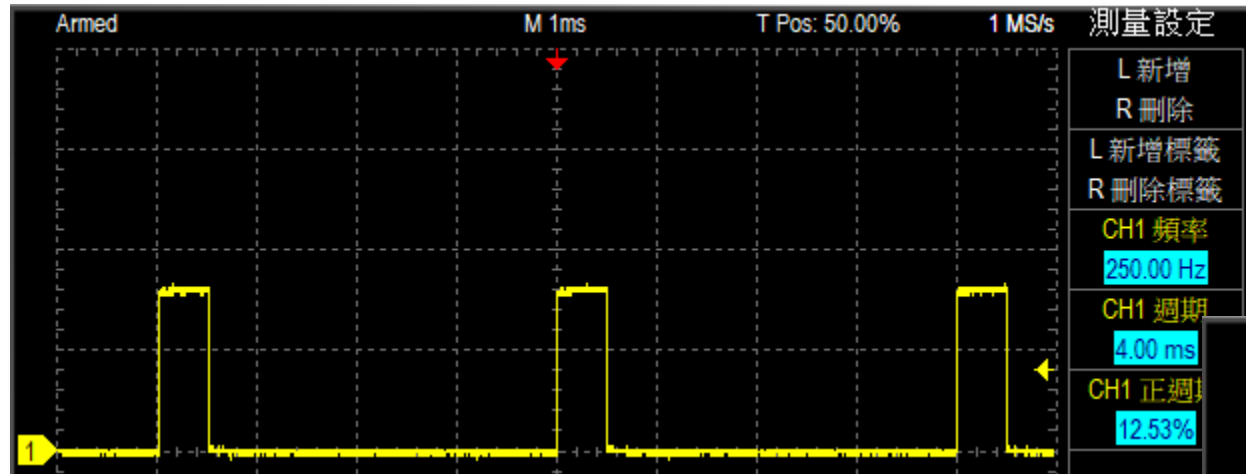
```
...
#include "mcc_generated_files/ext_int.h"
#include "mcc_generated_files/pwm.h"

volatile uint16_t ADC1_VR4_Buffer = 0;
uint16_t Servo_angle;

int main(void)
{
    ...
}
...
void ADC1_VR2_Callback (uint16_t adcVal)
{
    ADC1_VR2_Buffer = adcVal;
    PWM_DutyCycleSet(PWM_GENERATOR_2, 8000 + ((uint32_t) (10666 - 8000) * ADC1_VR2_Buffer / 4095));
    PWM_SoftwareUpdateRequest(PWM_GENERATOR_2);
}
void ADC1_VR4_Callback (uint16_t adcVal)
{
    ADC1_VR4_Buffer = adcVal;
    PWM_DutyCycleSet(PWM_GENERATOR_1, 8000 + ((uint32_t) (40000 - 8000) * ADC1_VR4_Buffer / 4095));
    PWM_SoftwareUpdateRequest(PWM_GENERATOR_1);
}
void DisplayTask( void )
{
    ...
    OLED_DrawString(60, 16, (char *) StringBuffer);
    Servo_angle = ((uint32_t) 15 * ADC1_VR2_Buffer / 4095);
    sprintf((char *) StringBuffer, "Servo_angle:%3d", Servo_angle);
    printf(" \033[6;1H%s", StringBuffer);
    OLED_DrawString(0, 24, (char *) StringBuffer);
    OLED_DrawString(0, 56, "By:Your Name");
    OLED_Update();
}
```

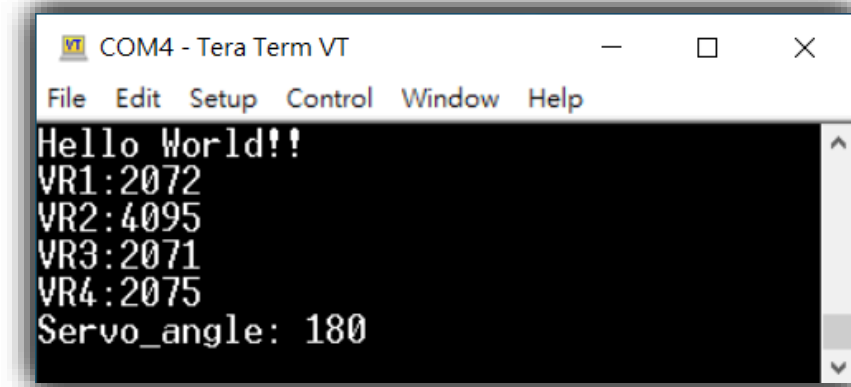
Lab9 PWM Output

Result (PWM_GENERATOR_1)

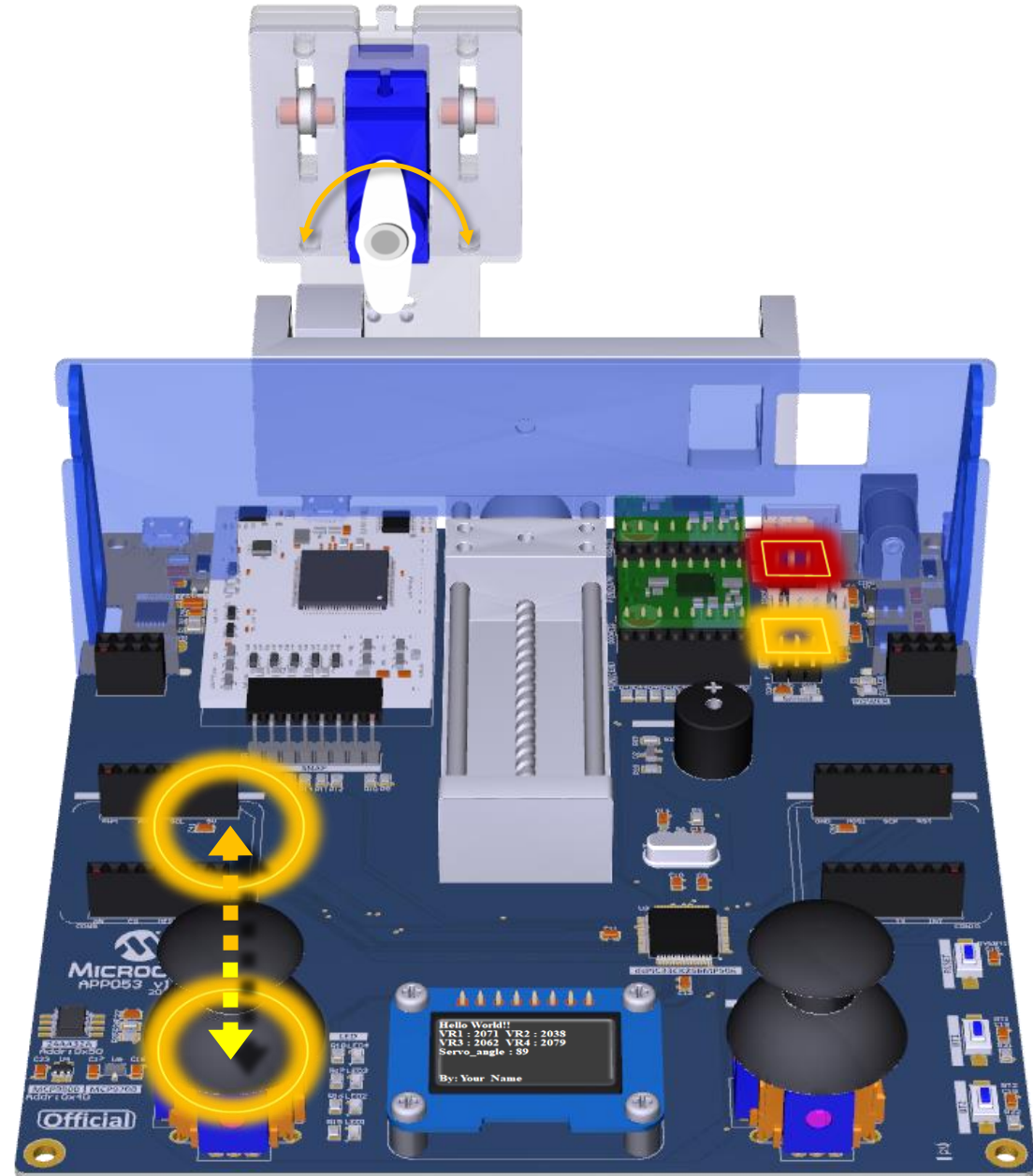


Lab9 PWM Output

Result (PWM_GENERATOR_1)



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
Hello World!!
VR1:2072
VR2:4095
VR3:2071
VR4:2075
Servo_angle: 180
```



MCCP & SCCP CAPTURE, COMPARE, PWM & TIMER MODULES

SCCP Module

- **SCCP can operate in one of three major modes:**
 - General Purpose Timer.
 - Input Capture.
 - Output Compare & PWM.
- **Output Compare/PWM Modes:**
 - Single Edge and Dual Edge Compare modes.
 - Center-Aligned Compare mode.
 - Variable Frequency Pulse mode.
 - External Input mode.
- **dsPIC33CK256MP506 include 8 SCCP and 1 MCCP.**
 - SCCP -> 1 x PWM Channel.
 - MCCP -> 6 x PWM Channels.
- **In this section, we focus on Output Compare & PWM mode.**

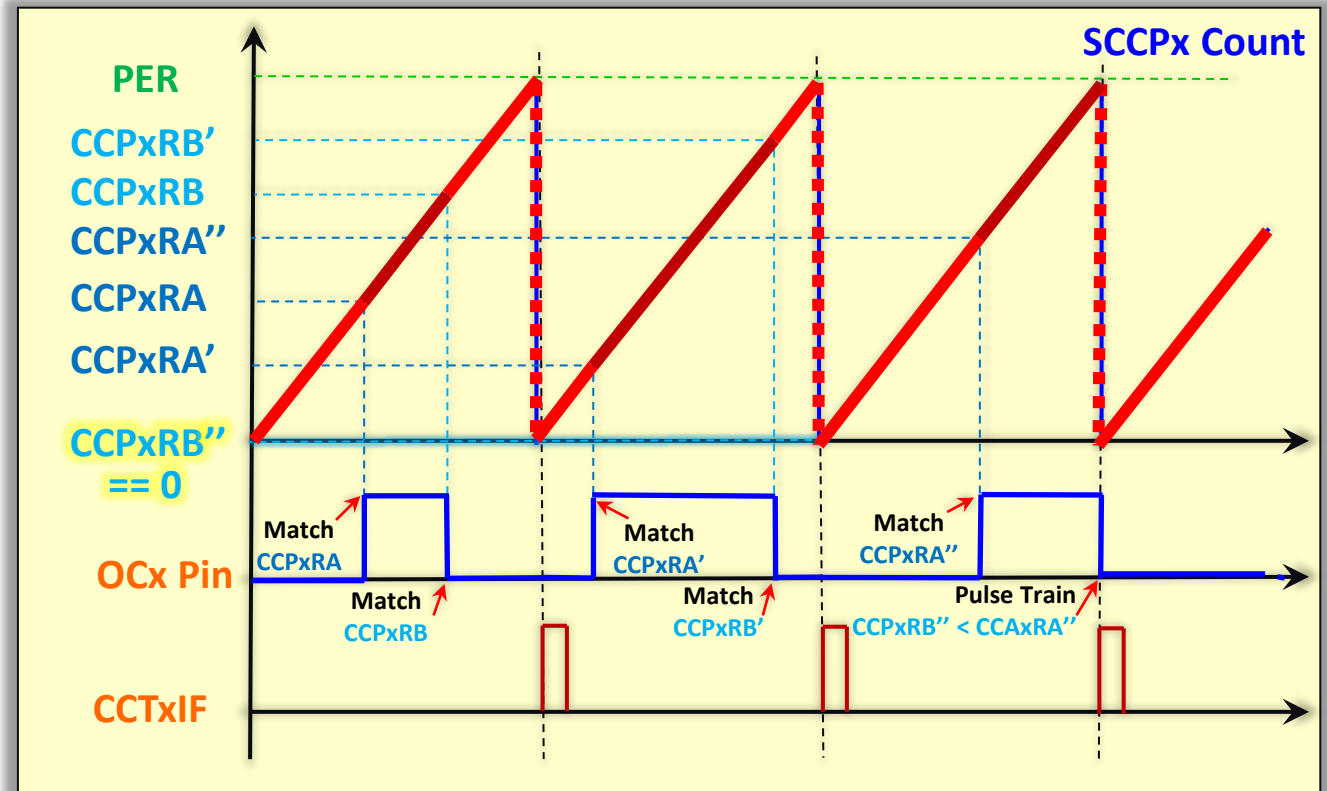
Output Compare & PWM Mode (Dual Edge Compare PWM)

- An example to describe Counter and Waveform output of PWM.
- For **Dual Edge Compare PWM** operation mode, PWM Period Time (T) has controlled through the PER register.

OCx Pin will "Set" while Counter match to **CCPxRA** and "Clear" if match **CCPxRB**.

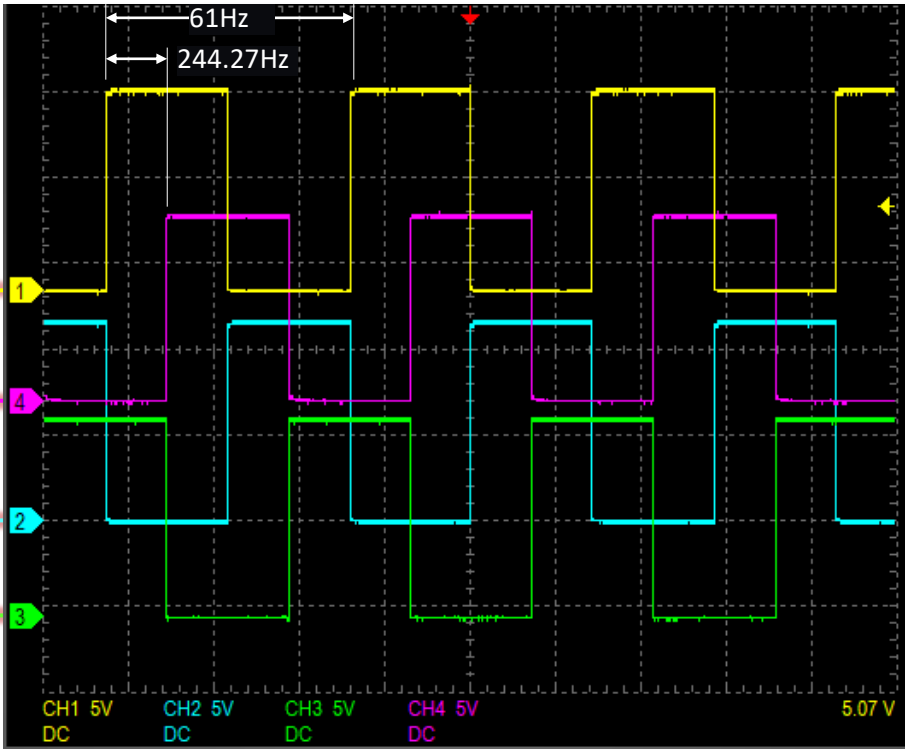
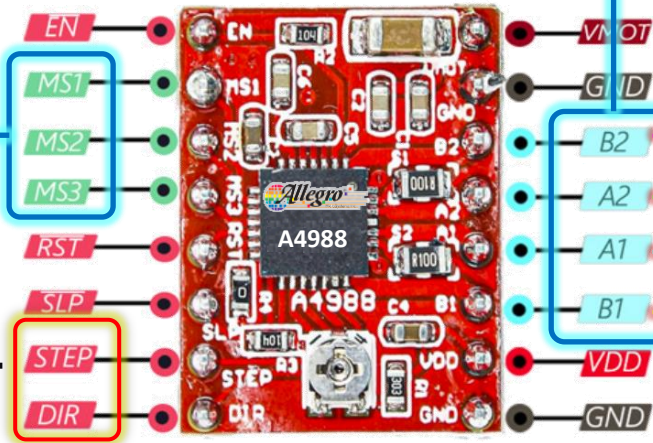
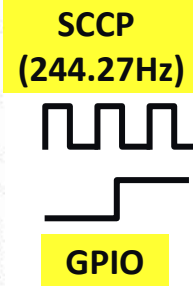
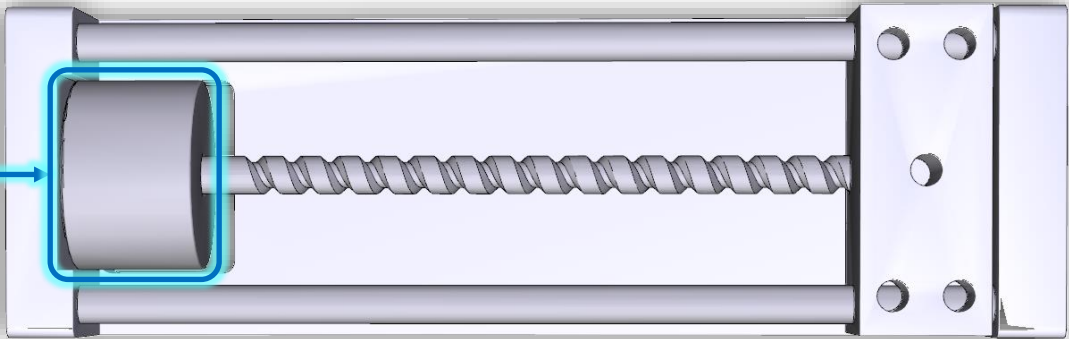
Rule

1. SCCPx Counter == CCPxRA
OCx Set to high
2. SCCPx Counter == CCPxRB
OCx Clear to low
3. If CCPxRB < CCPxRA
Pulse Train (CCPxRB ignore)
4. SCCPx Counter == PER
Counter Clear and Set CCTxIF



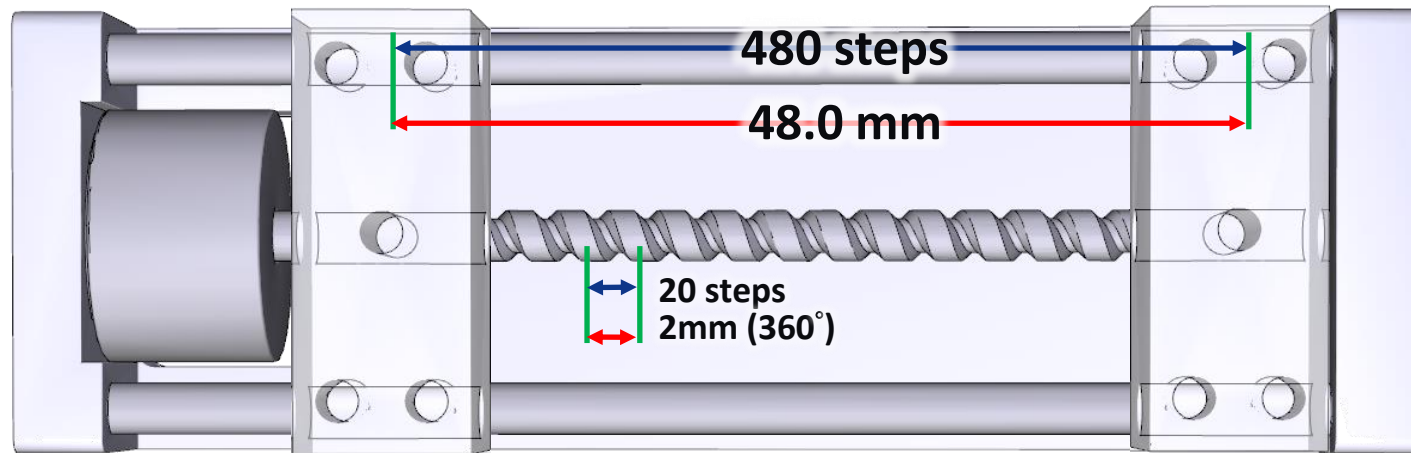
Stepper Motor Driver Board (Allegro A4988)

MS1	MS2	MS3	Resolution
Low	Low	Low	Full Step
High	Low	Low	Half Step
Low	High	Low	Quarter Step
High	High	Low	Eighth Step
High	High	High	Sixteenth Step



Stepper Motor(Summer Elite)

- Step angle : 18° .
- Steps per revolution : $360^\circ \div 18^\circ = 20$ (Steps/rev).
- Screw pitch : 2 (mm/rev).
- Screw length : 58.2 mm.
- Stroke block width : 10.2 mm.
- Slide travel : 48.0 mm. (58.2 - 10.2) mm
- Driver micro stepping : Full Step.

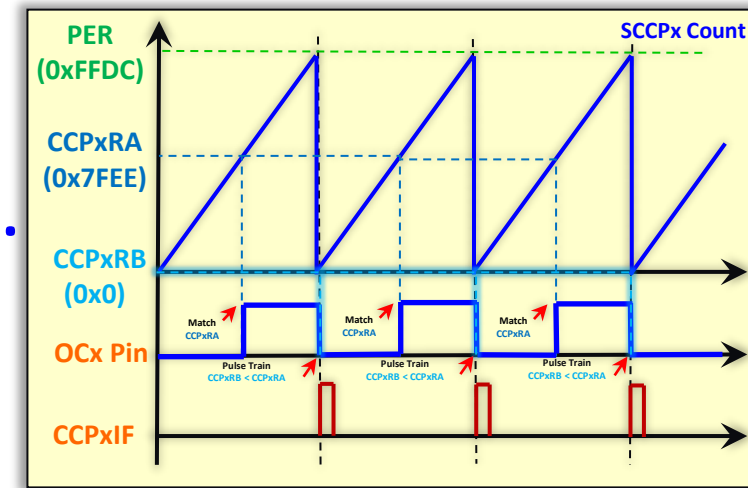


Slide travel in steps : $48.0 \text{ (mm)} / 2 \text{ (mm/rev)} \times 20 \text{ (Steps/rev)} = 480 \text{ (Steps)}$

Lab10 SCCP Compare/PWM Interrupt

Lab10 SCCP Compare/PWM Interrupt

- Base on current project or open `.\Lab10_xxx.X` to do exercises.
- Add **SCCP1**, **SCCP2** to your project for Stepper Motor control.
- Configure both **SCCP1**, **SCCP2** as below
 - Mode Selection : **Compare/PWM**
 - Enable MCCP : **Disable** (**SCCP1** and **SCCP2** set to disable while initialize)
 - Enable MCCP Interrupt : **Enable** (Enable **SCCP1** and **SCCP2** interrupt)
 - Clock Source : **FOSC/2 (16MHz)**
 - Mode : **Dual Edge Compare, Buffer(PWM)**
 - Primary Compare Value : **0x7FEE** (Duty cycle **50%**, $0xFFDC/2 = 0x7FEE$)
 - Secondary Compare Value : **0x0**
 - Timer Period : **0xFFDC** (Frequency **244.27Hz**, $16\text{MHz}/244.27\text{Hz} = 65500 = 0xFFDC$)
- Clock output on **OCM1\RC7**, **OCM2\RD11** and Dir GPIO on **DIR1\RD10**, **DIR2\RC3**.

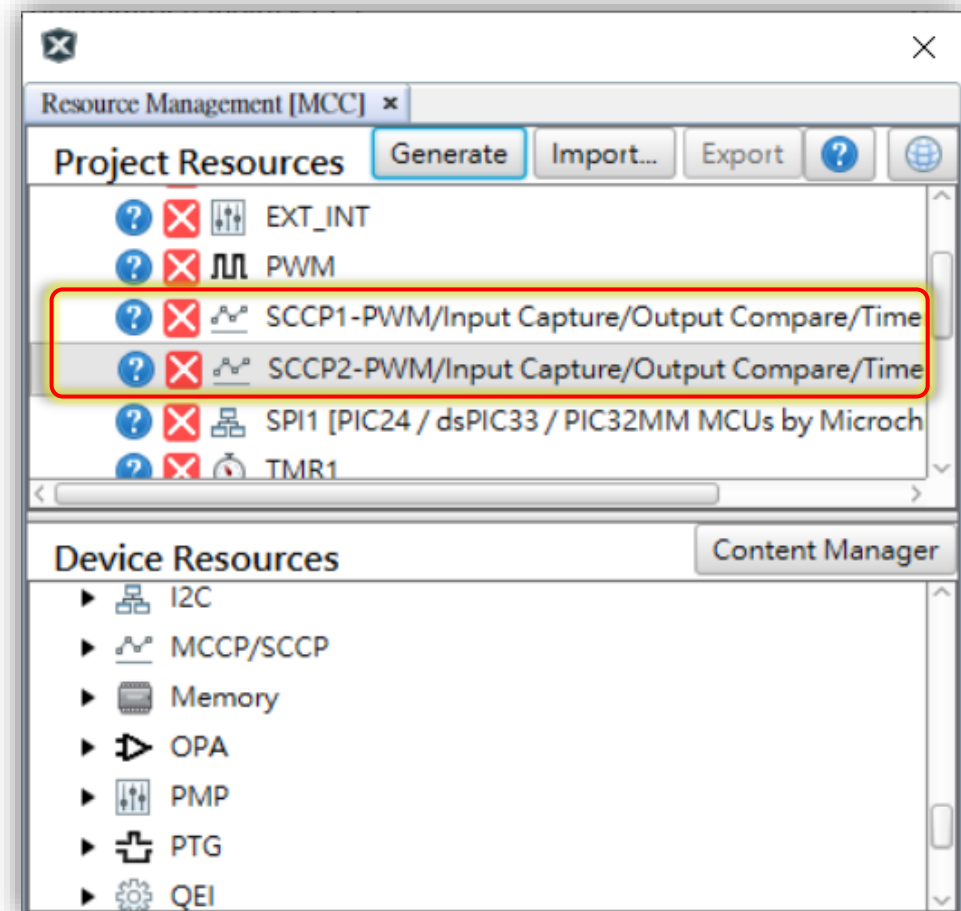
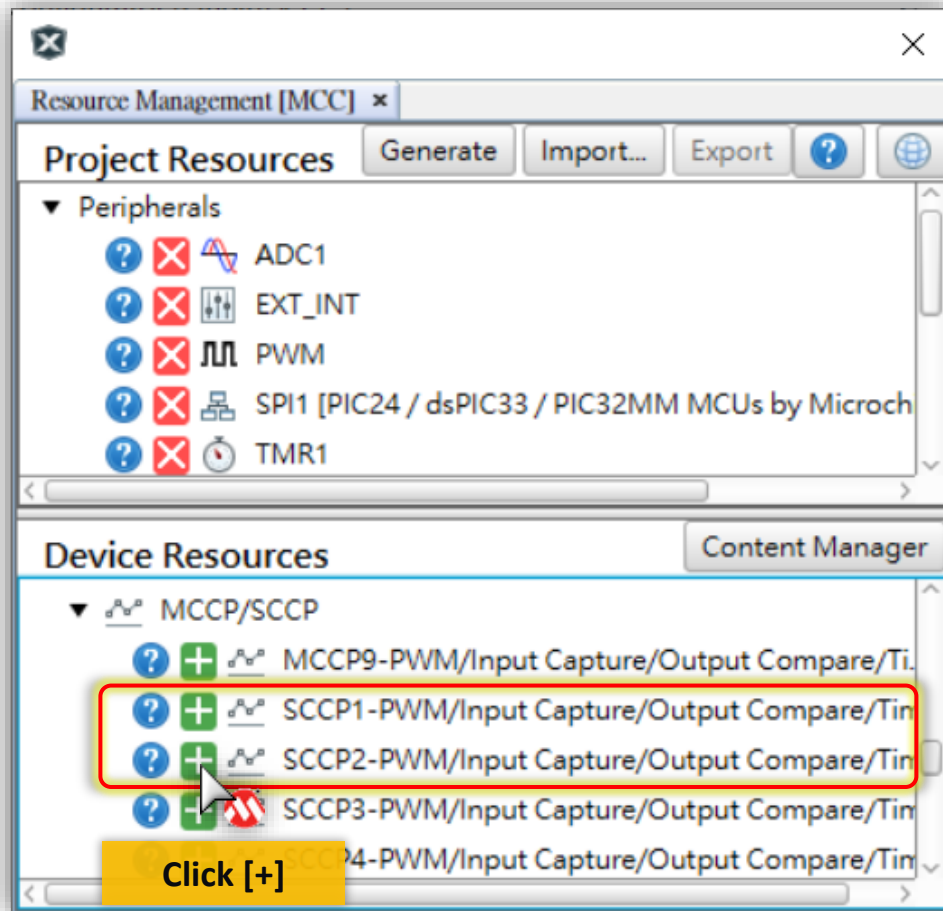


Let's go!

Lab10 SCCP Compare/PWM Interrupt

Step 1

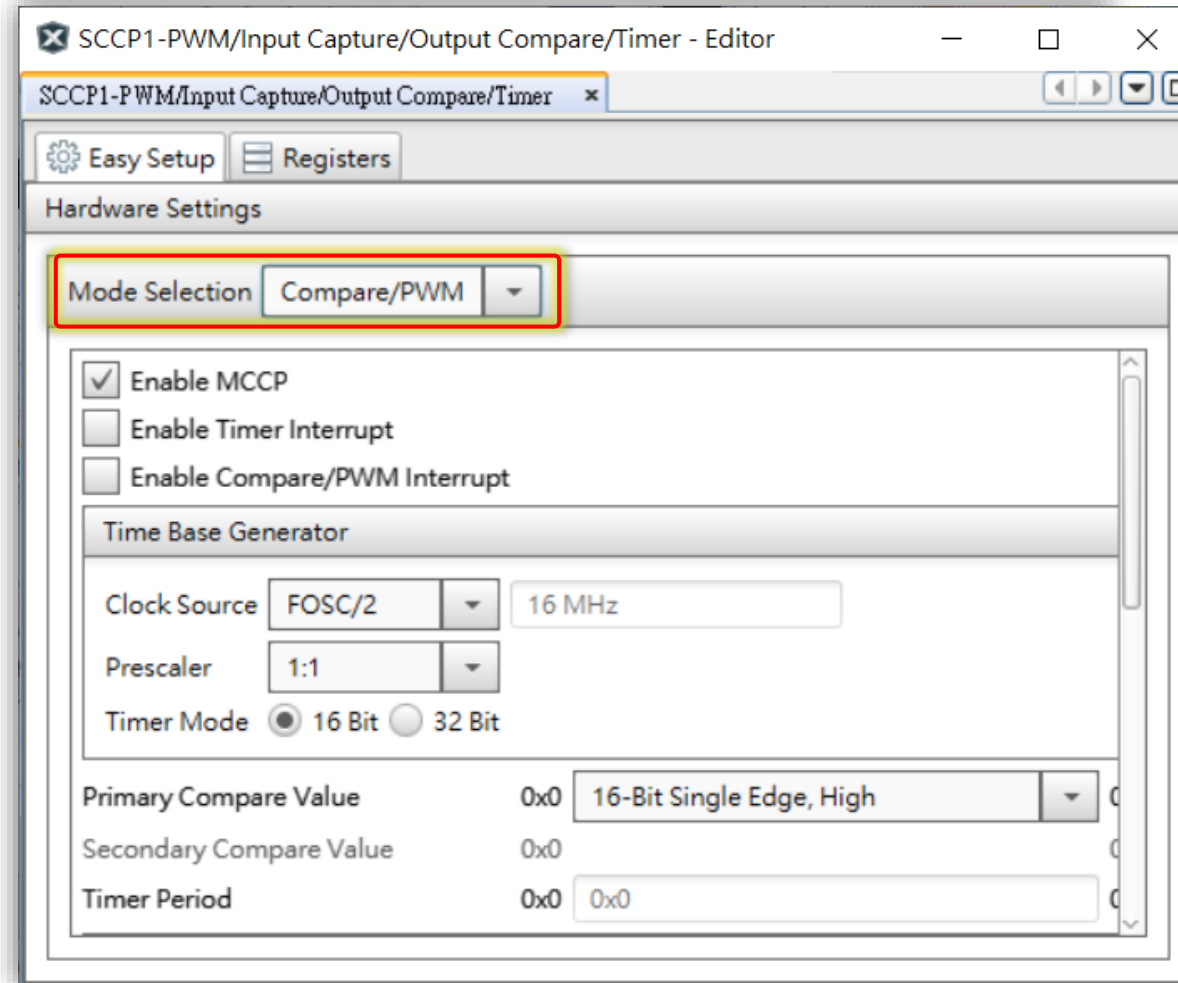
- Click [+] to add **MCCP/SCCP** > both **SCCP1** and **SCCP2** Module.



Lab10 SCCP Compare/PWM Interrupt

Step 2

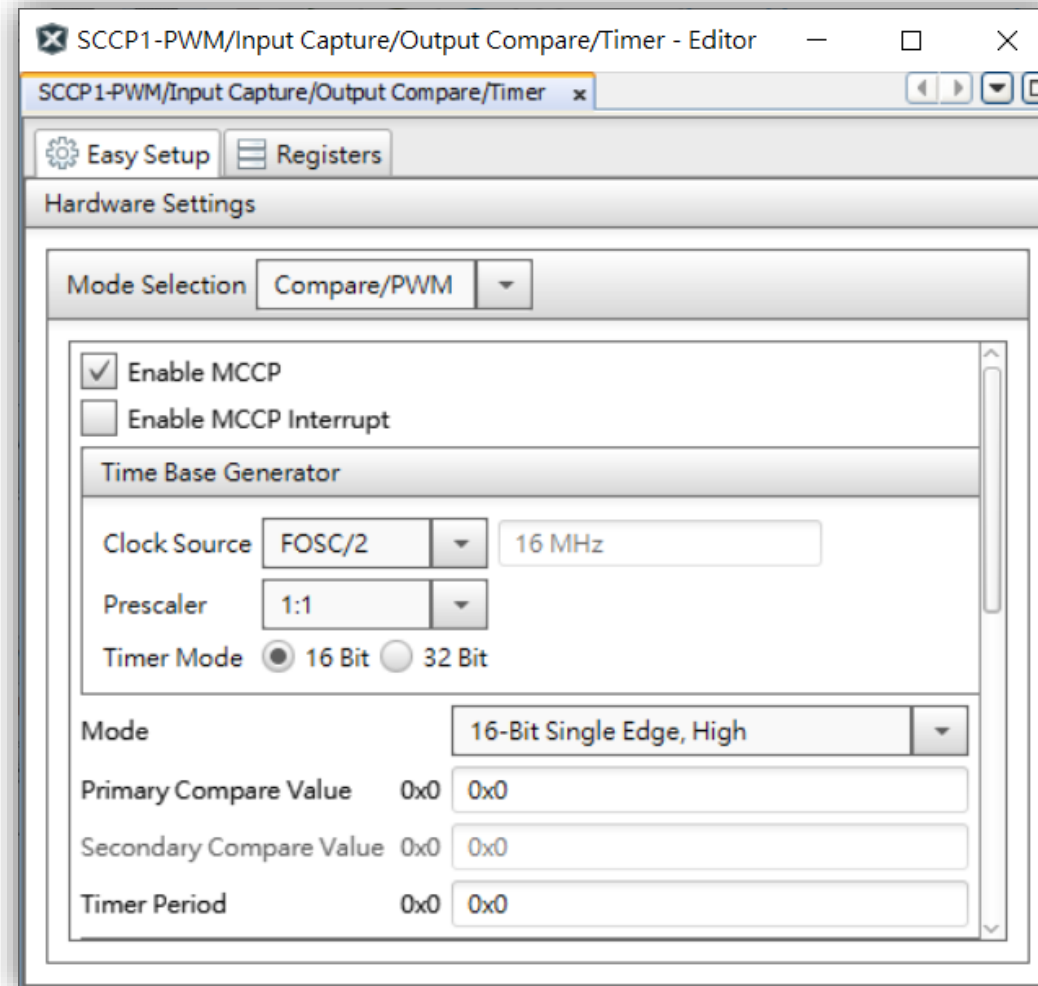
- Set **MCCP1 and 2 Mode Selection** : **Timer > Compare/PWM**.



MCC's MCCP&SCCP Module

Notice

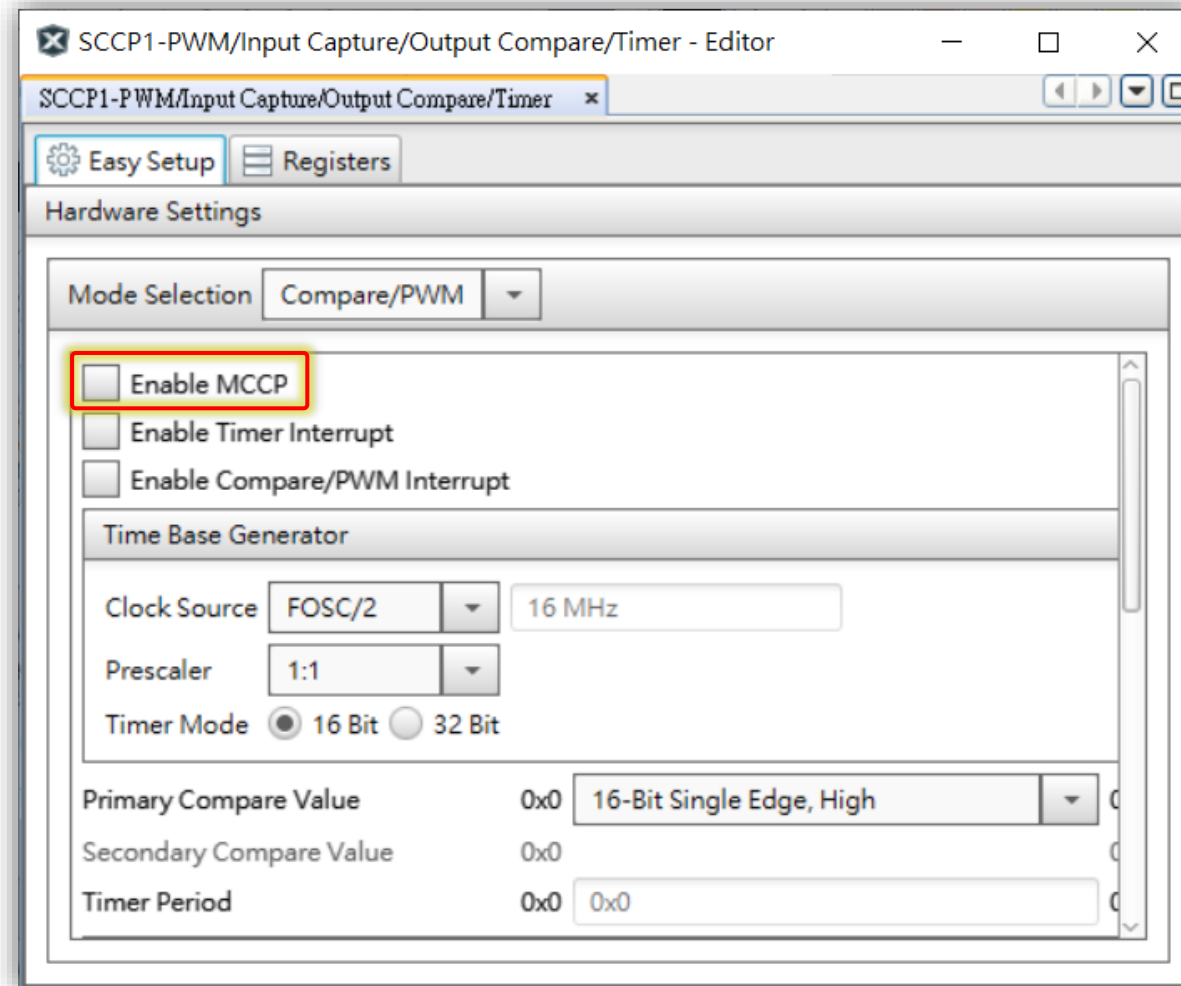
- **SCCP/MCCP default settings window.**



Lab10 SCCP Compare/PWM Interrupt

Step 3

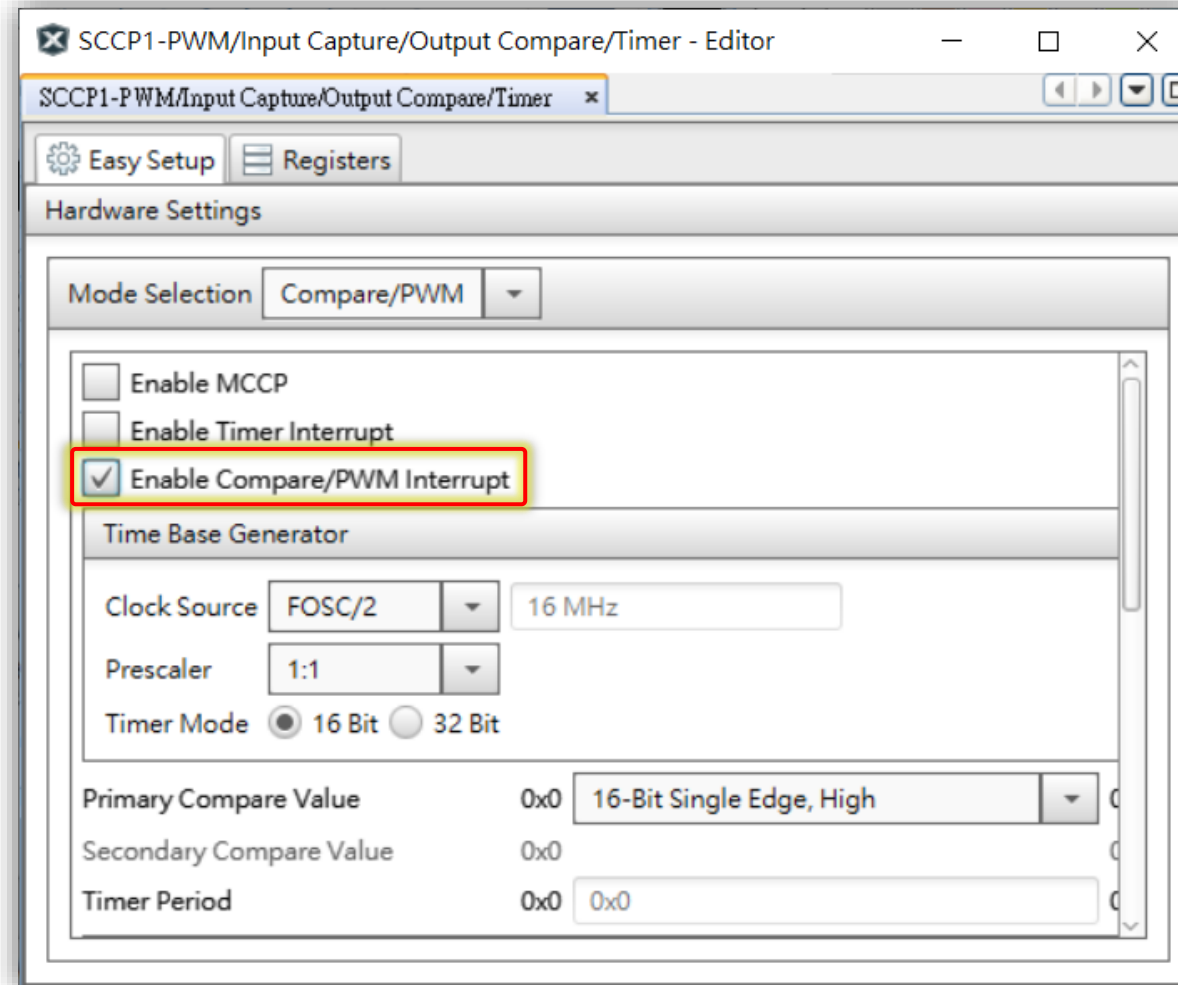
- Set **MCCP1** and **2 MCCP State** : **Enable MCCP** > **Disable MCCP**.



Lab10 SCCP Compare/PWM Interrupt

Step 4

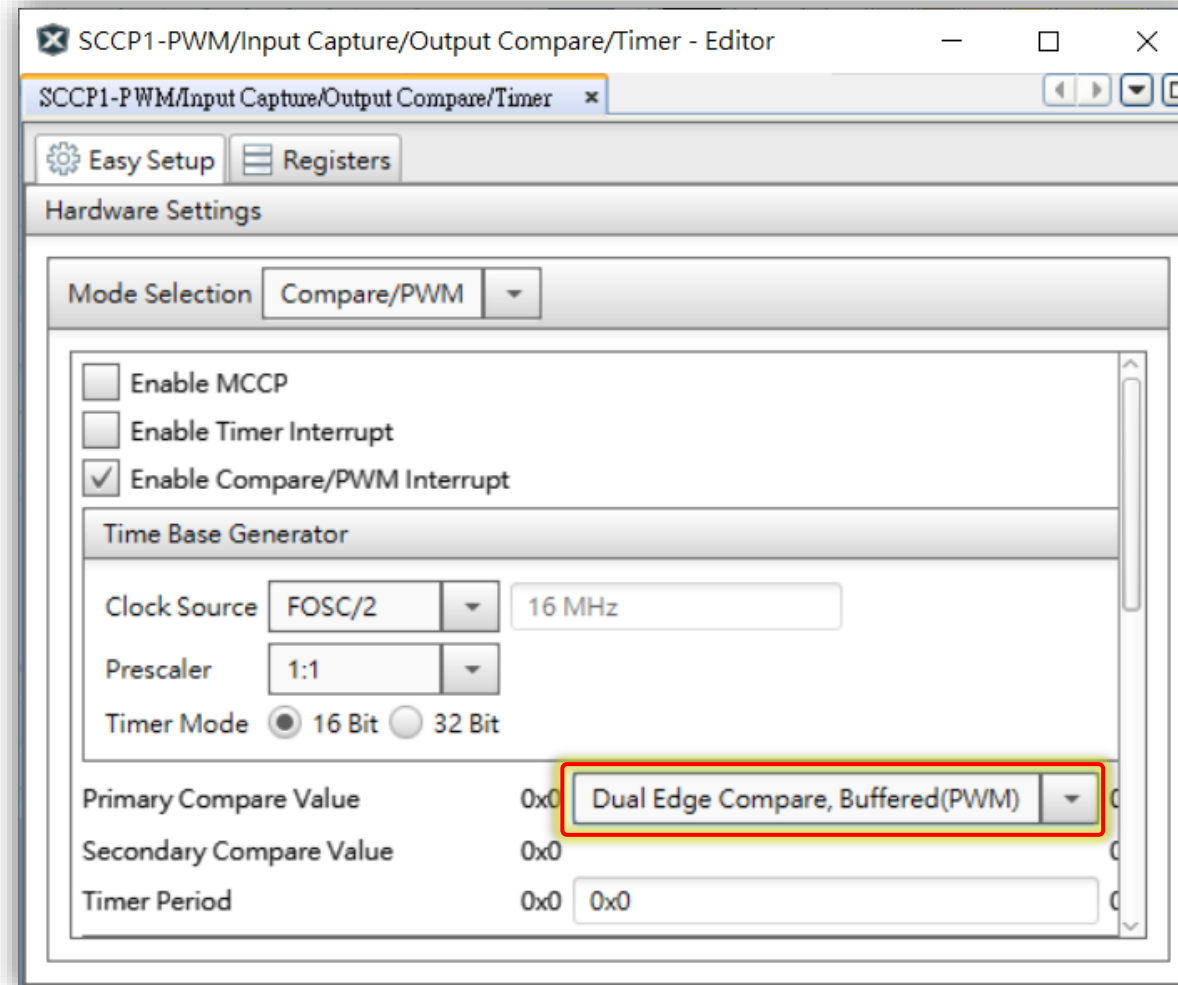
- Set **MCCP1** and **2 MCCP State** : **Enable Compare/PWM Interrupt**.



Lab10 SCCP Compare/PWM Interrupt

Step 5

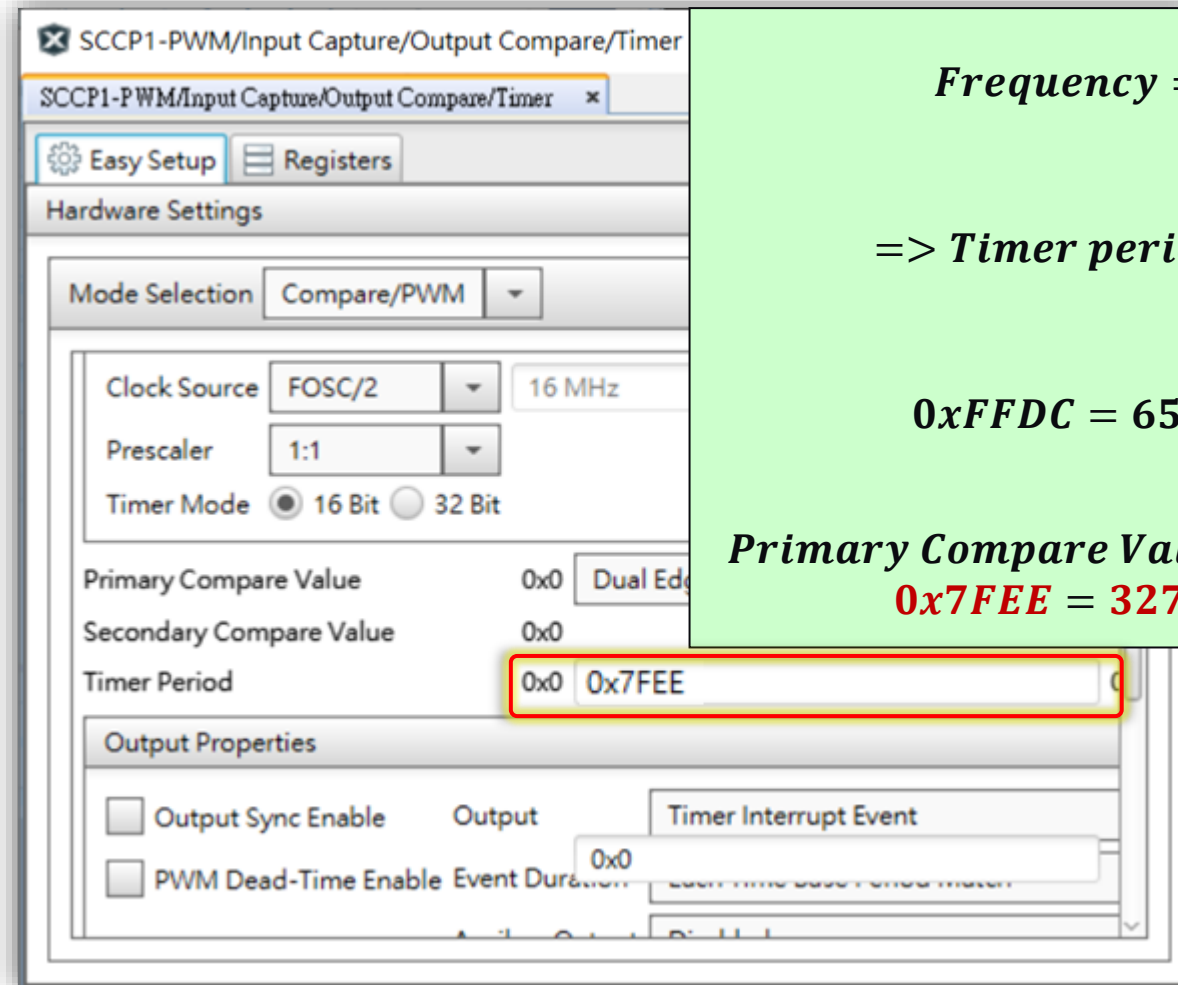
- Set **MCCP1 and 2 Mode** : **16-Bit Single Edge, High > Dual Edge Compare, Buffer(PWM)**.



Lab10 SCCP Compare/PWM Interrupt

Step 6

- Set **MCCP1** and **2 Primary Compare Value** : **0x0 > 0x7FEE(32750)**.



$$Frequency = \left(\frac{Clock\ Source}{Timer\ period} \right)$$

$$\Rightarrow Timer\ period = \left(\frac{Clock\ Source}{Frequency} \right)$$

$$0xFFDC = 65500 = \left(\frac{16MHz}{244.27Hz} \right)$$

$$Primary\ Compare\ Value = (Timer\ period \times 50\%)$$

$$0x7FEE = 32750 = (65500 \times 0.5)$$

Lab10 SCCP Compare/PWM Interrupt

Step 7

- Set **MCCP1** and **2** **Timer Period** : **0x0 > 0xFFDC(65500)**.

SCCP1-PWM/Timer

SCCP1-PWM/Timer

Easy Setup Registers

Hardware Settings

Mode Selection Compare/PWM

Clock Source FOSC/2 16 MHz

Prescaler 1:1

Timer Mode 16 Bit 32 Bit

Primary Compare Value 0x0 Dual Edge Compare, Buffered(PWM)

Secondary Compare Value 0x0

Timer Period 0x0 0x7FEE

Output Properties

Output Sync Enable Output

PWM Dead-Time Enable Event Duration

Timer Interrupt Event 0xFFDC

$$Frequency = \left(\frac{Clock\ Source}{Timer\ period} \right)$$

$$\Rightarrow Timer\ period = \left(\frac{Clock\ Source}{Frequency} \right)$$

$$0xFFDC = 65500 = \left(\frac{16MHz}{244.27Hz} \right)$$

Lab10 SCCP Compare/PWM Interrupt

Step 8

- Assign OCM1 and OCM2.
 - Pin Manager : RC7 > OCM1, RD11 > OCM2.

Pin Manager: Grid View

			Port C																Port D															
Module	Function	Direction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pin Module	GPIO	input																																
	GPIO	output																																
SCCP1-PWM/Input Capture/Output Compare/Timer	OCFA	input																																
	OCFB	input																																
	OCFC	input																																
	OCFD	input																																
	OCM1	output																																
	TCKI1	input																																
SCCP2-PWM/Input Capture/Output Compare/Timer	OCFA	input																																
	OCFB	input																																
	OCFC	input																																
	OCFD	input																																
	OCM2	output																																
	TCKI2	input																																

Lab10 SCCP Compare/PWM Interrupt

Step 9

- Set **PORTD** : **Pin10 (RD10)** and **PORTC** : **Pin3 (RC3)** to digital output mode.

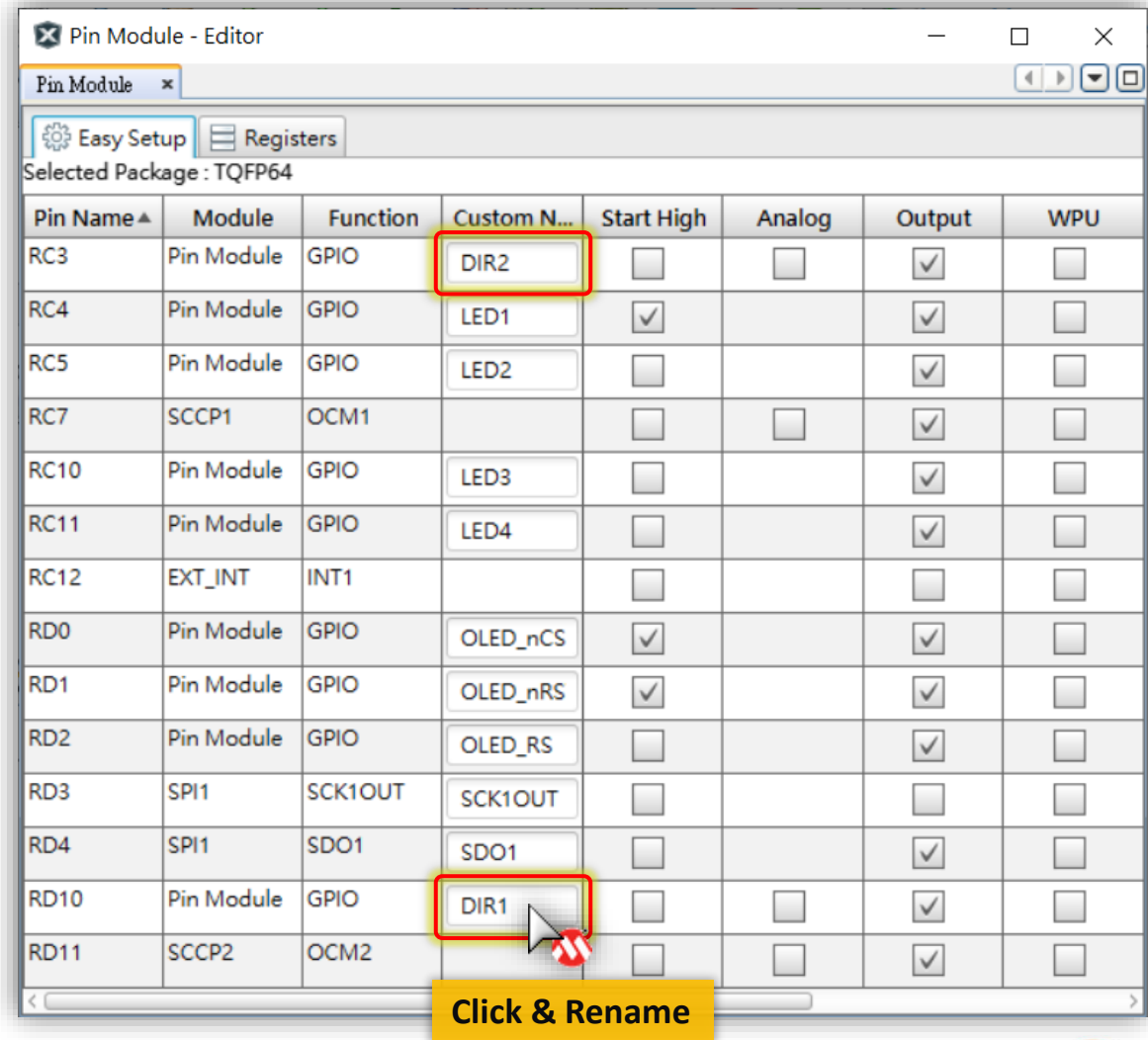
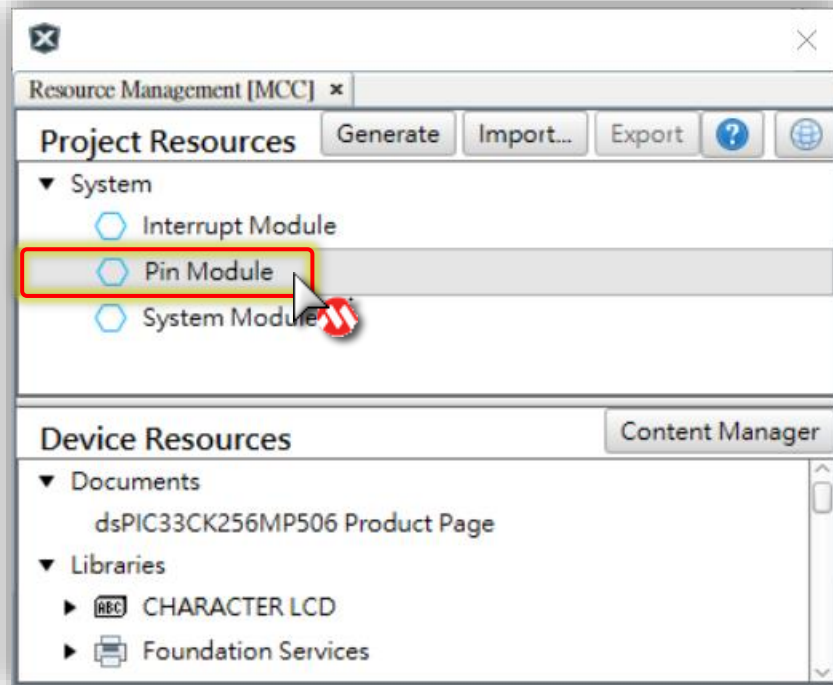
Pin Manager: Grid View

			Port C																Port D															
Module	Function	Direction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pin Module	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
SCCP1-PWM/Input Capture/Output Compare/Timer	OCFA	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFB	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFC	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFD	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCM1	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	TCKI1	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	
SCCP2-PWM/Input Capture/Output Compare/Timer	OCFA	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFB	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFC	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCFD	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	OCM2	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	TCKI2	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		

Lab10 SCCP Compare/PWM Interrupt

Step 10

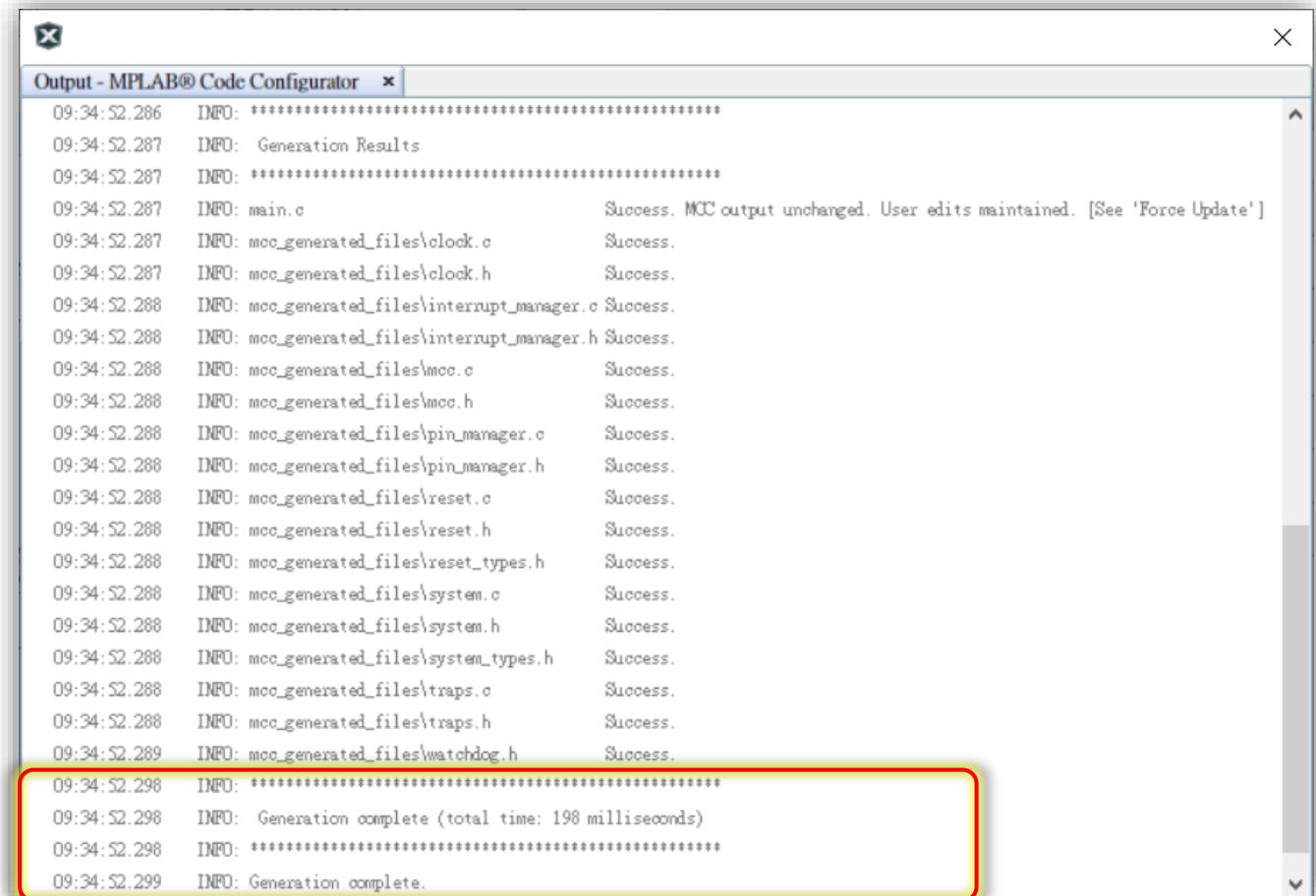
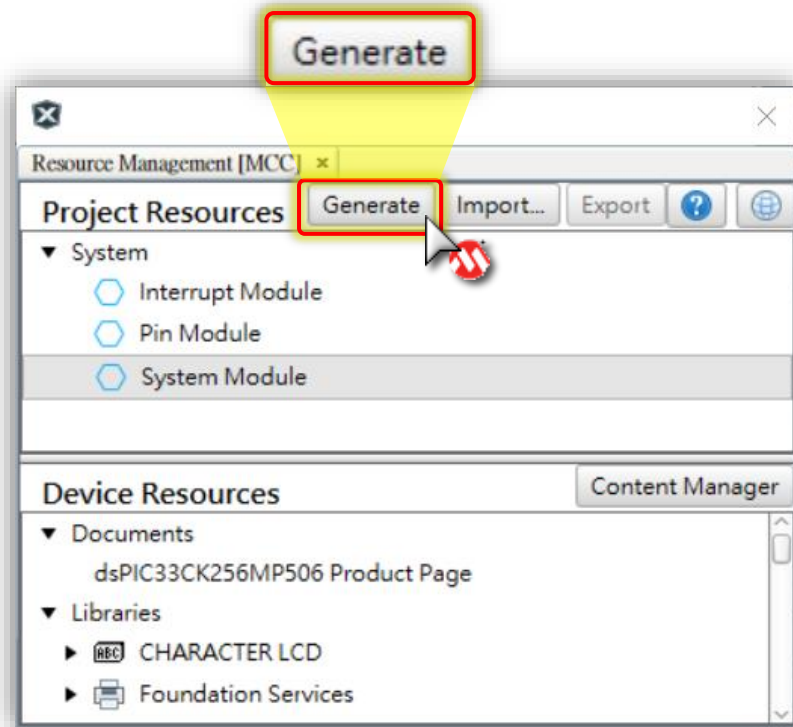
- Set Alias for RD10, RC3.
 - Pin Module : RD10 Custom Name > DIR1.
 - Pin Module : RC3 Custom Name > DIR2.



Lab10 SCCP Compare/PWM Interrupt

Step 11

- Click "**Generate**" Button to generate your code.



Lab10 SCCP Compare/PWM Interrupt

Notice

- Copy Source Files paste on folder.
(\Appexdix\MCCLibraries to \Labn_xxx.X\ mcc_generated_files)
 - Source Files > "sccp1_compare.c", "sccp2_compare.c".

 reset_types	C Header File
 sccp1_compare	C Source File
 sccp1_compare	C Header File
 sccp2_compare	C Source File
 sccp2_compare	C Header File

Lab10 SCCP Compare/PWM Interrupt

Hint

- Confirm "**sccp1_compare.c** , **sccp2_compare.c** " CCP**xRA** and CCP**xRB** Value.

```
void SCCP1_COMPARE_Initialize (void)
```

```
{
```

```
...
```

```
//TMR 0;
```

```
CCP1TMRH = 0x00;
```

```
//PR 65500;
```

```
CCP1PRL = 0xFFDC;
```

```
//PR 0;
```

```
CCP1PRH = 0x00;
```

```
//CMP 32750;
```

```
CCP1RA = 0x7FEE;
```

```
//CMP 0;
```

```
CCP1RB = 0x00;
```

```
//BUF 0;
```

```
CCP1BUFL = 0x00;
```

```
//BUF 0;
```

```
CCP1BUFH = 0x00;
```

```
...
```

```
}
```

```
void SCCP2_COMPARE_Initialize (void)
```

```
{
```

```
...
```

```
//TMR 0;
```

```
CCP2TMRH = 0x00;
```

```
//PR 65500;
```

```
CCP2PRL = 0xFFDC;
```

```
//PR 0;
```

```
CCP2PRH = 0x00;
```

```
//CMP 32750;
```

```
CCP2RA = 0x7FEE;
```

```
//CMP 0;
```

```
CCP2RB = 0x00;
```

```
//BUF 0;
```

```
CCP2BUFL = 0x00;
```

```
//BUF 0;
```

```
CCP2BUFH = 0x00;
```

```
...
```

```
}
```

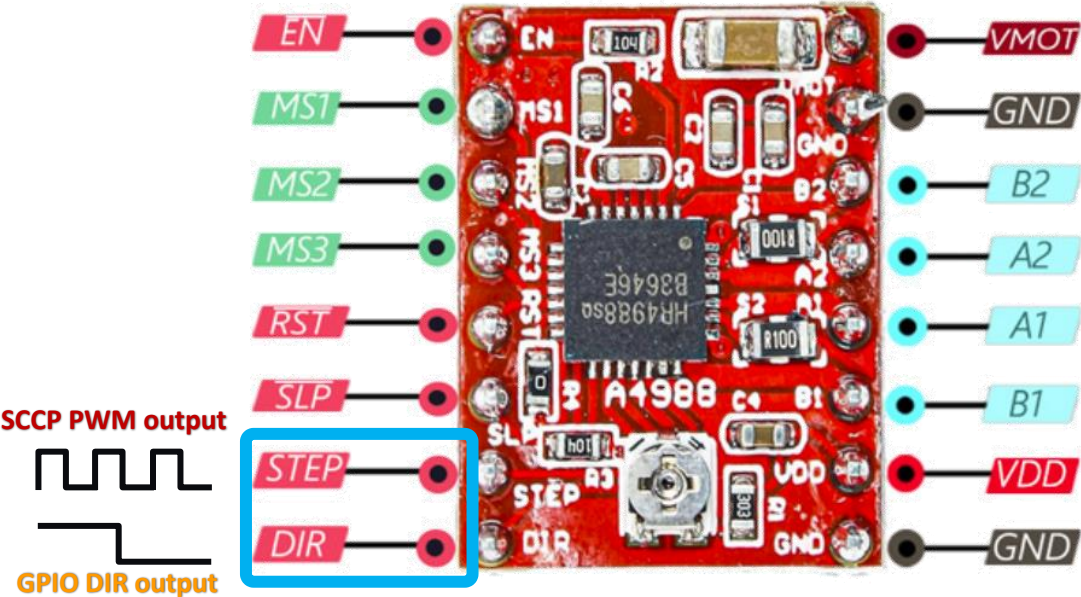
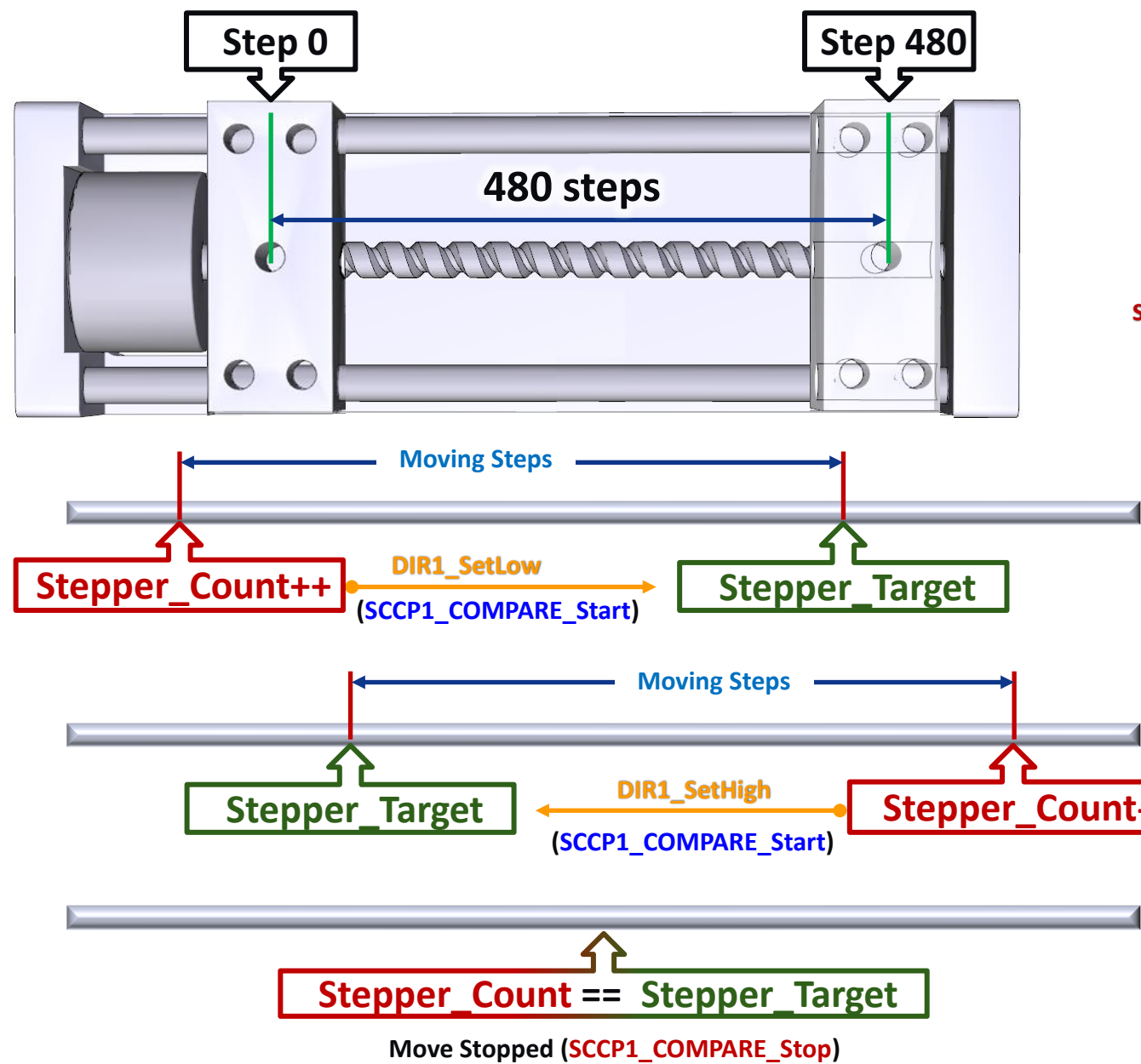

MCCP&SCCP Functions of MCC

- MCC Provide below functions for SCCP_n:

Prototype definition : "sccp_n_compare.h"

- void SCCP_n_COMPARE_Initialize (void);
- void SCCP_n_COMPARE_Start(void);
- void SCCP_n_COMPARE_Stop(void);
- void SCCP_n_COMPARE_Callback(void);
- void SCCP_n_COMPARE_DualCompareValueSet(uint16_t priVal, uint16_t secVal);
- void SCCP_n_COMPARE_CenterAlignedPWMConfig(uint16_t priVal, uint16_t secVal);
- void SCCP_n_COMPARE_DualEdgeBufferedConfig(uint16_t priVal, uint16_t secVal);
- void SCCP_n_COMPARE_EdgeAlignedPWMConfig(uint16_t priVal, uint16_t secVal);
- void SCCP_n_COMPARE_SingleCompareStatusClear(void);
- bool SCCP_n_COMPARE_SingleCompareStatusGet(void);
- bool SCCP_n_COMPARE_IsCompareCycleComplete(void);

Stepper Motor Control(Summer Elite)



SCCPx_COMPARE_Callback
Stepper_Count++ or Stepper_Count--

Moving Direction = DIR_UP

Moving Steps = Target - Step

Moving Direction = DIR_DN

Lab10 SCCP Compare/PWM Interrupt

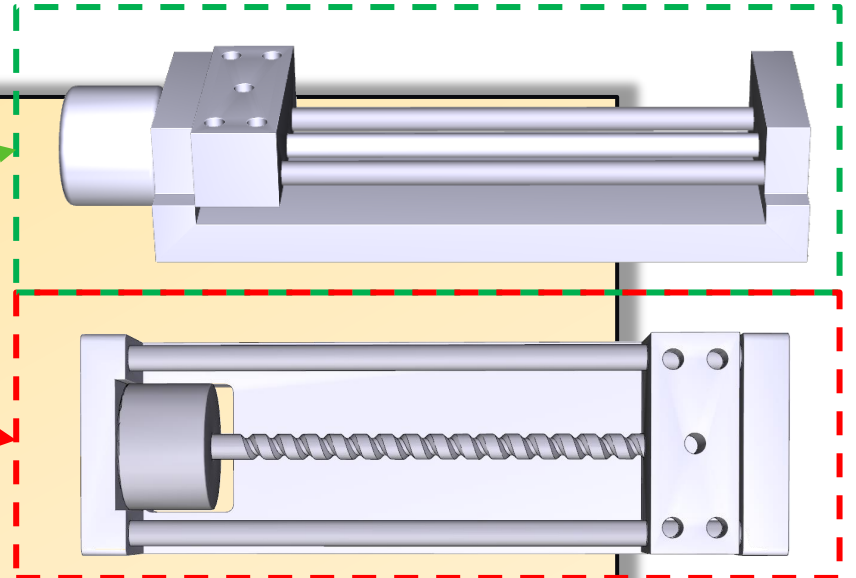
Code Example

```
#include "mcc_generated_files/pwm.h"
#include "mcc_generated_files/sccp1_compare.h"
#include "mcc_generated_files/sccp2_compare.h"
/*
    Main application
*/
#define Motor_Model1 0 //Winter Elite Motor
#define Motor_Model2 1 //Summer Elite Motor

#define MOTOR_TYPE Motor_Model2

#if MOTOR_TYPE == Motor_Model1
#define STEPPER_SKETCH_RECT (57.6) // 57.6 mm (Slide travel distance)
#define STEPPER_STEP_MODE 1 // 1:Full, 2:Half, 4:Quarter, 8:Eighth, 16:Sixteenth Step Mode
#define STEPPER_SCREW_PITCH 3 // mm
#define STEPPER_STEP_PER_REV 20 // (Steps/rev)
#define STEPPER_STEP_COUNT (STEPPER_SKETCH_RECT*STEPPER_STEP_PER_REV*STEPPER_STEP_MODE/STEPPER_SCREW_PITCH) // 384 Steps
#define STEPPER_X_DIR_LEFT DIR1_SetLow
#define STEPPER_X_DIR_RIGHT DIR1_SetHigh
#define STEPPER_Y_DIR_UP DIR2_SetLow
#define STEPPER_Y_DIR_DOWN DIR2_SetHigh

#elif MOTOR_TYPE == Motor_Model2
#define STEPPER_SKETCH_RECT (48.0) // 48.8 mm (Slide travel distance)
#define STEPPER_STEP_MODE 1 // 1:Full, 2:Half, 4:Quarter, 8:Eighth, 16:Sixteenth Step Mode
#define STEPPER_SCREW_PITCH 2 // mm
#define STEPPER_STEP_PER_REV 20 // (Steps/rev)
#define STEPPER_STEP_COUNT (STEPPER_SKETCH_RECT*STEPPER_STEP_PER_REV*STEPPER_STEP_MODE/STEPPER_SCREW_PITCH) // 488 Steps
#define STEPPER_X_DIR_LEFT DIR1_SetHigh
#define STEPPER_X_DIR_RIGHT DIR1_SetLow
#define STEPPER_Y_DIR_UP DIR2_SetHigh
#define STEPPER_Y_DIR_DOWN DIR2_SetLow
#endif
```



Lab10 SCCP Compare/PWM Interrupt

Code Example

```
uint16_t Servo_angle;
uint8_t Stepper_Dir = 0;
volatile uint16_t Stepper1_Count = 0;
volatile uint16_t Stepper2_Count = 0;
volatile uint16_t Stepper1_Target = STEPPER_STEP_COUNT;
volatile uint16_t Stepper2_Target = STEPPER_STEP_COUNT;

int main(void)
{
    while (1)
    {
        if ( !BT2_GetValue() )
        {
            while ( !BT2_GetValue() );
            if( Stepper_Dir )
            {
                Stepper1_Target = STEPPER_STEP_COUNT;
                Stepper2_Target = STEPPER_STEP_COUNT;
            }
            else
            {
                Stepper1_Target = 0;
                Stepper2_Target = 0;
            }
            Stepper_Dir = !Stepper_Dir;
        }
        if (TMR1Flag)
        {
            ...
            Stepper1_Task();
            Stepper2_Task();
        }
    }
    return 1;
}
```

```
int main(void)
{
    return 1;
}
...
void EX_INT1_CallBack(void)
{ ... }

void SCCP1_COMPARE_CallBack( void )
{
    if ( Stepper1_Count < Stepper1_Target )
    {
        Stepper1_Count++;
    }
    else if ( Stepper1_Count > Stepper1_Target )
    {
        Stepper1_Count--;
    }
}

void SCCP2_COMPARE_CallBack( void )
{
    if ( Stepper2_Count < Stepper2_Target )
    {
        Stepper2_Count++;
    }
    else if ( Stepper2_Count > Stepper2_Target )
    {
        Stepper2_Count--;
    }
}
```

Lab10 SCCP Compare/PWM Interrupt

Code Example

```
int main(void)
{
    return 1;
}

...
void EX_INT1_CallBack(void)
{ ... }

void SCCP1_COMPARE_CallBack( void )
{ ... }

void SCCP2_COMPARE_CallBack( void )
{ ... }

void Stepper1_Task( void )
{
    if ( Stepper1_Count < Stepper1_Target )
    {
        DIR1_SetLow();
        SCCP1_COMPARE_Start();
    }
    else if ( Stepper1_Count > Stepper1_Target )
    {
        DIR1_SetHigh();
        SCCP1_COMPARE_Start();
    }
    else if ( Stepper1_Count == Stepper1_Target )
    {
        SCCP1_COMPARE_Stop();
    }
}
```

```
...
void EX_INT1_CallBack(void)
{ ... }

void SCCP1_COMPARE_CallBack( void )
{ ... }

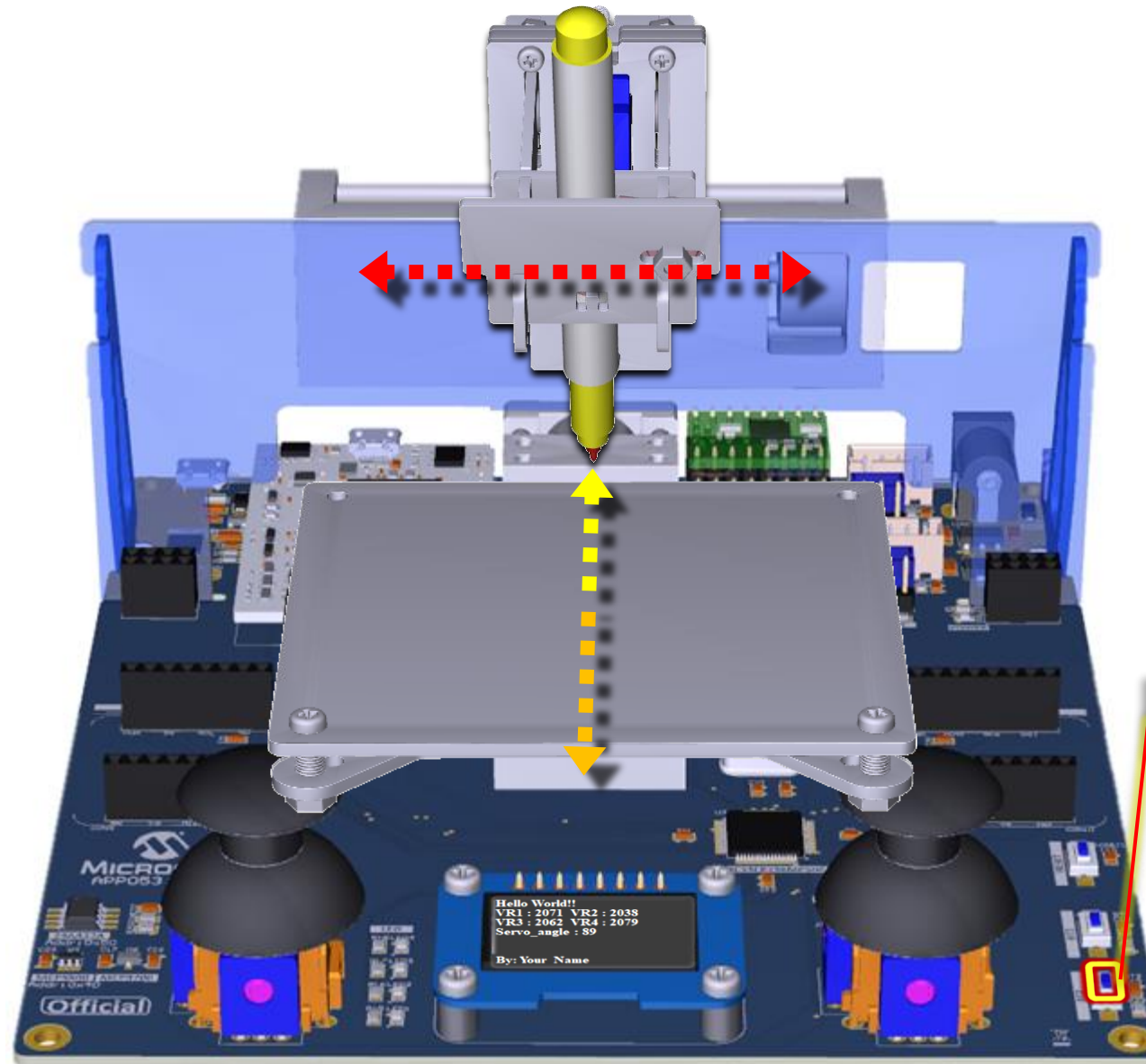
void SCCP2_COMPARE_CallBack( void )
{ ... }

void Stepper1_Task( void )
{ ... }

void Stepper2_Task( void )
{
    if ( Stepper2_Count < Stepper2_Target )
    {
        DIR2_SetLow();
        SCCP2_COMPARE_Start();
    }
    else if ( Stepper2_Count > Stepper2_Target )
    {
        DIR2_SetHigh();
        SCCP2_COMPARE_Start();
    }
    else if ( Stepper2_Count == Stepper2_Target )
    {
        SCCP2_COMPARE_Stop();
    }
}
```


Lab10 SCCP Compare/PWM Interrupt

Result

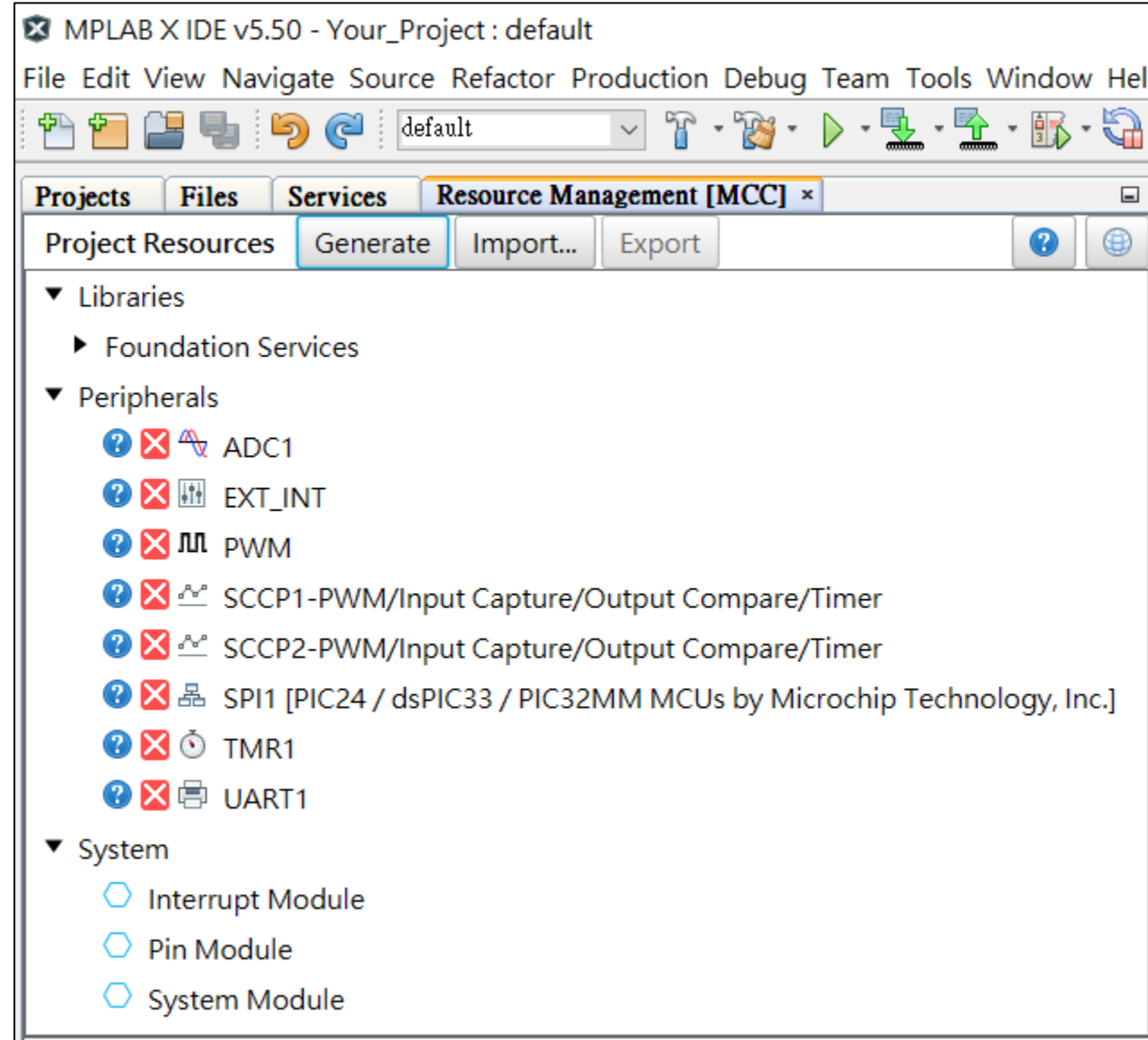


Stroke block move, When BT2 Pressed.

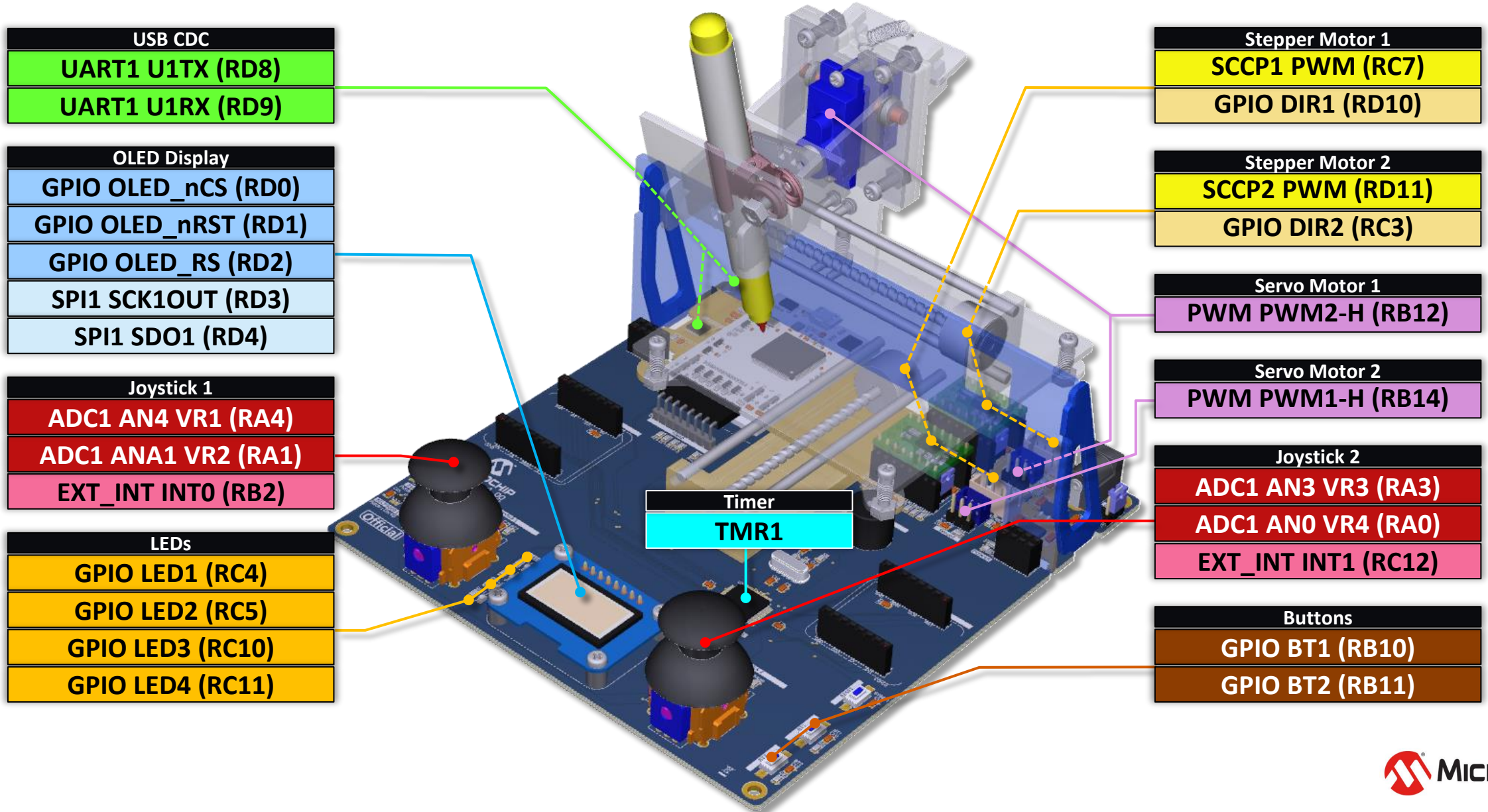
2D Sketch Drawing Application

Peripheral (MCC) and Application Implement

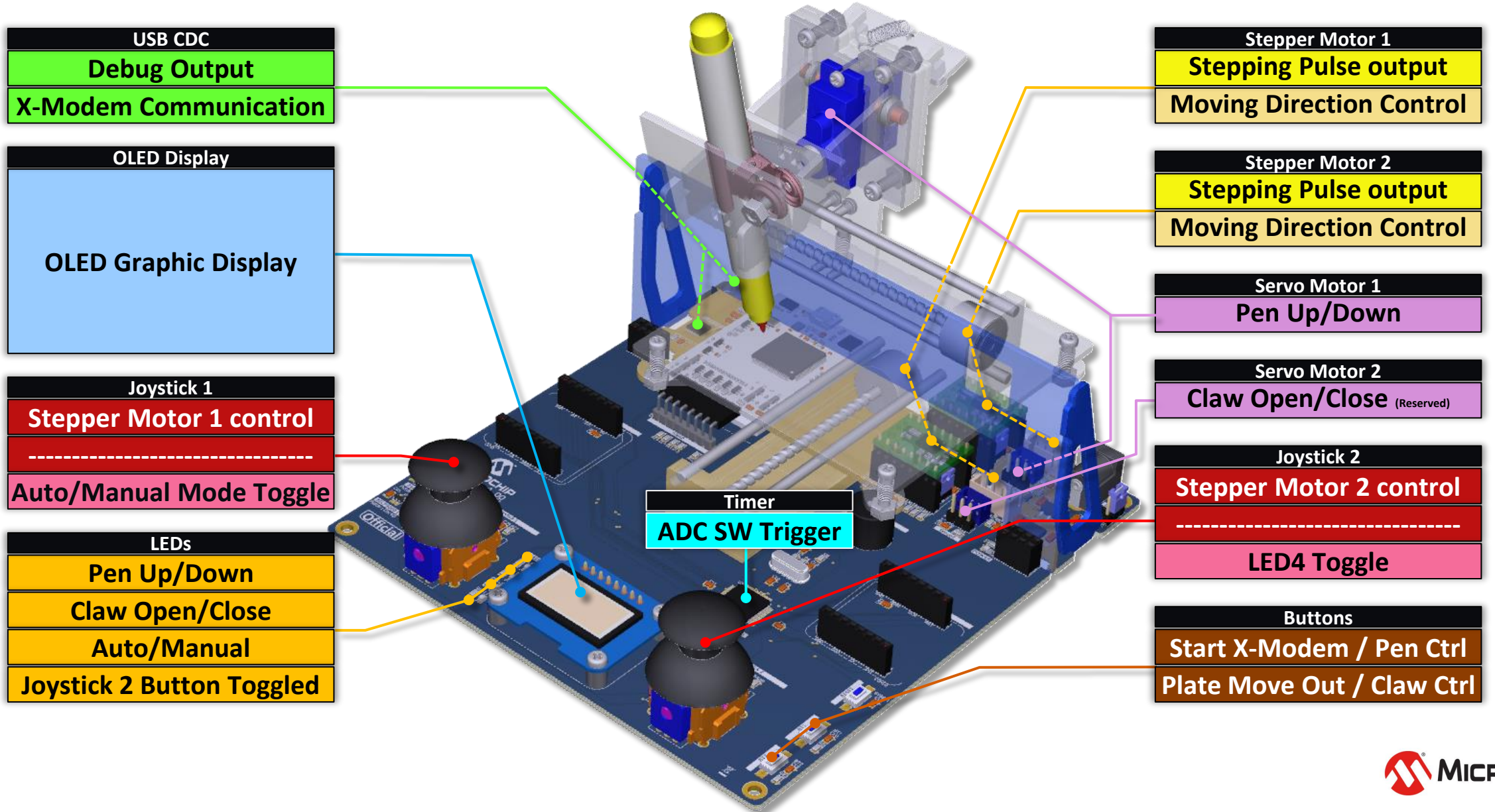
Peripheral in MCC



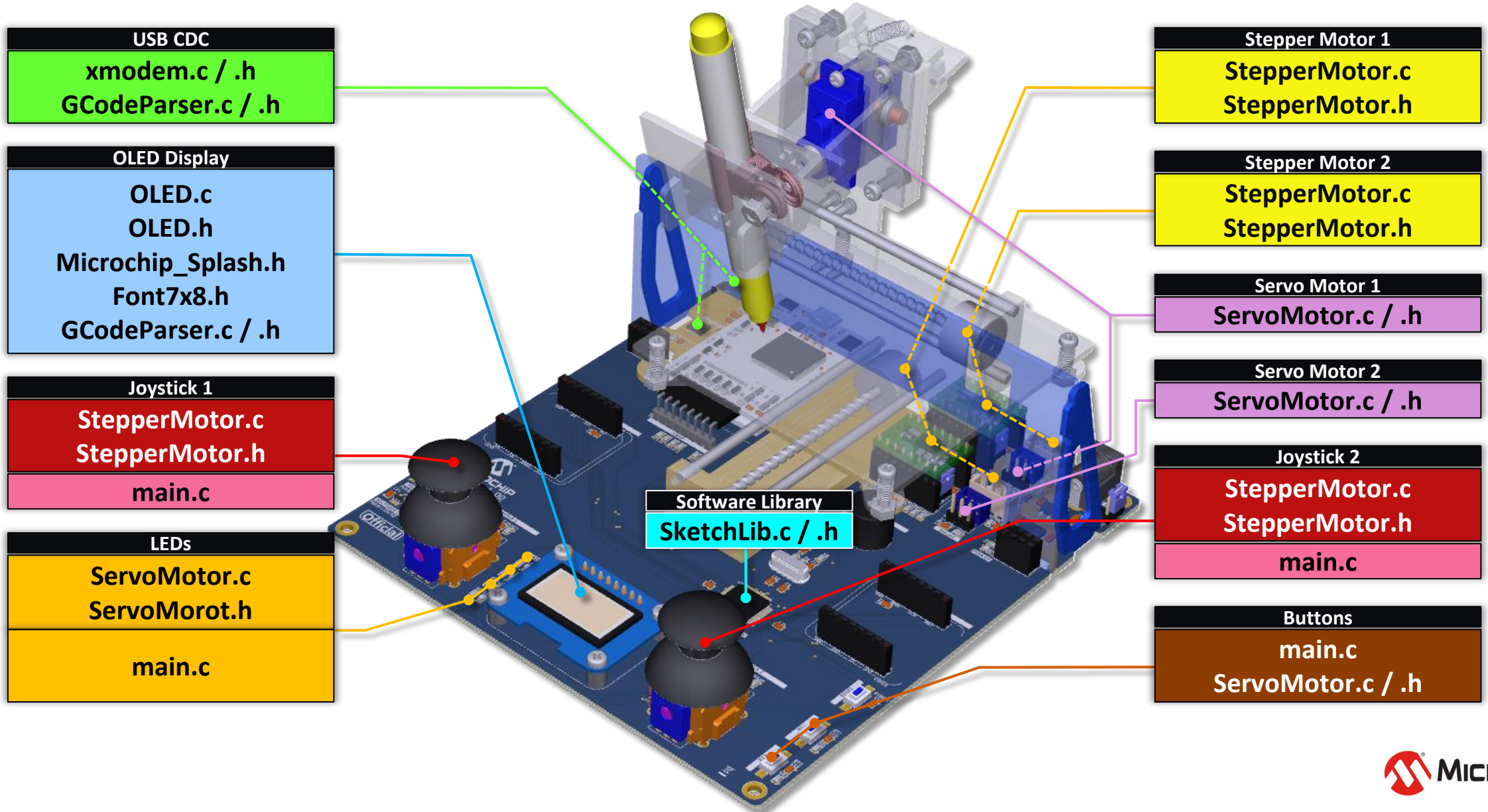
Application Involved Peripheral



Application



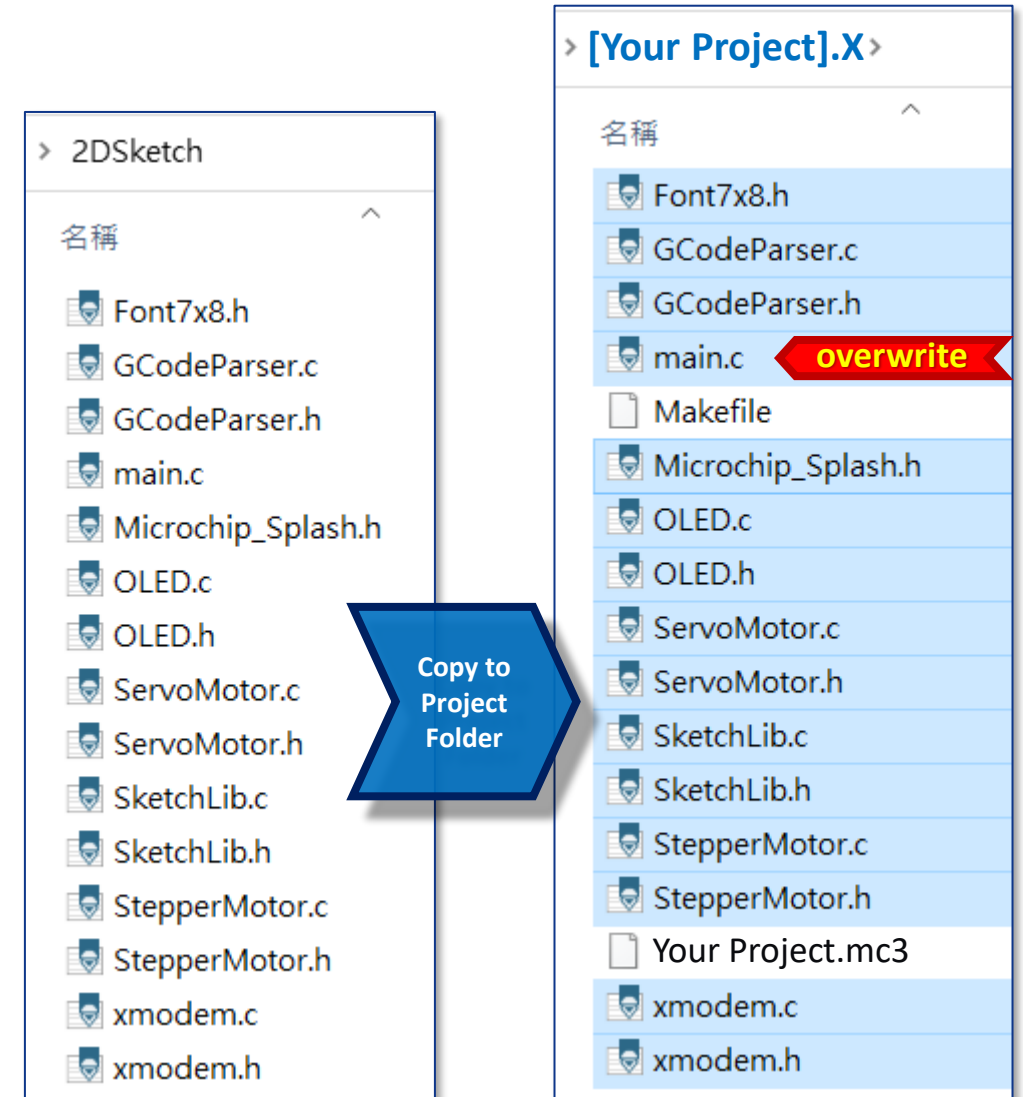
Application Implementation



Lab Preparation

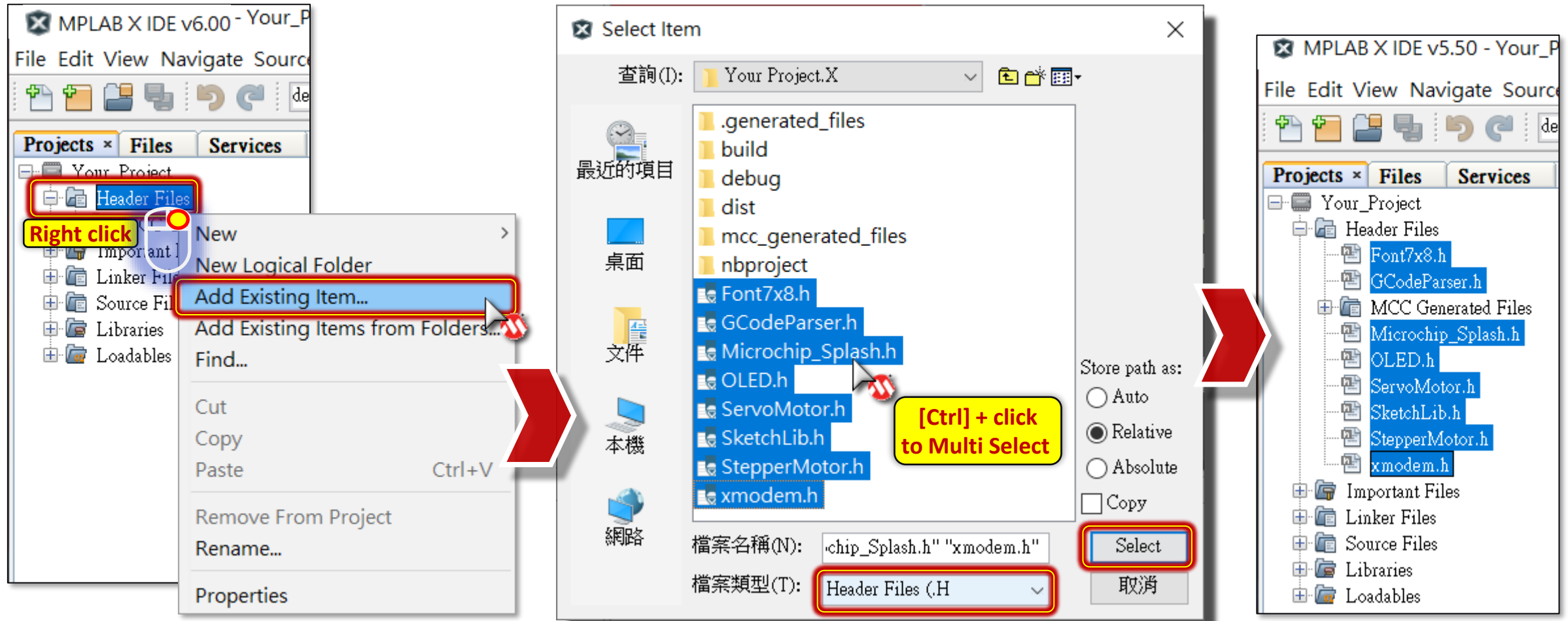
Lab Preparation

- Copy **\2DSketch*.*** to **\[Your Project].X**
 - **OLED Driver** (You should be done in SPI OLED section)
 - **OLED.c , OLED.h**
 - **Font and Splash bitmap** (You should be done in SPI OLED section)
 - **Microchip_Splash.h , Font7x8.h**
 - **Main flow (Application finished)**
 - **main.c** **Backup your main.c before overwrite it**
 - **X-Modem library**
 - **xmodem.c , xmodem.h**
 - **G-Code parser library**
 - **GCodeParser.c , GCodeParser.h**
 - **2D Sketch library**
 - **SketchLib.c , SketchLib.h**
 - **Servo Motor Driver**
 - **ServoMotor.c , ServoMotor.h**
 - **Stepper Motor Driver**
 - **StepperMotor.c , StepperMotor.h**



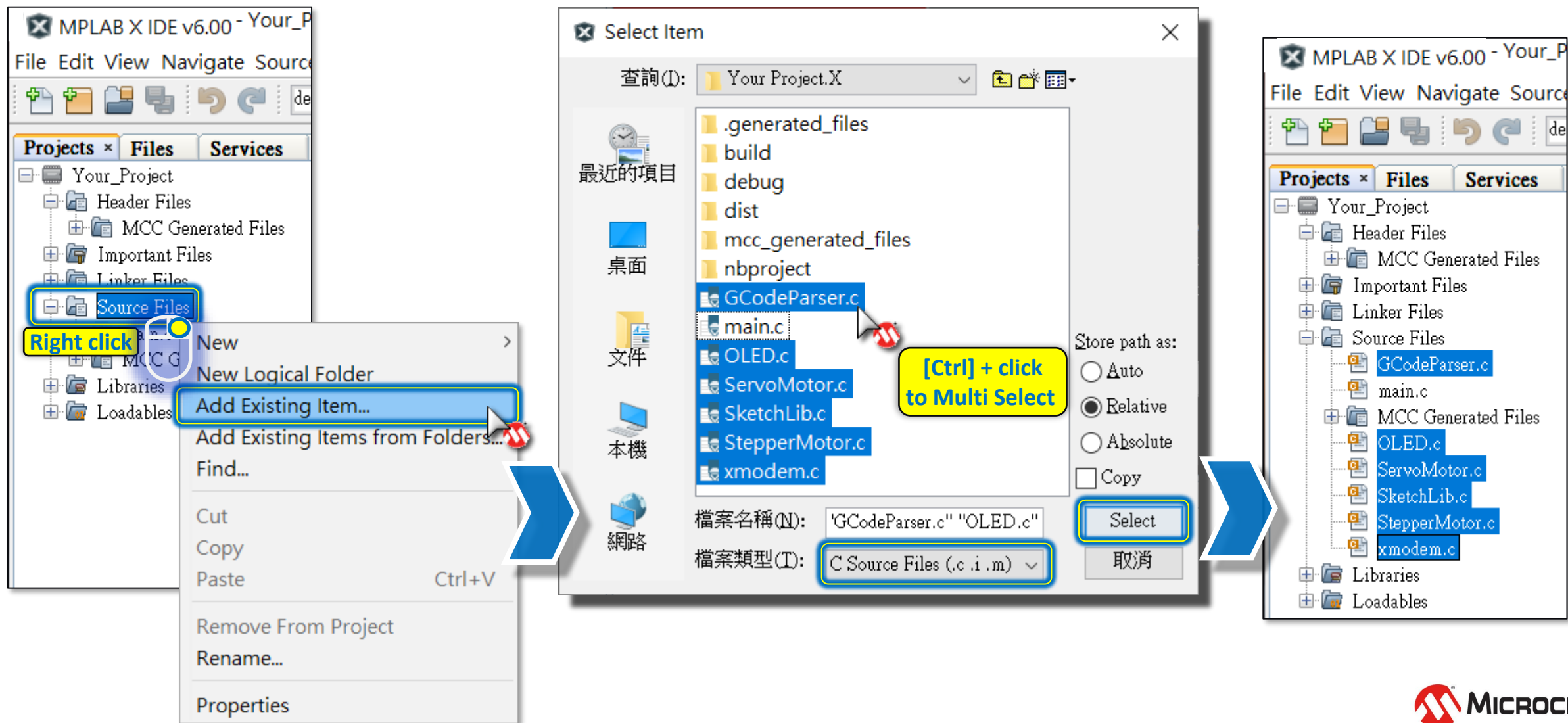
Lab Preparation

- Add *.h files to \[Your Project]\Header Files\



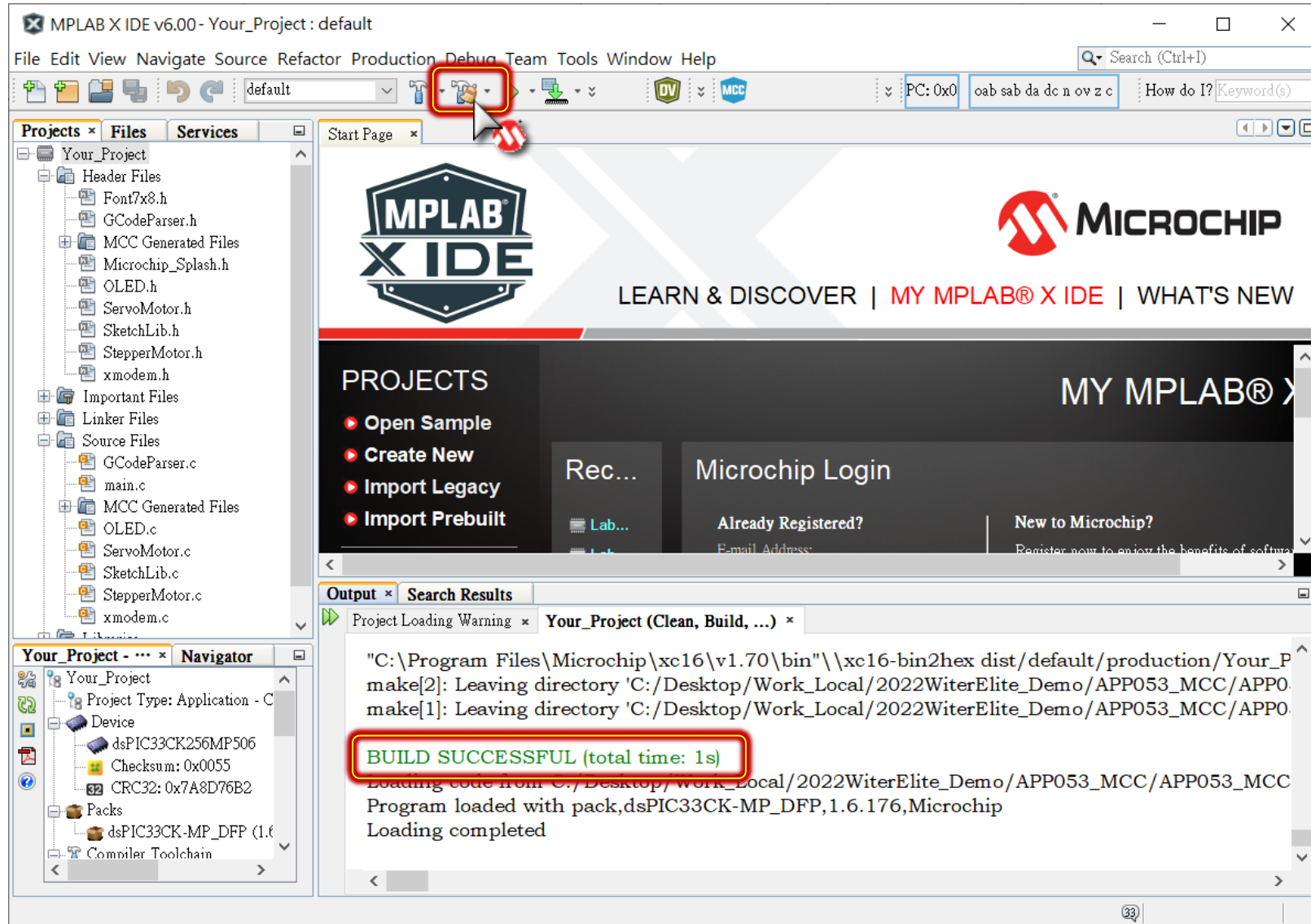
Lab Preparation

- Add *.c files (except main.c) to \[Your Project]\Source Files\



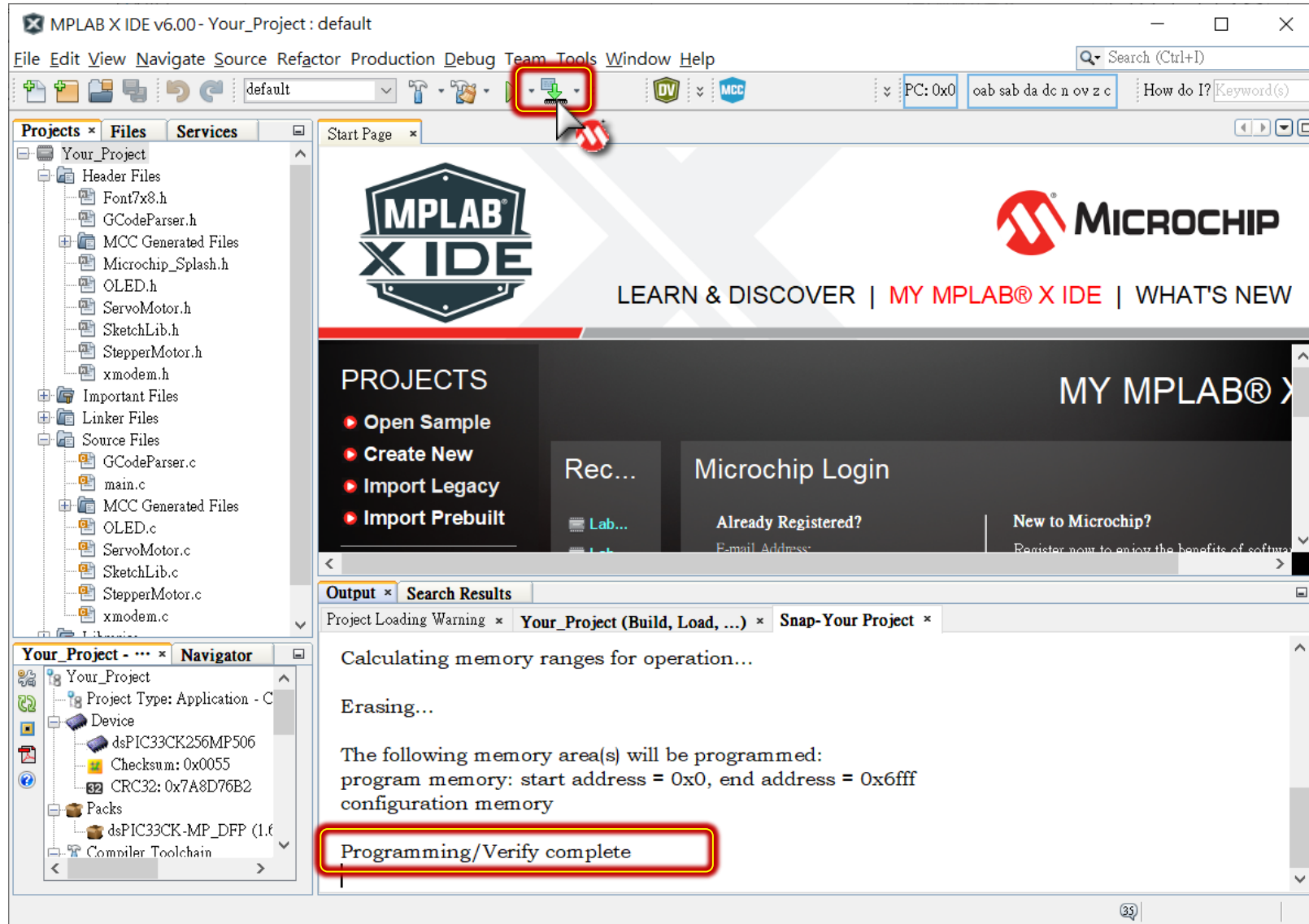
Lab Preparation

- Click  **Clean and Build** your project.



Lab Preparation

- Click  **Make and Program Device** to program your project to EVB.



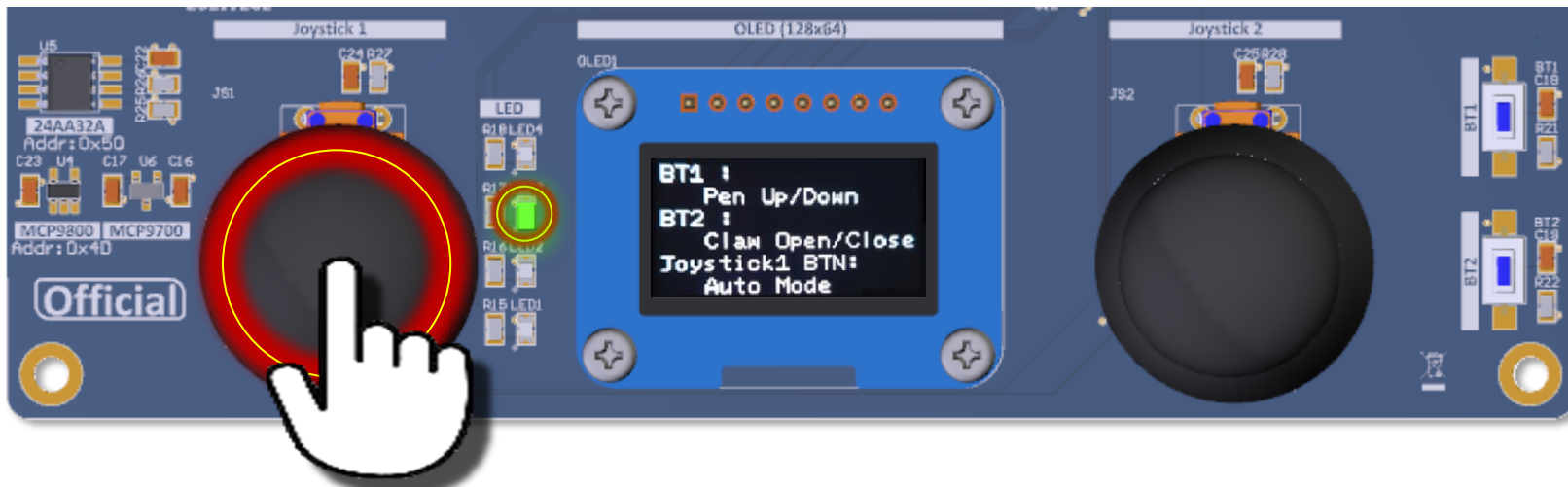
Function Test

Function Select (Auto or Manual Mode)

- After the Splash Screen, Push **Joystick 1 button** could toggle **Auto** and **Manual** Mode.



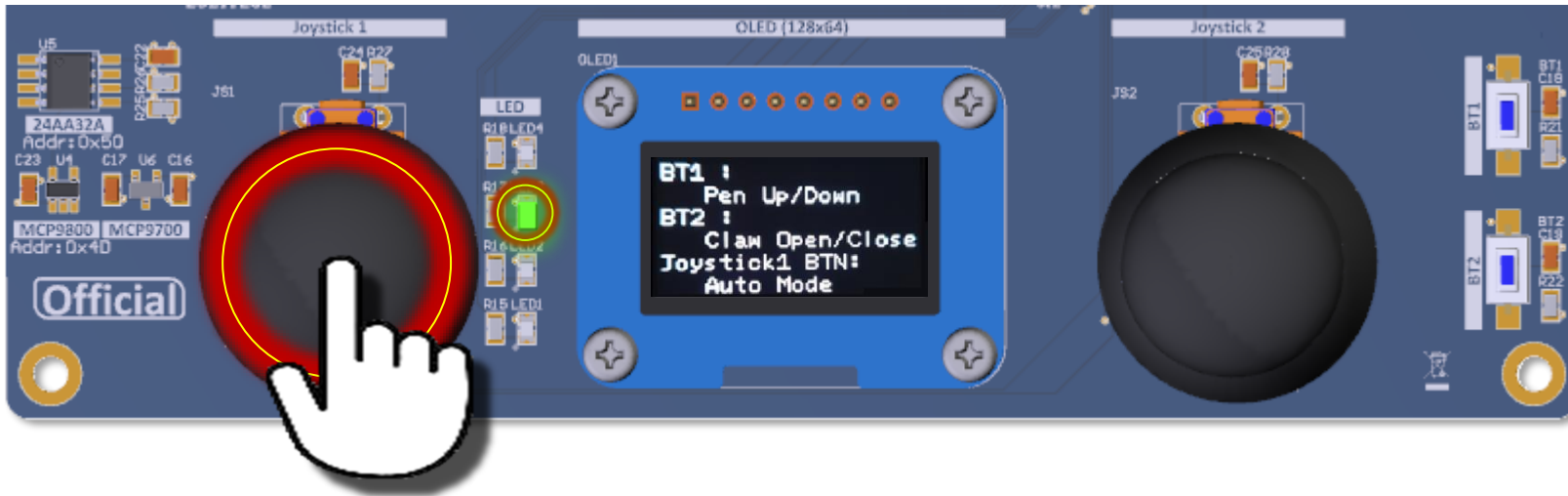
Initial State to
Auto Mode (LED3 OFF)



Click to Toggle
Manual Mode (LED3 ON)
And
Auto Mode (LED3 OFF)

Function Test (Auto Mode)

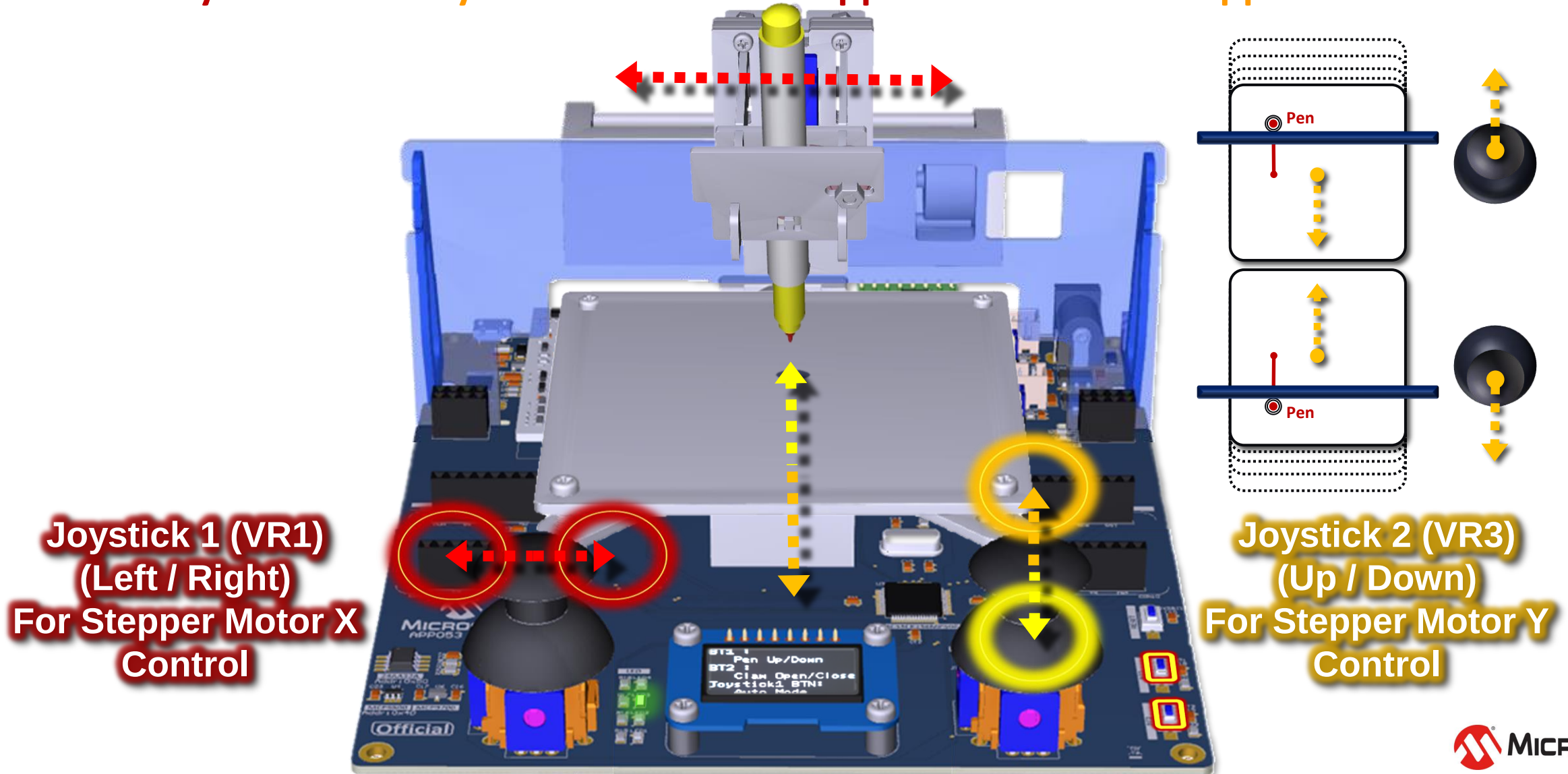
- Push **Joystick 1 button** and toggle to **Manual Mode**.



Push Joystick1 Button
and toggle to
Manual Mode (**LED3 ON**)

Function Test (Manual Mode)

- Move **Joystick 1** and **Joystick 2** to control **Stepper Motor 1** and **Stepper Motor 2**.

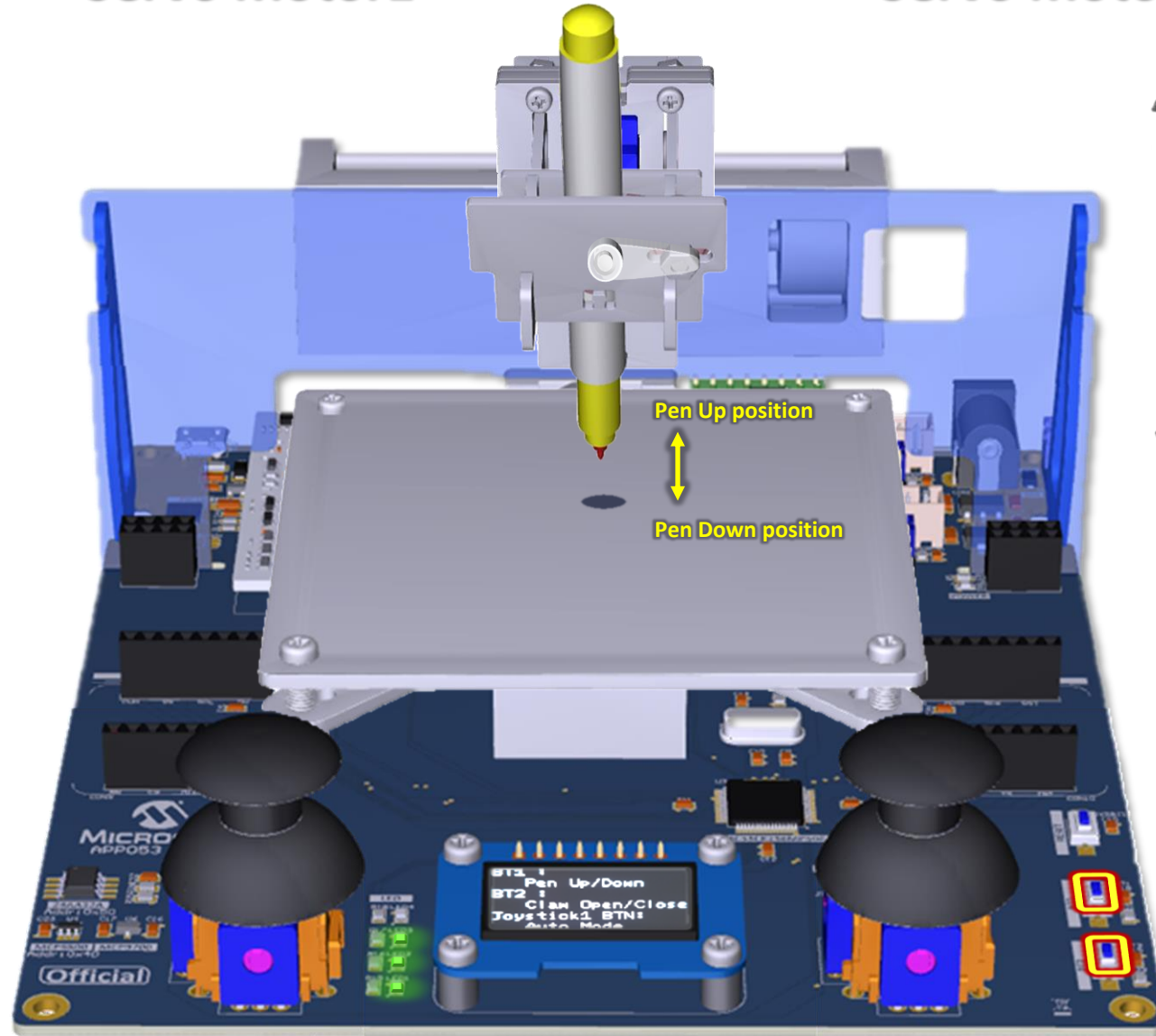


Function Test (Manual Mode)

- Push **BT1** to control **Servo Motor1** and **BT2** to control **Servo Motor2** (Reserved).

Pen Down
LED1 ON
Pen Up
LED1 OFF

Claw Close
LED2 ON
Claw Open
LED2 OFF

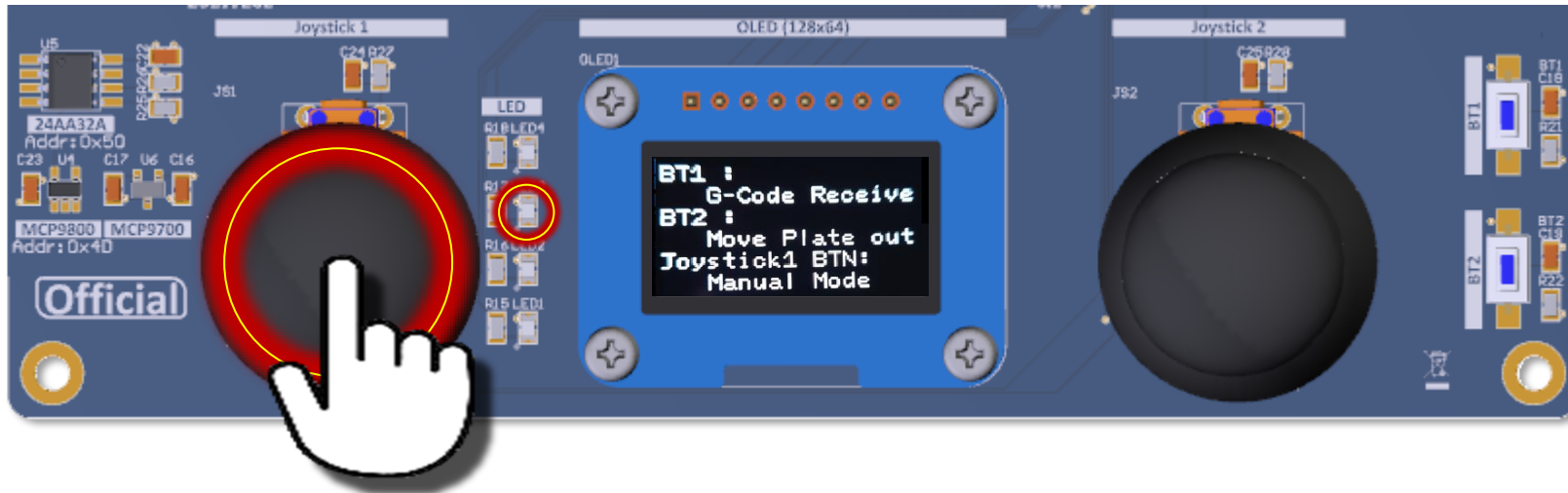


BT1 for
Servo Motor 1
(Pen Up / Down)

BT2 for
Servo Motor 2
(Claw Open / Close)

Function Test (Auto Mode)

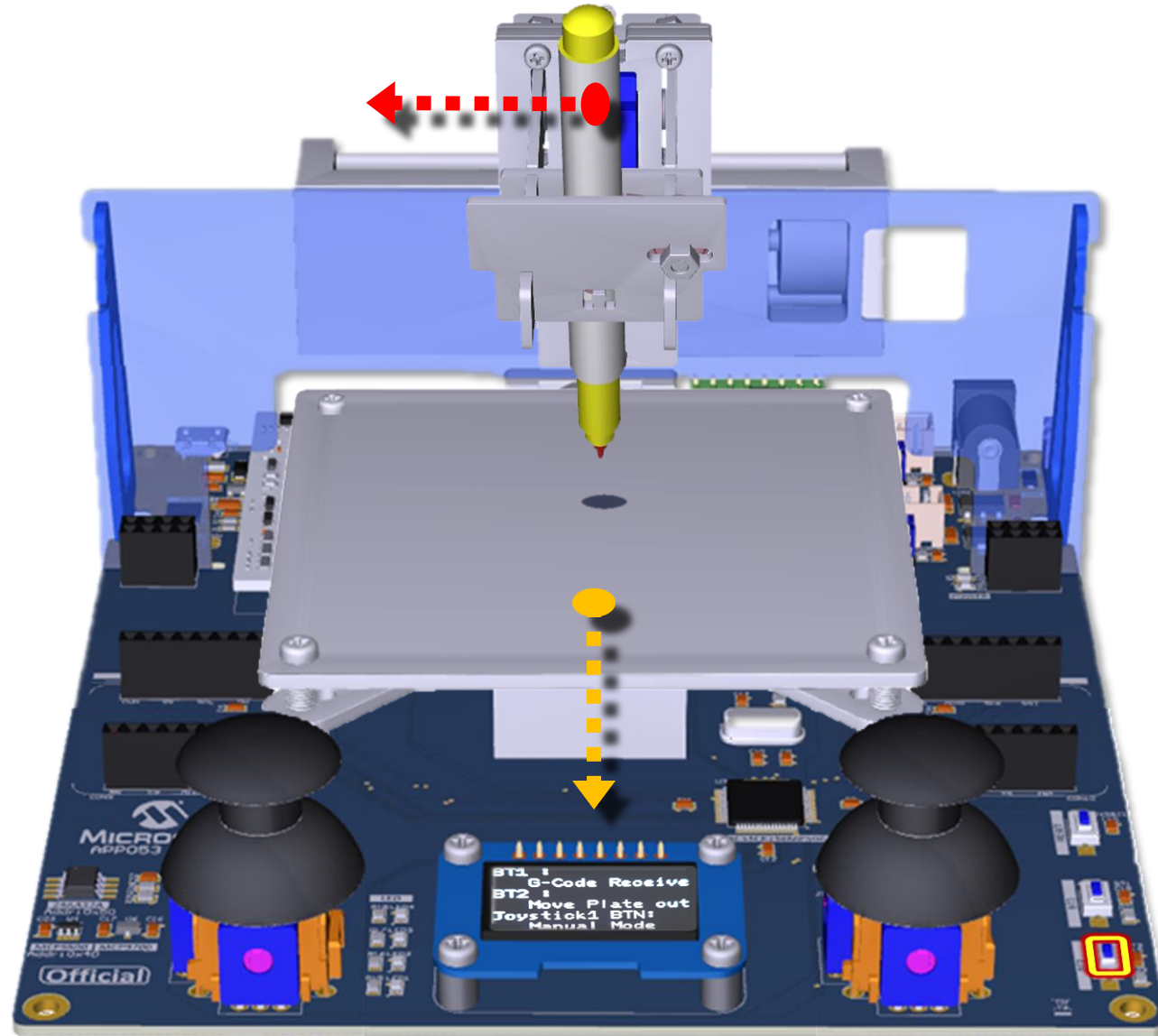
- Push **Joystick 1 button** and toggle to **Auto** Mode.



Push Joystick1 Button
and toggle to
Auto Mode (LED3 OFF)

Function Test (Auto Mode)

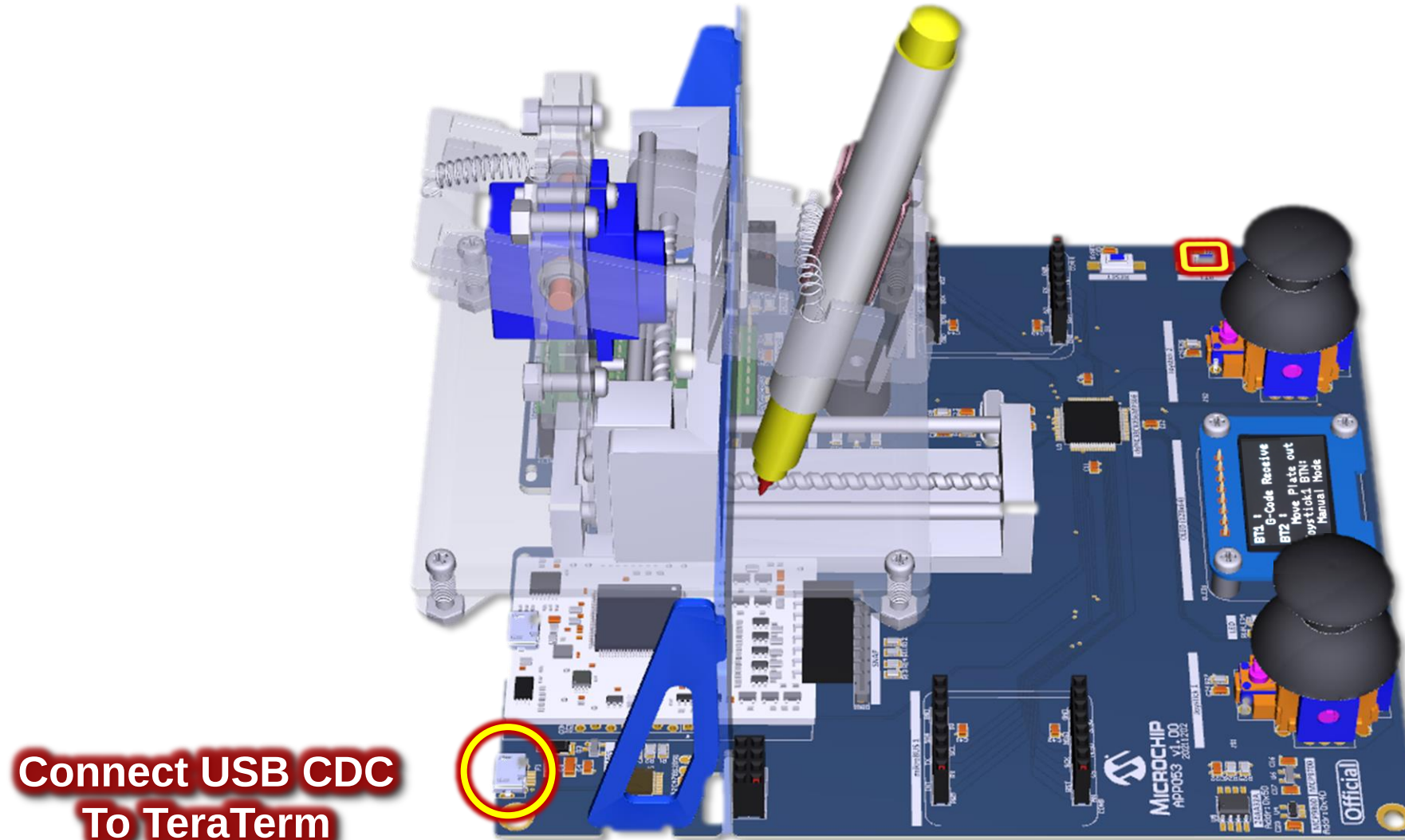
- Push **BT2** to move Pen Left and Draw Plate out for clean.



BT2 to move
Pen Left and
Draw Plate out

Function Test (Auto Mode)

- Push **BT1** to enter **X-Modem G-code receive mode**, connect **USB CDC** to PC **Tera Term**.

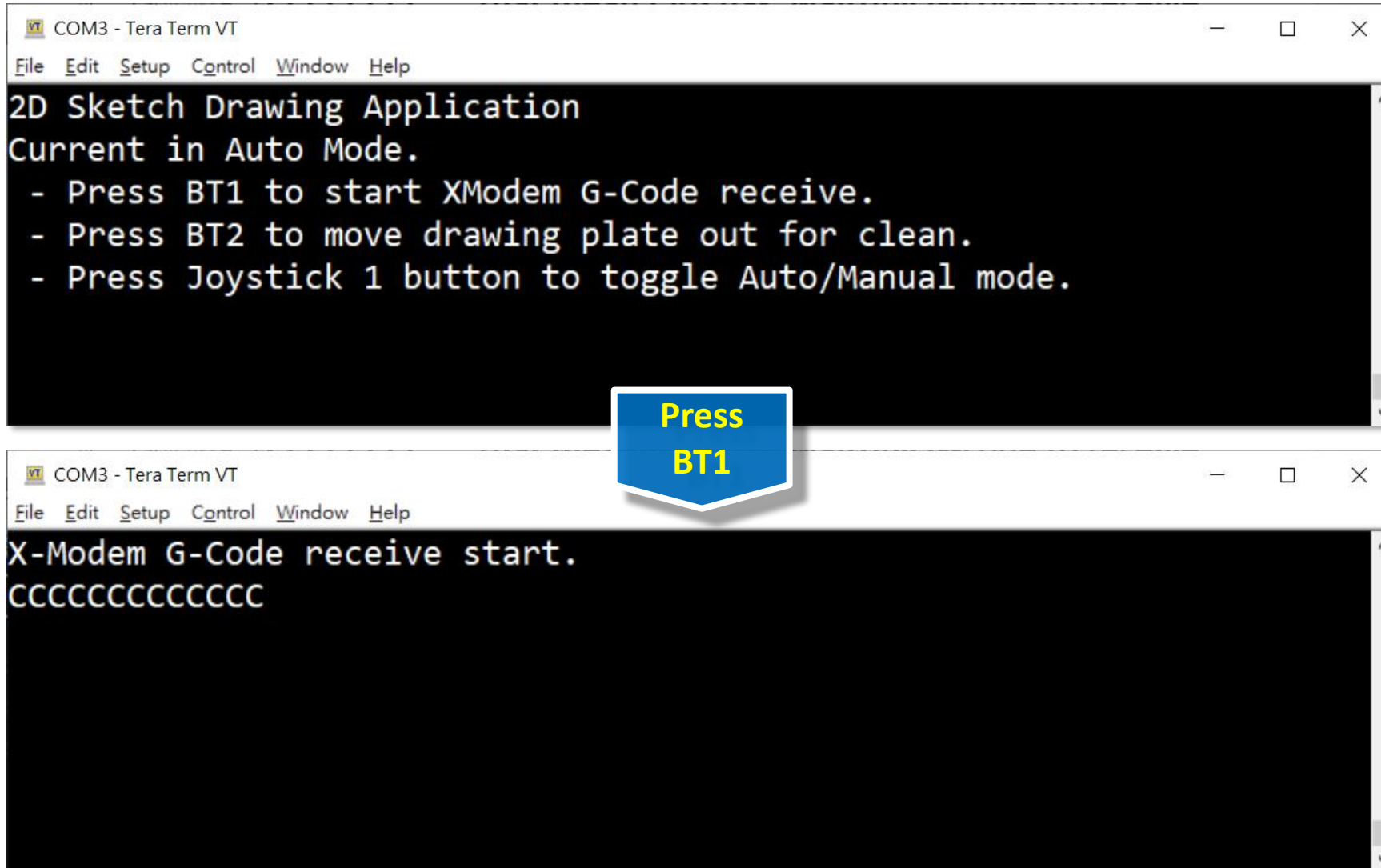


**BT1 to Enter
X - Modem
Receive Mode**

**Connect USB CDC
To TeraTerm**

Function Test (Auto Mode)

- PC **Tera Term** will show "**CCCCCCCCC...**" that means EVB has waiting G-code to receive.



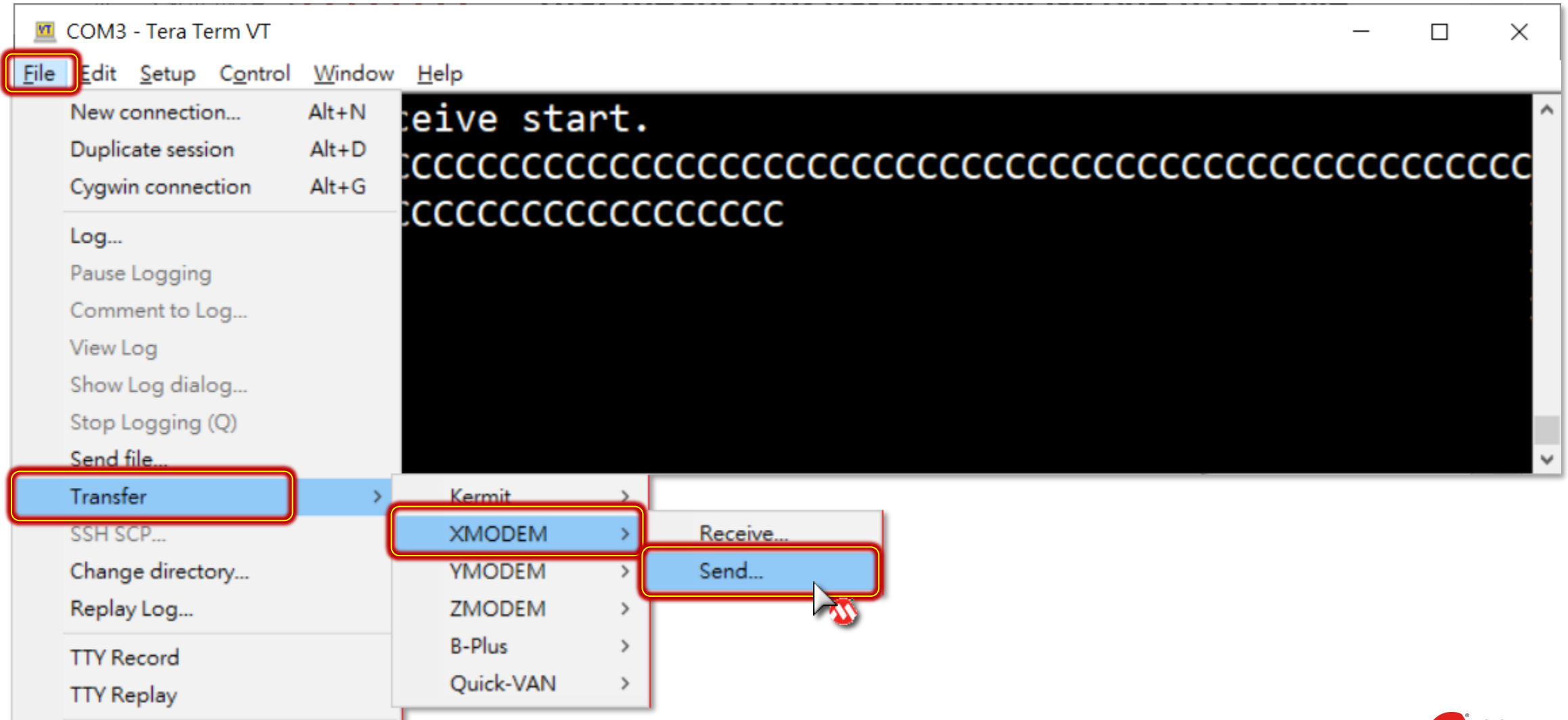
The image displays two screenshots of the Tera Term VT window, which is titled 'COM3 - Tera Term VT'. The window has a menu bar with 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The first screenshot shows the text '2D Sketch Drawing Application' and 'Current in Auto Mode.' followed by three instructions: '- Press BT1 to start XModem G-Code receive.', '- Press BT2 to move drawing plate out for clean.', and '- Press Joystick 1 button to toggle Auto/Manual mode.' A blue callout box with a white border and a drop shadow points to the first instruction, containing the text 'Press BT1'. The second screenshot shows the text 'X-Modem G-Code receive start.' followed by a line of ten 'C' characters, 'CCCCCCCCCCCC', indicating that the device is waiting for G-code.

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
2D Sketch Drawing Application
Current in Auto Mode.
- Press BT1 to start XModem G-Code receive.
- Press BT2 to move drawing plate out for clean.
- Press Joystick 1 button to toggle Auto/Manual mode.

COM3 - Tera Term VT
File Edit Setup Control Window Help
X-Modem G-Code receive start.
CCCCCCCCCCCC
```

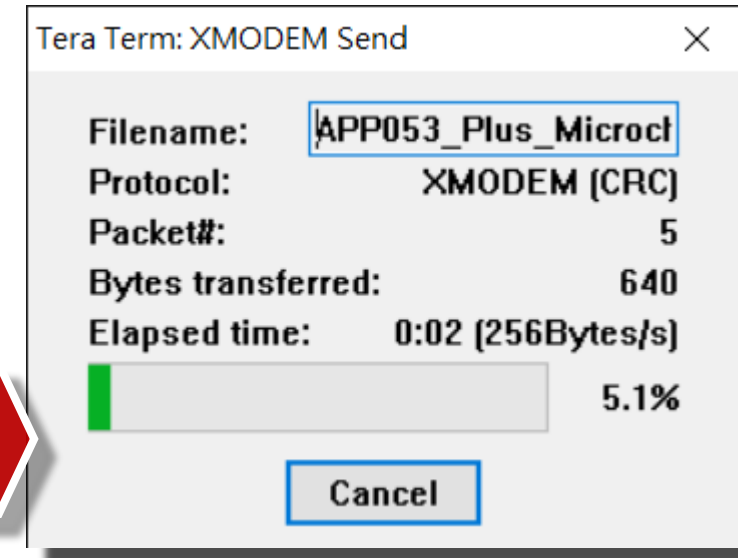
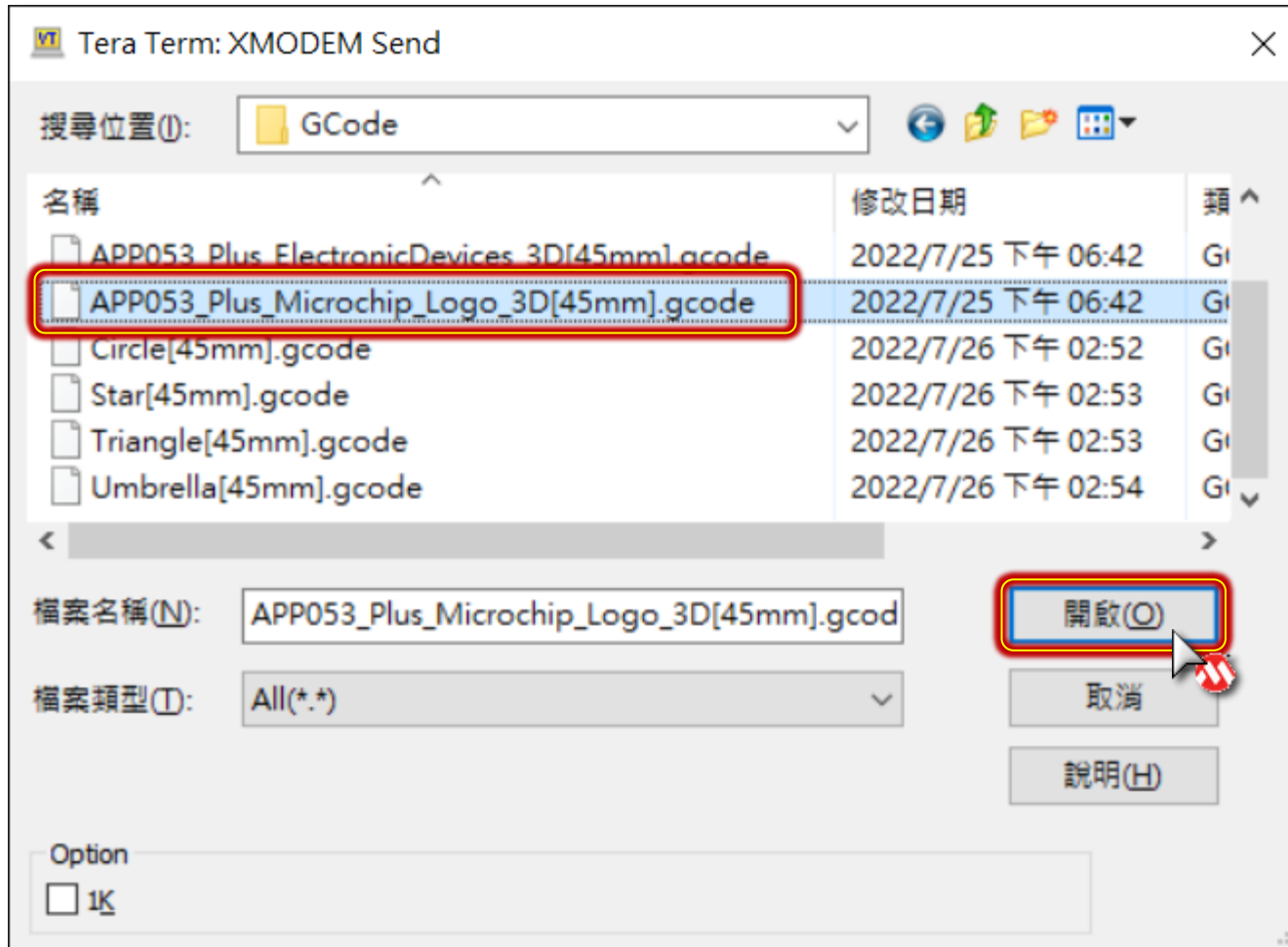
Function Test (Auto Mode)

- Select **File** > **Transfer** > **XMODEM** > **Send...** in **Tera Term**



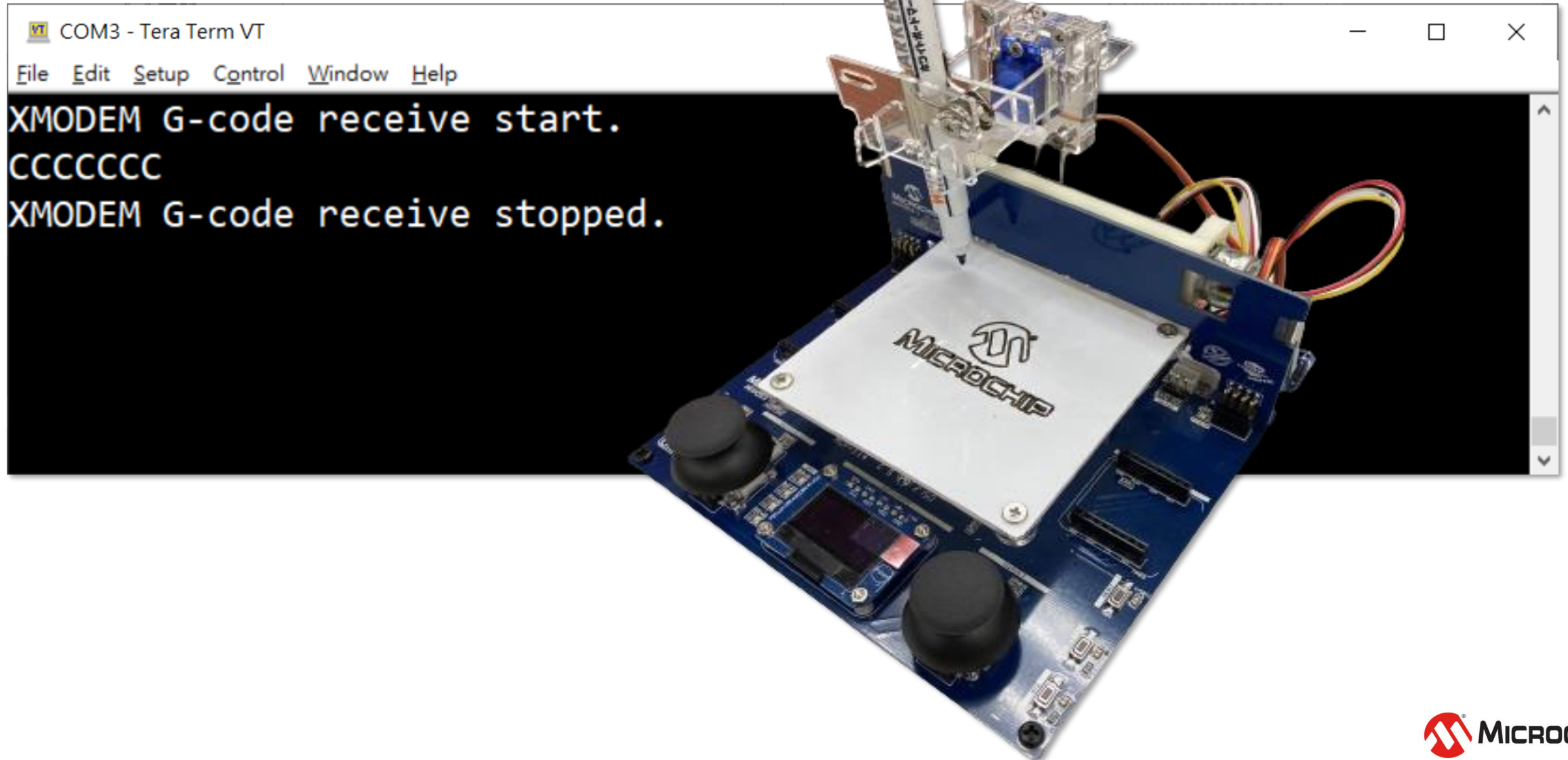
Function Test (Auto Mode)

- Select a G-Code to output and start drawing.



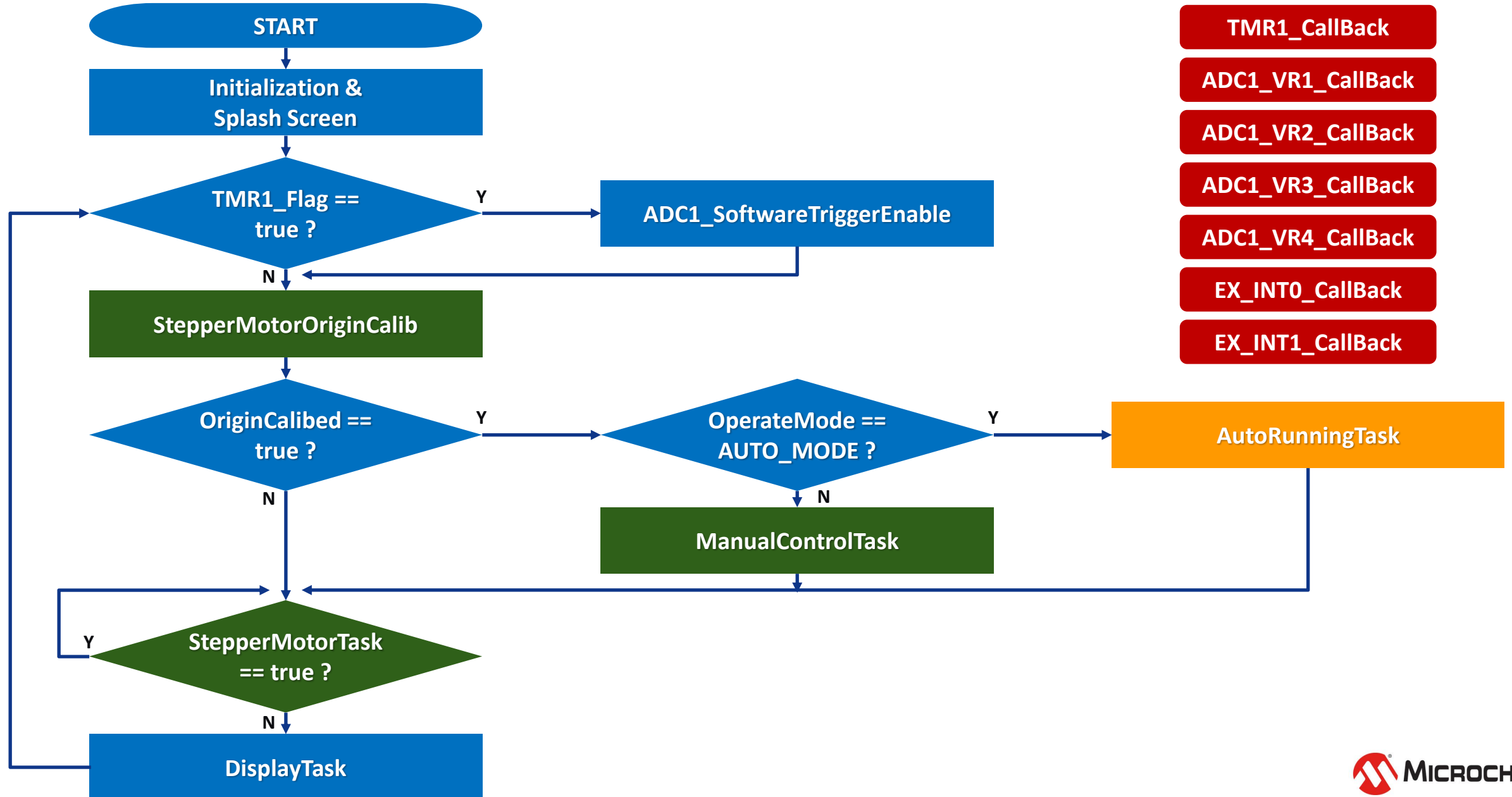
Function Test (Auto Mode)

- Result example.



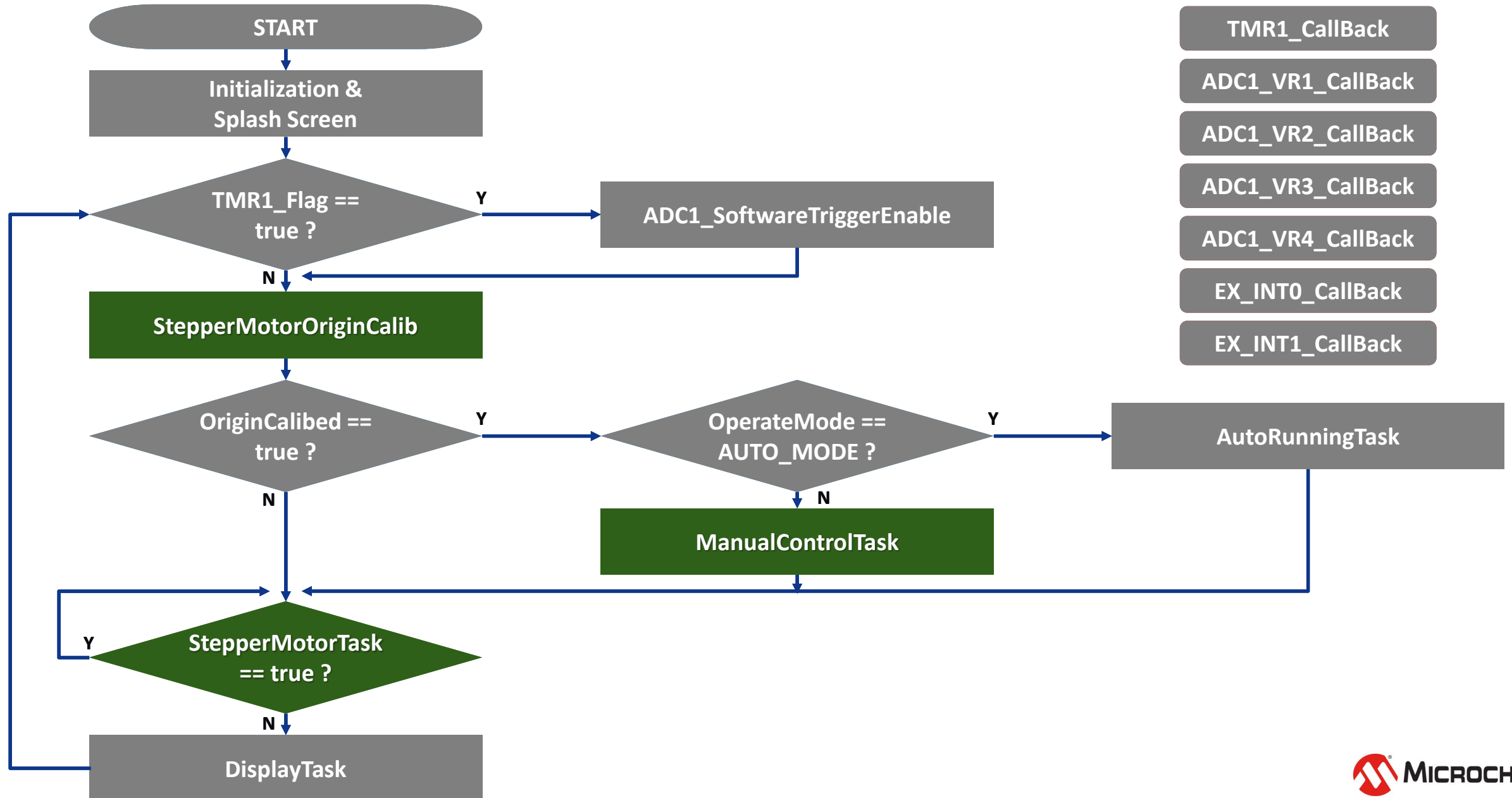
Function Flow Chart

Main Flow (main.c)

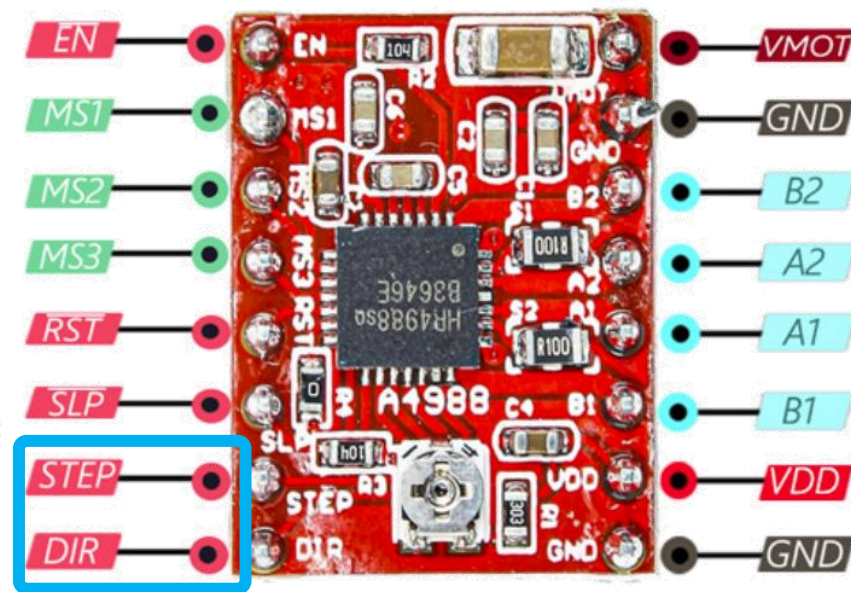
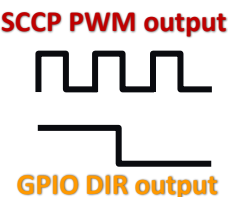
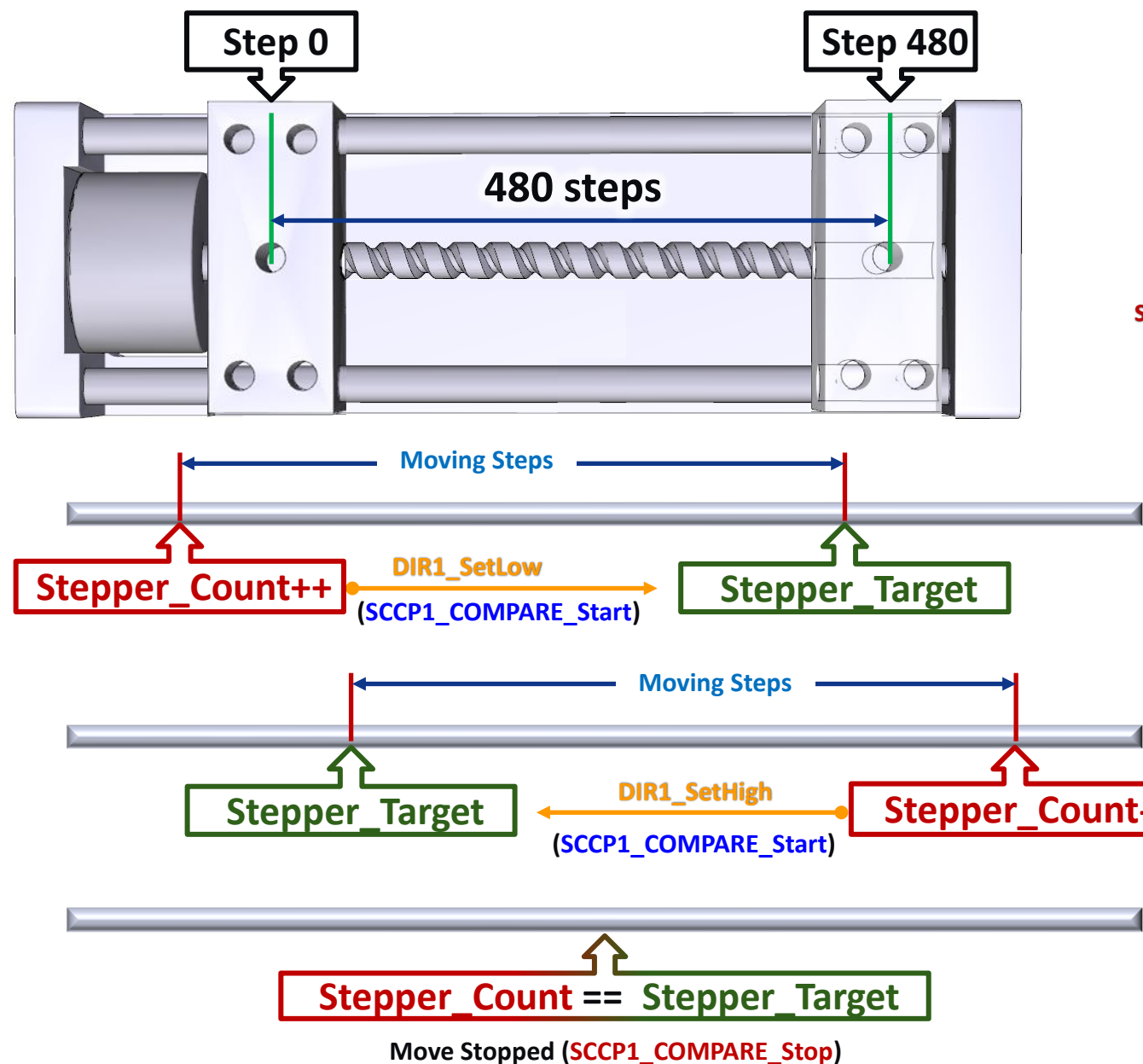


Stepper Motor Control

Main Flow relate of Stepper Motor (main.c)



Stepper Motor Control(Summer Elite)



SCCPx_COMPARE_CallBack
Stepper_Count++ or Stepper_Count--

Moving Direction = DIR_UP

Moving Steps = Target - Step

Moving Direction = DIR_DN

Stepper Motor Motion Control

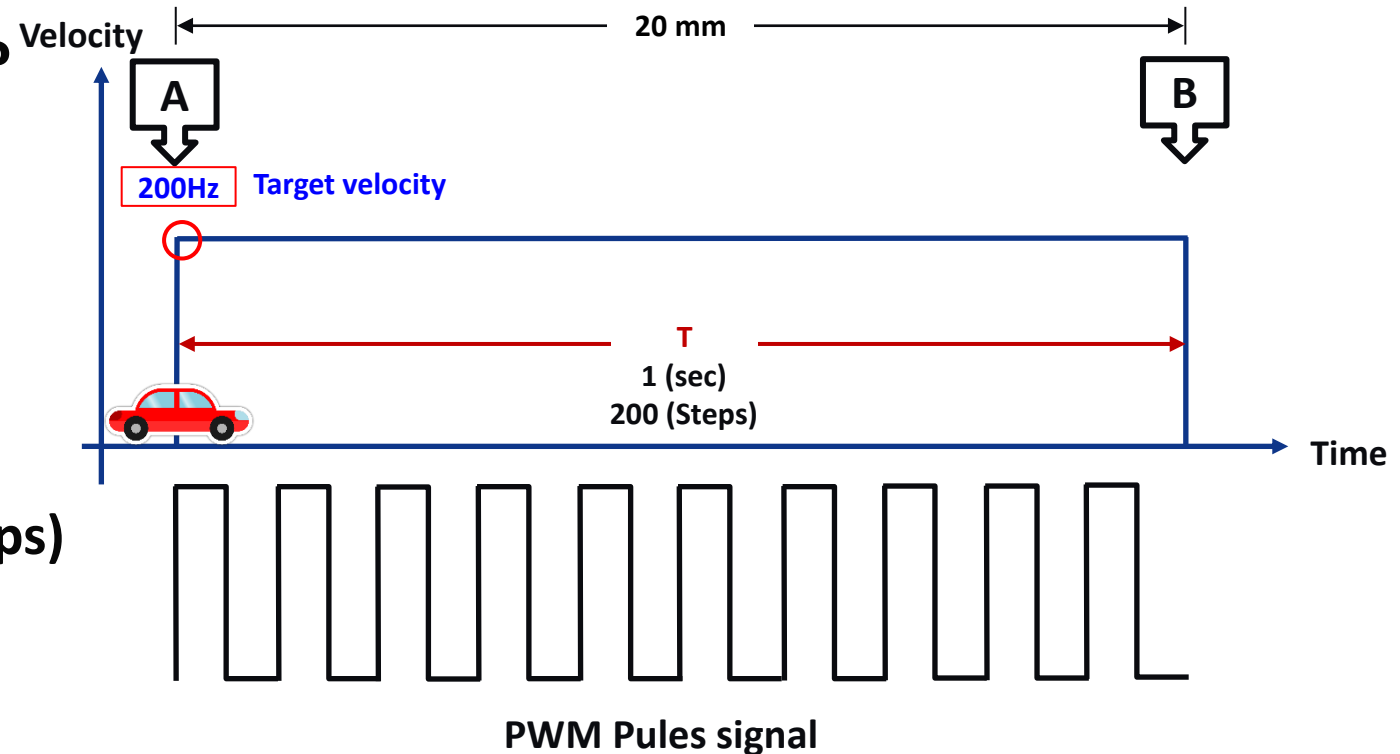
- Motor Specifications :
 - Step angle : 18°
 - Steps per revolution : $360^\circ \div 18^\circ = 20$ (Steps/rev)
 - Screw pitch : 2 (mm/rev)
 - Maximum starting pulse rate(fs) : 250 Hz
 - Maximum slewing pulse rate(fr) : 1 kHz

Q : What's the time **T** take from A to B?

- A moves B distance : 20 mm
- Target pulse rate(velocity) : 200 Hz

A :

- Number of revolutions of the motor :
 $\Rightarrow 20 \text{ (mm)} \div 2 \text{ (mm/rev)} = 10 \text{ (rev)}$
- Circle conversion Steps :
 $\Rightarrow 10 \text{ (rev)} \times 20 \text{ (Steps/rev)} = 200 \text{ (Steps)}$
- Move time :
 $200 \text{ (Steps)} \div 200 \text{ (Hz)}$
 $= 200 \text{ (pulse)} \div 200 \text{ (pulse/sec)}$
 $= \mathbf{1.0 \text{ (Sec)}} = \mathbf{T}$



Stepper Motor Motion Control

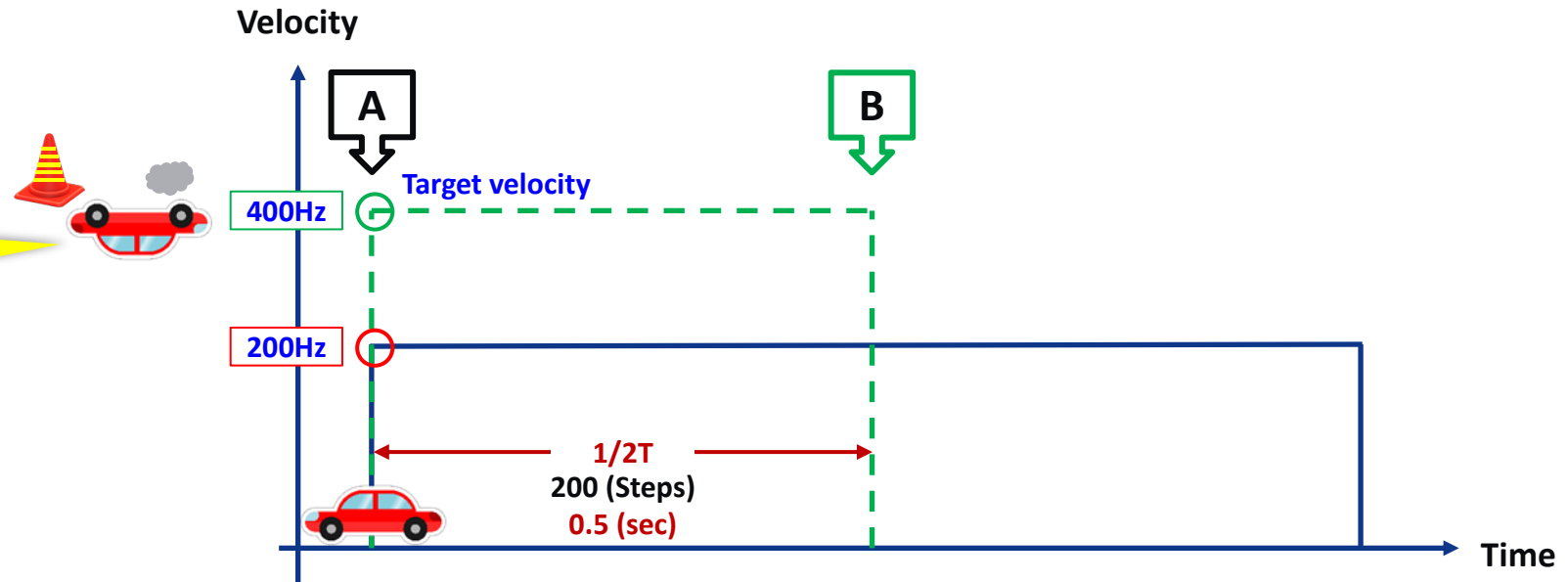
Q : How to reduce movement time **T** from **1.0 second** to **0.5 second** ?

- Hint : Thinking between PWM frequency and the motor velocity.

A : Increase target velocity :

New Target pulse rate(Velocity) : 200 (Steps) ÷ 0.5 (Sec) = 400 (pulse/sec) = 400 Hz

**stepper motor
fall out of step**



- 400 Hz greater than 250 Hz is out of range.
- The Maximum starting pulse rate (f_s) of the stepper motor is 250Hz.

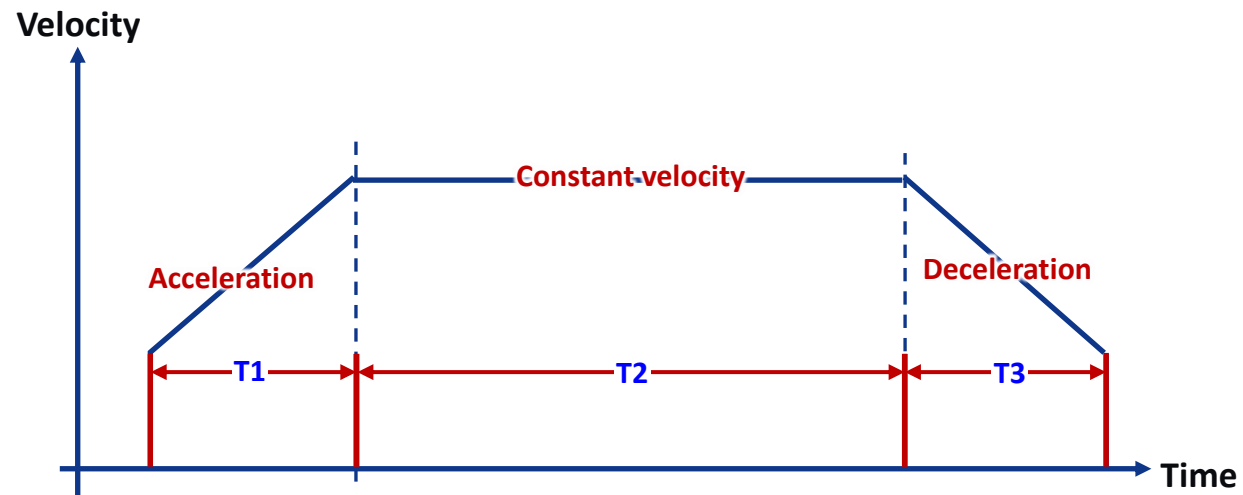
Stepper Motor Motion Control

- Q : How to solve the motor fall out of step?
 - Hint : Confirm motor starting pulse rate at 200Hz but can reach top velocity to 400Hz.
- Introduction to Trapezoidal Velocity Curve :
 - When the system needs to change the speed of the motor, the trapezoidal velocity curve can realize the corresponding control function.
 - In generally, the acceleration of the trapezoidal velocity curve not change during the movement, so the acceleration and deceleration curves are the same.
 - Trapezoidal velocity curve divides the entire motion process into three stages:

Acceleration (T1)

Constant velocity (T2)

Deceleration (T3)



Stepper Motor Motion Control

- Q : How to apply a trapezoidal velocity curve?
 - Plan Acceleration(equal to Deceleration), Constant velocity time.

- A :
 - Calculate target velocity :
 $\Rightarrow 200 \text{ (Steps)} \div (0.5 - 0.1) = 500 \text{ Hz}$
 - Calculate acceleration :
PWM Initial velocity : 200 Hz
Acceleration (in Frequency domain)
 $= (500\text{Hz} - 200\text{Hz}) \div 0.1 \text{ (sec)}$

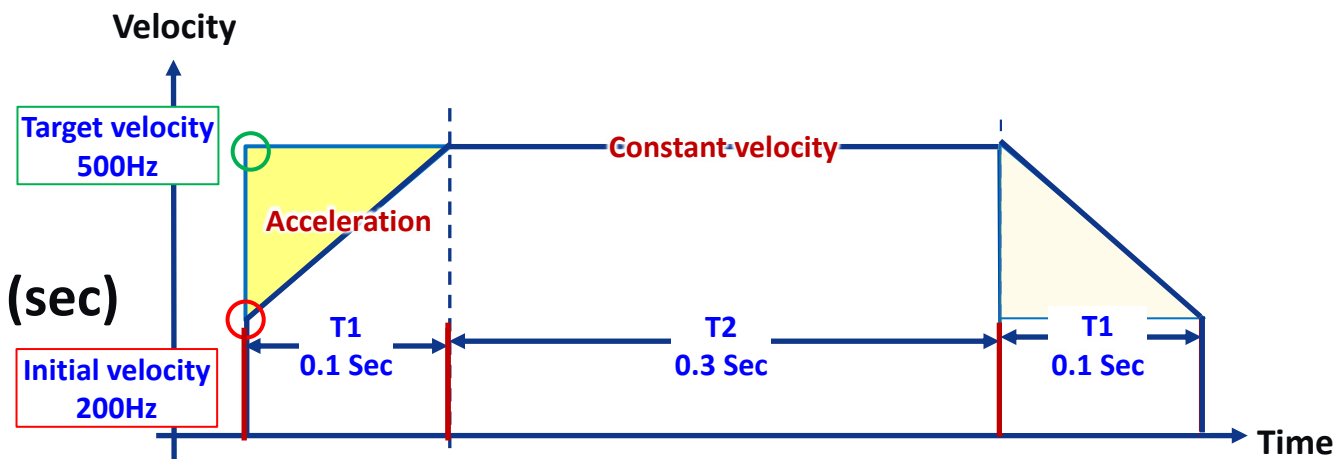
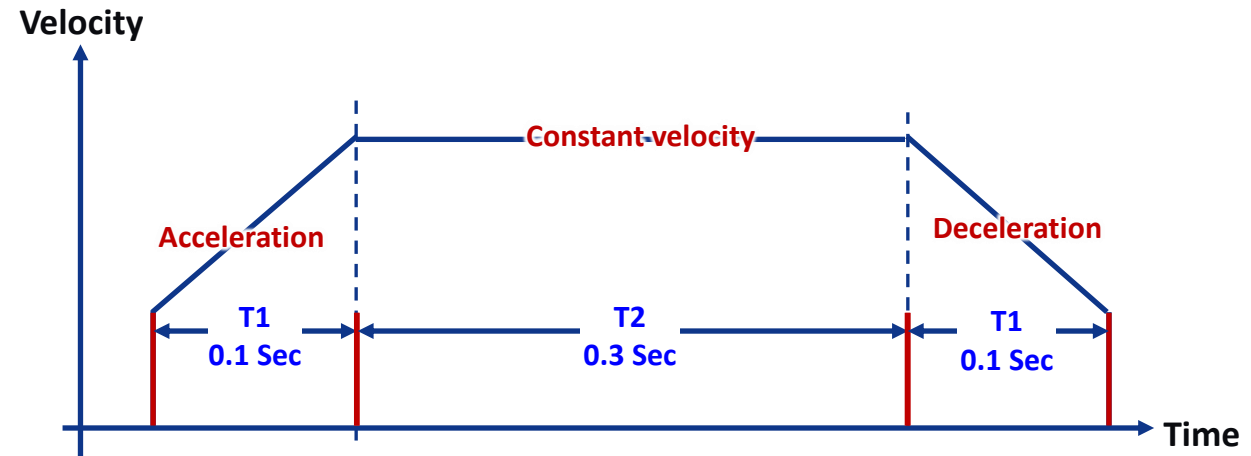
$$\text{Period Count} = \left(\frac{\text{Clock Source}}{\text{Frequency}} \right)$$

Acceleration (In Time period count)

$$= ((8\text{MHz}/500\text{Hz}) - (8\text{MHz}/200\text{Hz})) \div 0.1 \text{ (sec)}$$

$$= (16000 - 40000)(\text{counts}) \div 100 \text{ (ms)}$$

$$= -240 \text{ (counts/ms)}$$



Stepper Motor Motion Control

- Trapezoidal Command Curve Conditions :

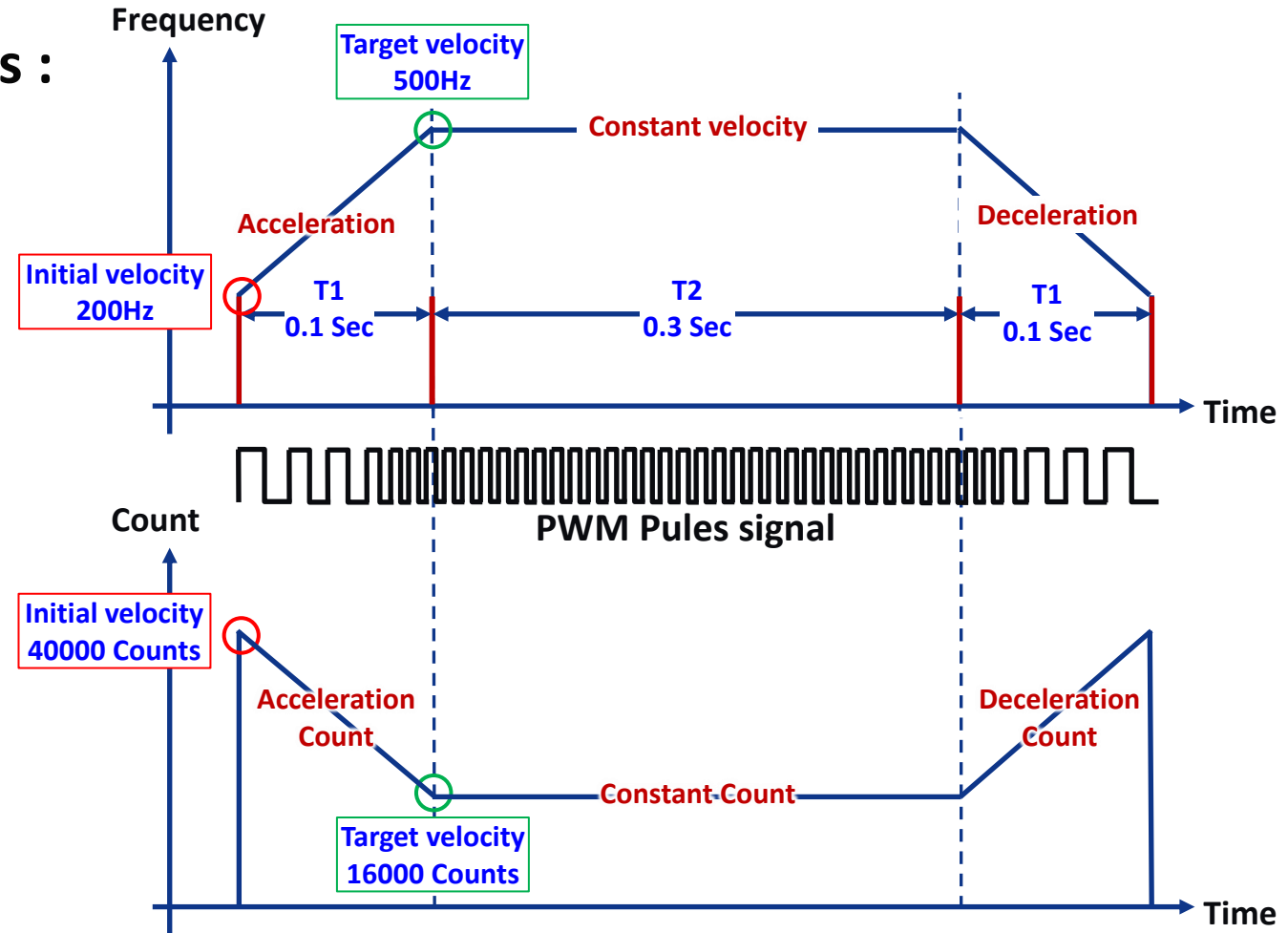
- Move time : 0.5 Sec
- PWM Initial velocity : 200 Hz
- PWM Target velocity : 500 Hz
- Acceleration : -240 Count/ms
- Deceleration : +240 Count/ms

- MCU PWM frequency by Count status :

- PWM Initial Count :
- $\Rightarrow 8 \text{ MHz} \div 200 \text{ Hz} = 40000 \text{ (Count)}$
- PWM Target Count :
- $\Rightarrow 8 \text{ MHz} \div 500 \text{ Hz} = 16000 \text{ (Count)}$
- Acceleration Count : --240 Count
- Deceleration Count : ++240 Count

- PWM frequency by Count changes :

- 40000 —1ms—> 39760 —2ms—> —99ms—> 16240 —100ms—> 16000 (Acceleration)
- 16000 —101ms—> 16000 —102ms—> —399ms—> 16000 —400ms—> 16000 (Constant)
- 16000 —401ms—> 16240 —402ms—> —499ms—> 39760 —500ms—> 40000 (Deceleration)



Stepper Motor Motion Control

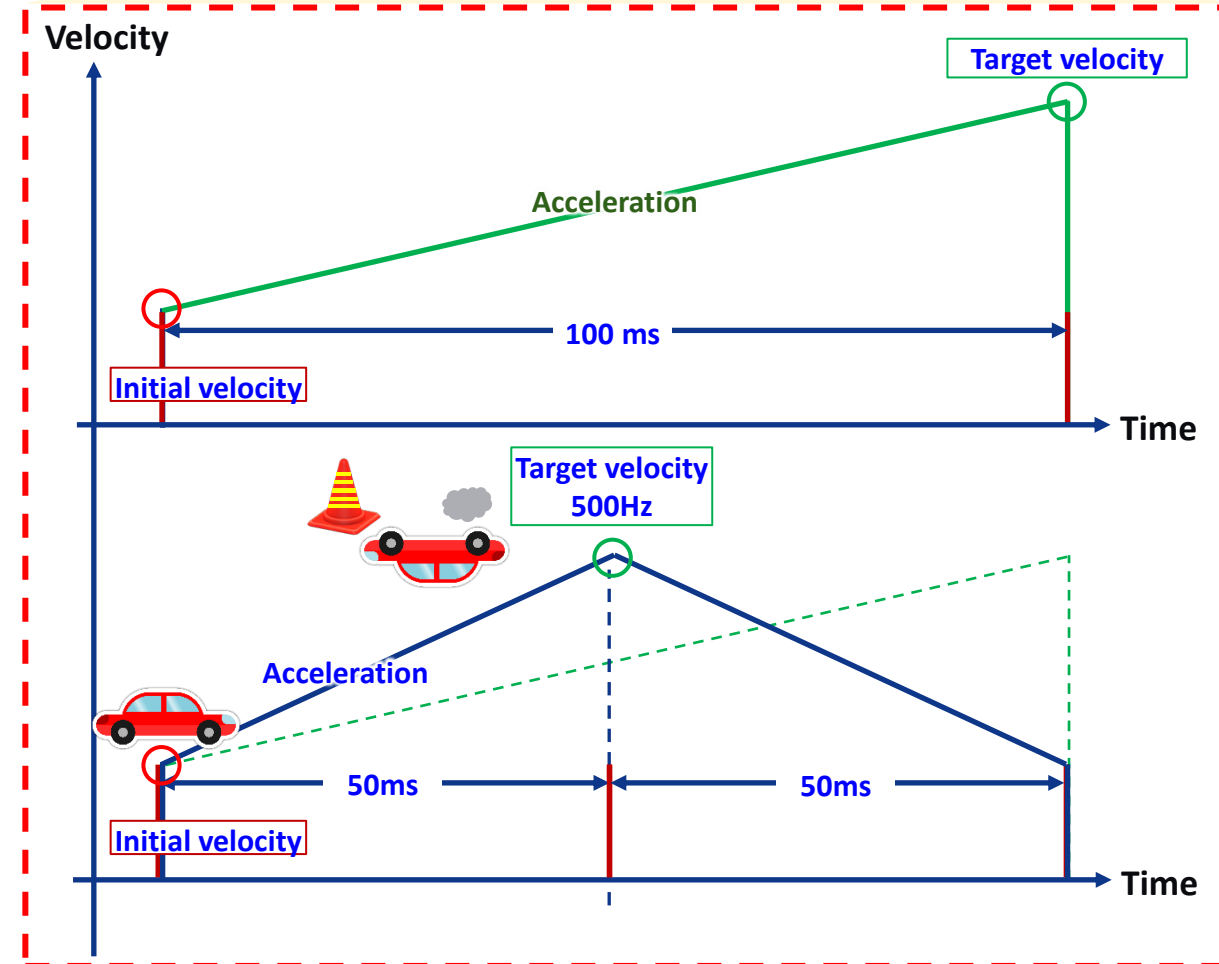
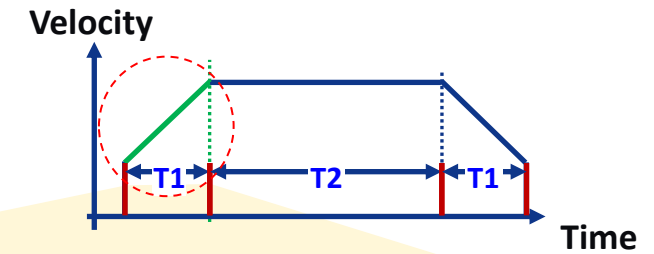
- Q : If total moving time should finish within 100ms.

- PWM Initial Count : 200 Hz => 40000 Counts
- PWM Target Count : 500 Hz => 16000 Counts
- Acceleration Count : -240 Count/ms
- Acceleration time =
 $(16000 - 40000) \div -240 \text{ Count/ms} = 100 \text{ ms}$

- A : To increase acceleration directly :

- When acceleration equals to deceleration.
- Acceleration time = $100\text{ms} \div 2 = 50 \text{ ms}$
- Acceleration Count
 $= (16000 - 40000) \div 50\text{ms} = -480 \text{ Count/ms}$

**stepper motor
fall out of step**



Stepper Motor Motion Control

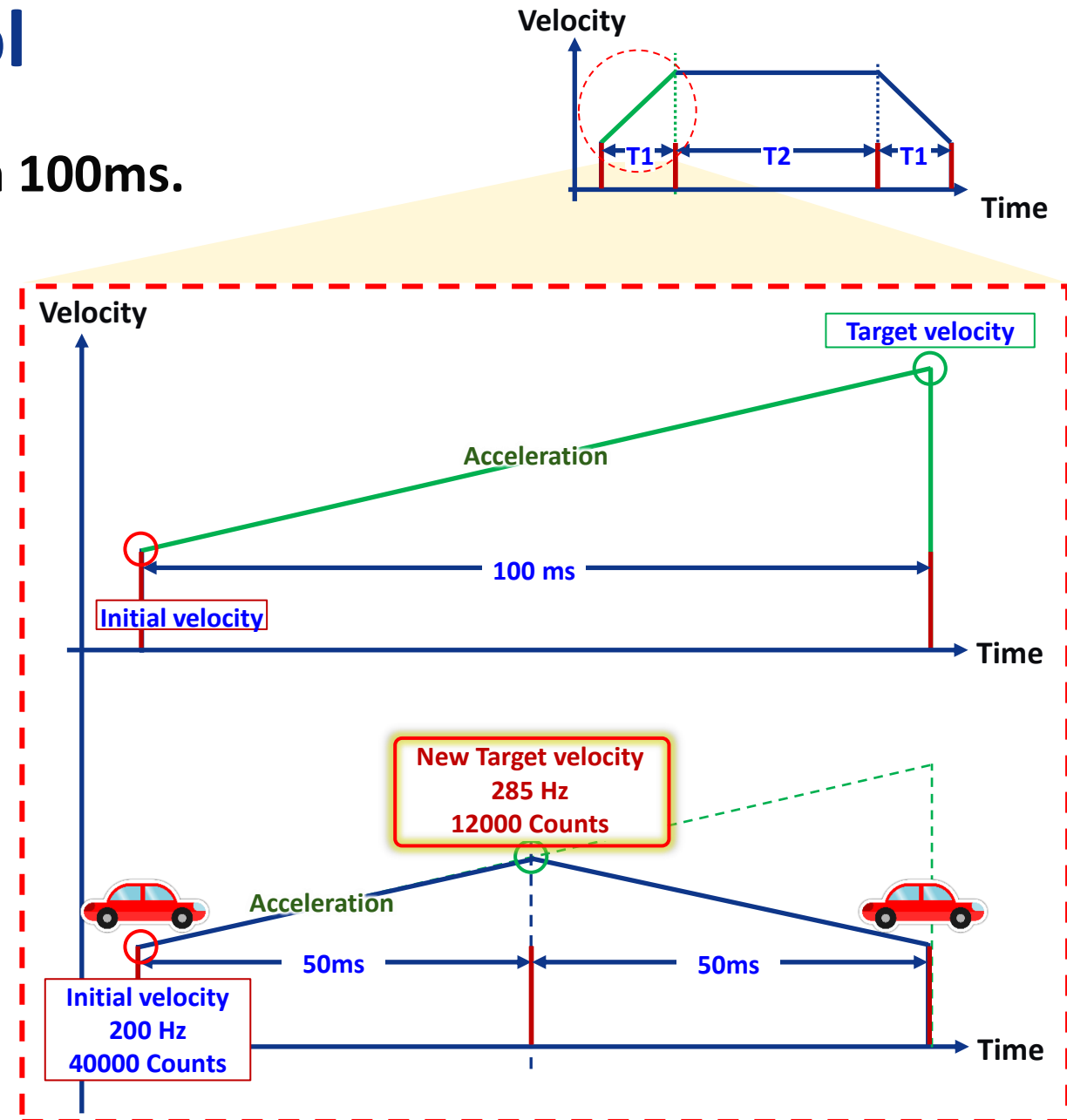
- Q : If total moving time should finish within 100ms.

- PWM Initial Count : 200 Hz => 40000 Counts
- PWM Target Count : 500 Hz => 16000 Counts
- Acceleration Count : -240 Count/ms
- Acceleration time =
 $(16000 - 40000) \div -240 \text{ Count/ms} = 100 \text{ ms}$

Hint 1 : Keep the original acceleration.

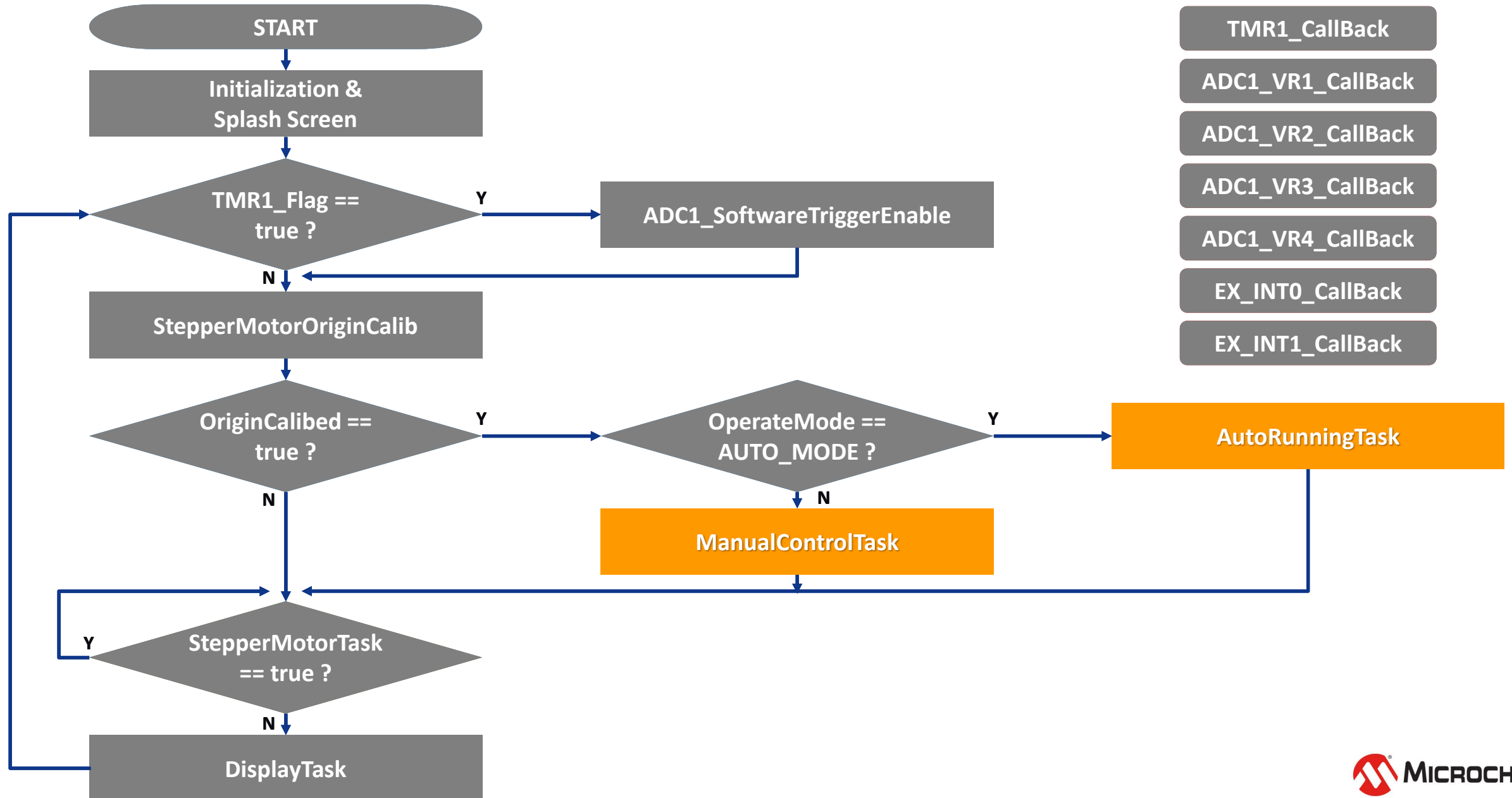
Hint 2 : Calculate the new target velocity.

- A : To recalculate target velocity by using original acceleration **240 (Count/ms)**:
 - The New Target velocity by Count
 $= 50 \text{ ms} \times 240 \text{ (Count/ms)} = 12000 \text{ Count}$
 - The New Target velocity by PWM frequency
 $= 8 \text{ MHz} \div (40000 - 12000) = 285 \text{ Hz}$



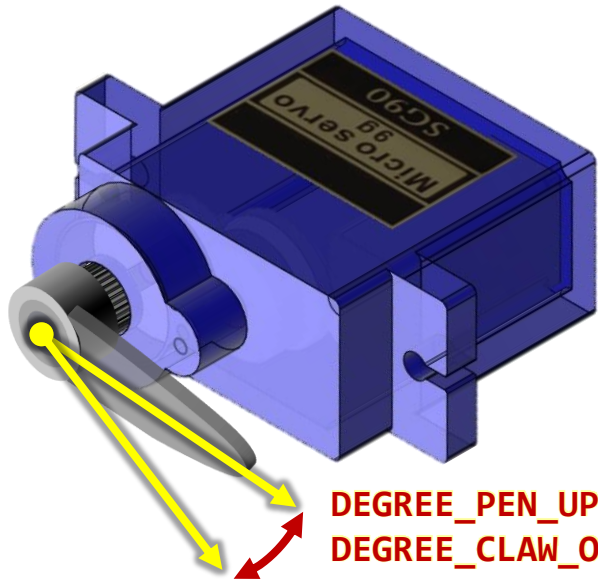
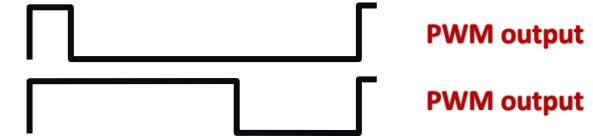
Servo Motor Control

Main Flow relate of Servo Motor (main.c)



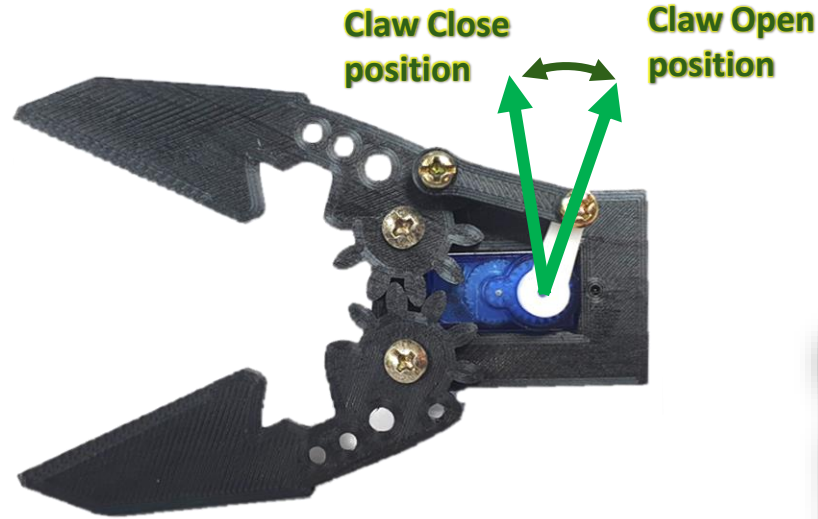
Servo Motor Control Task (ServoMotor.c / .h)

```
#define DEGREE_000_DUTY      (12.5)  // Duty 12.5% @ 250Hz(4ms)
#define DEGREE_180_DUTY     (62.5)  // Duty 62.5% @ 250Hz(4ms)
#define DEGREE_PEN_DOWN     (0)      // Pen Down   Servo Motor Degree
#define DEGREE_PEN_UP       (10)     // Pen Up     Servo Motor Degree
#define DEGREE_CLAW_CLOSE   (0)      // Claw Close Servo Motor Degree
#define DEGREE_CLAW_OPEN    (10)     // Claw Open  Servo Motor Degree
#define SERVO_MOTOR_1_DELAY 200      // Pen Delay
#define SERVO_MOTOR_2_DELAY 200     // Claw Delay
```



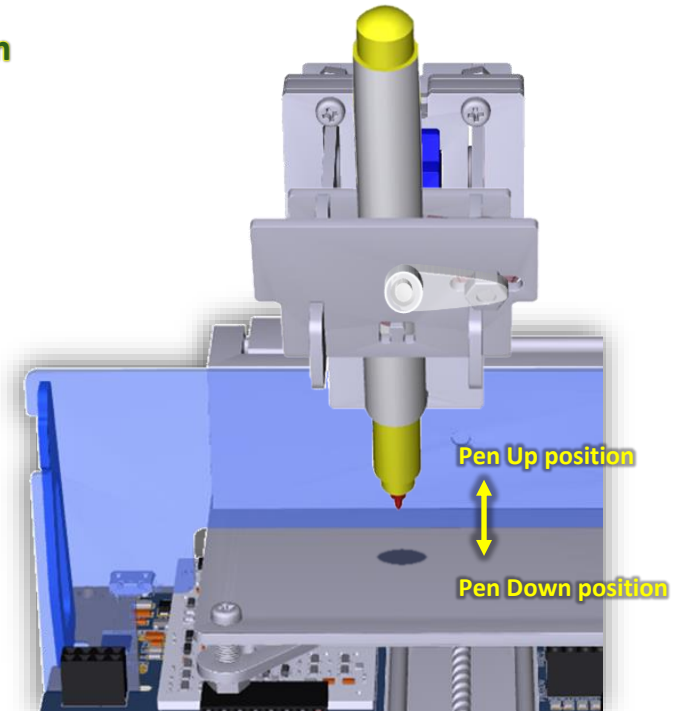
DEGREE_PEN_UP (10)
DEGREE_CLAW_OPEN (10)

DEGREE_PEN_DOWN (0)
DEGREE_CLAW_CLOSE (0)



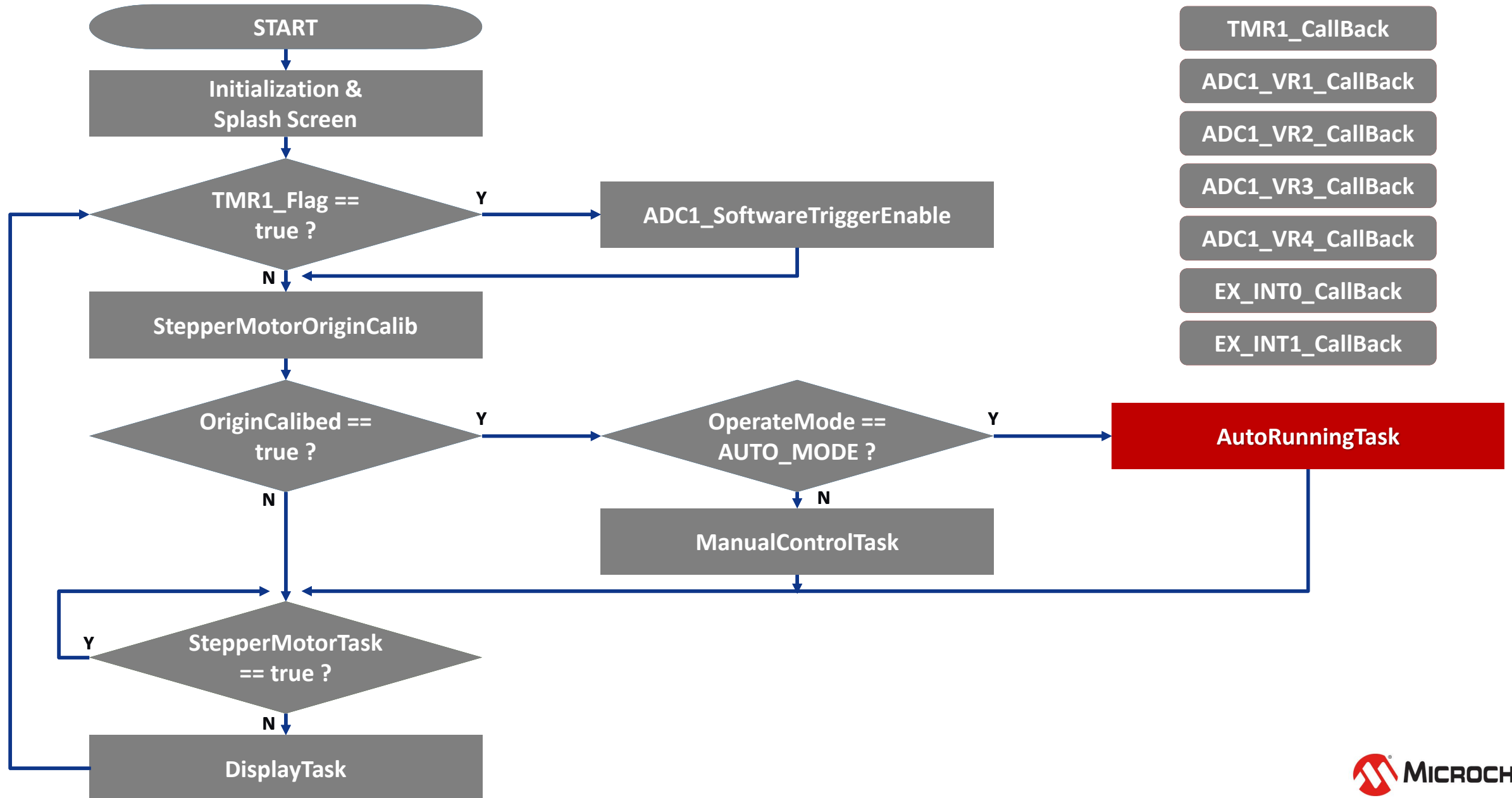
PWM output Duty 15.3%

PWM output Duty 12.5%

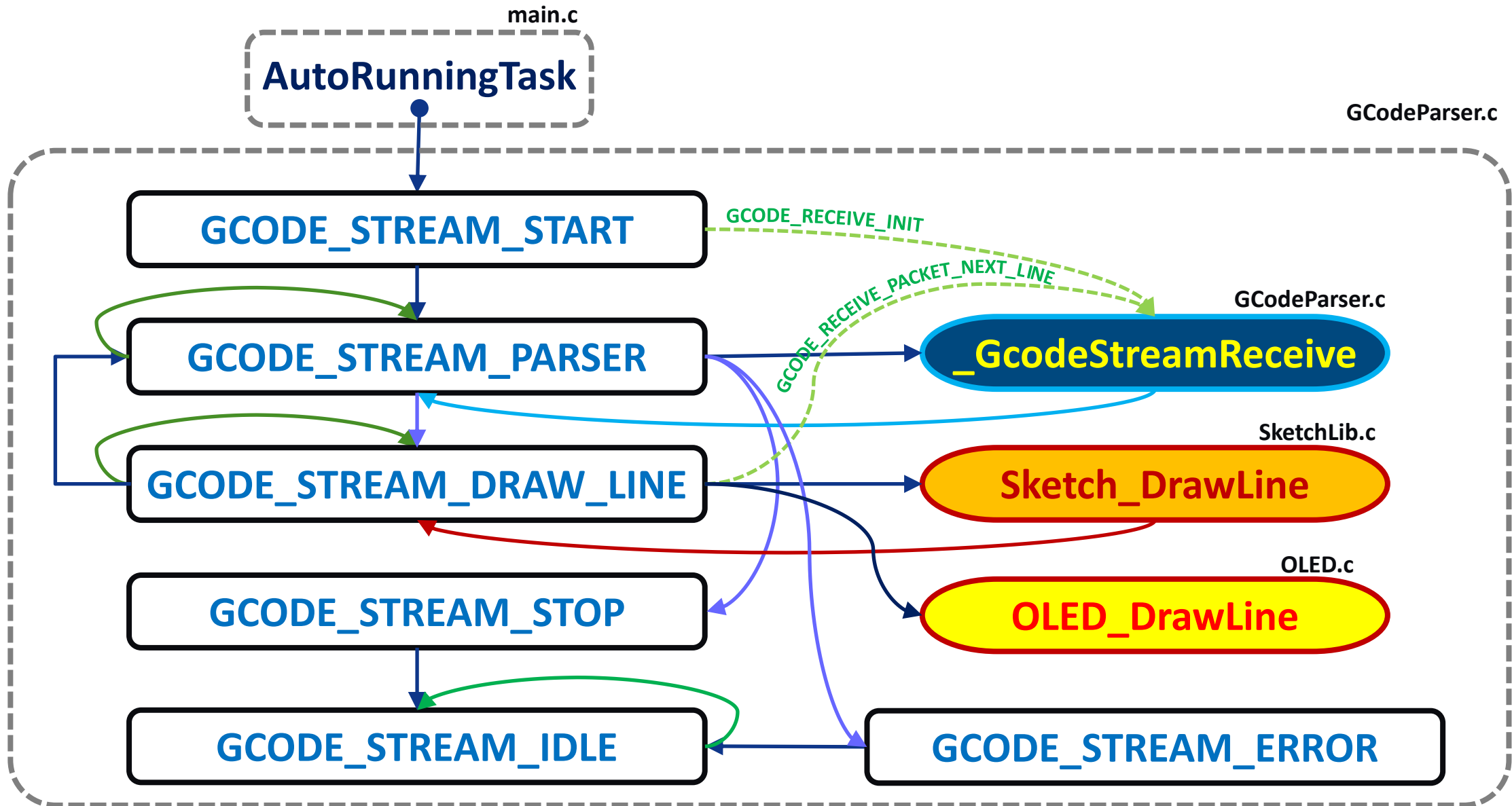


G-code Receive and Parser

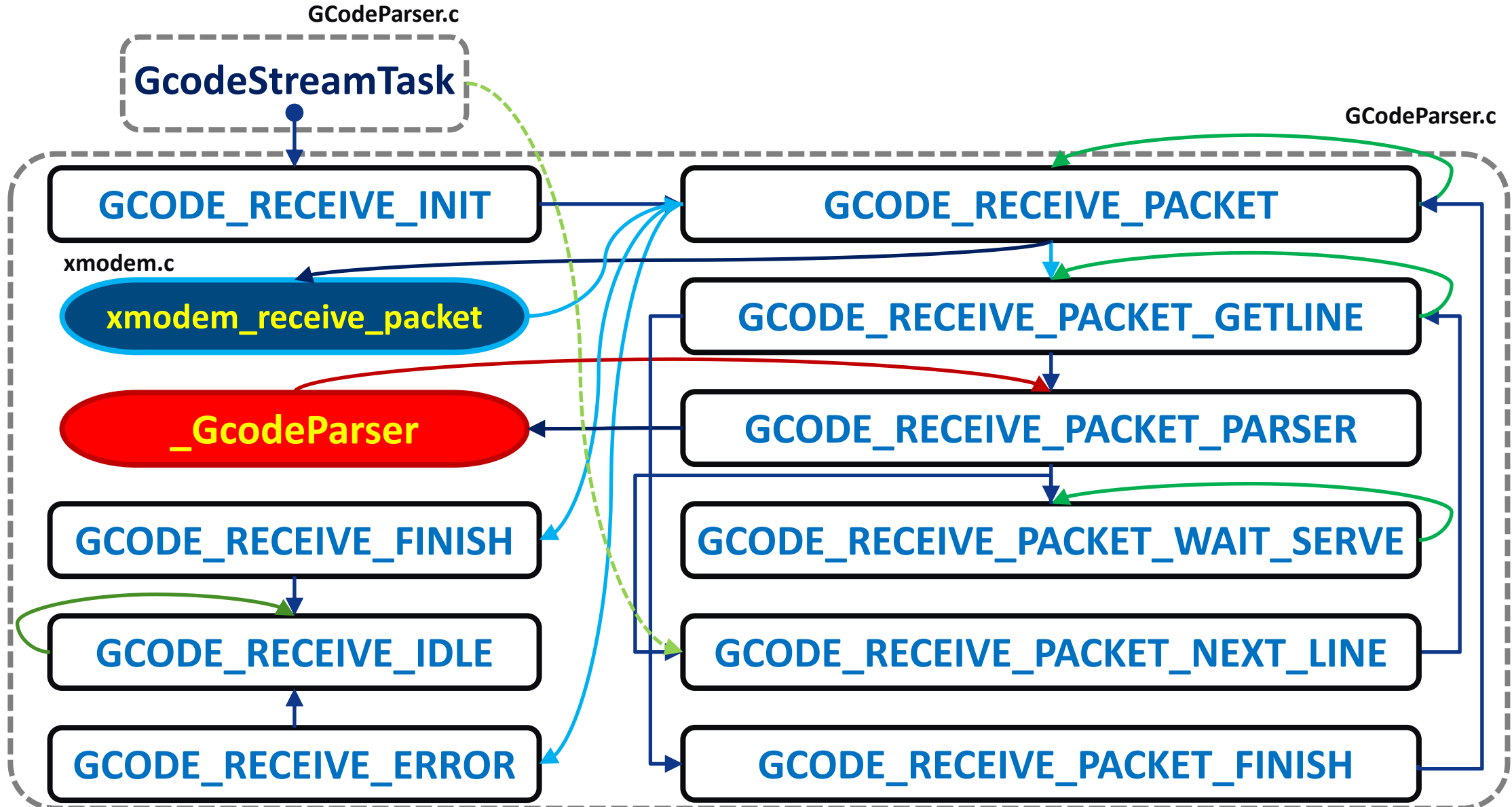
Main Flow of G-code Receive and Parser (main.c)



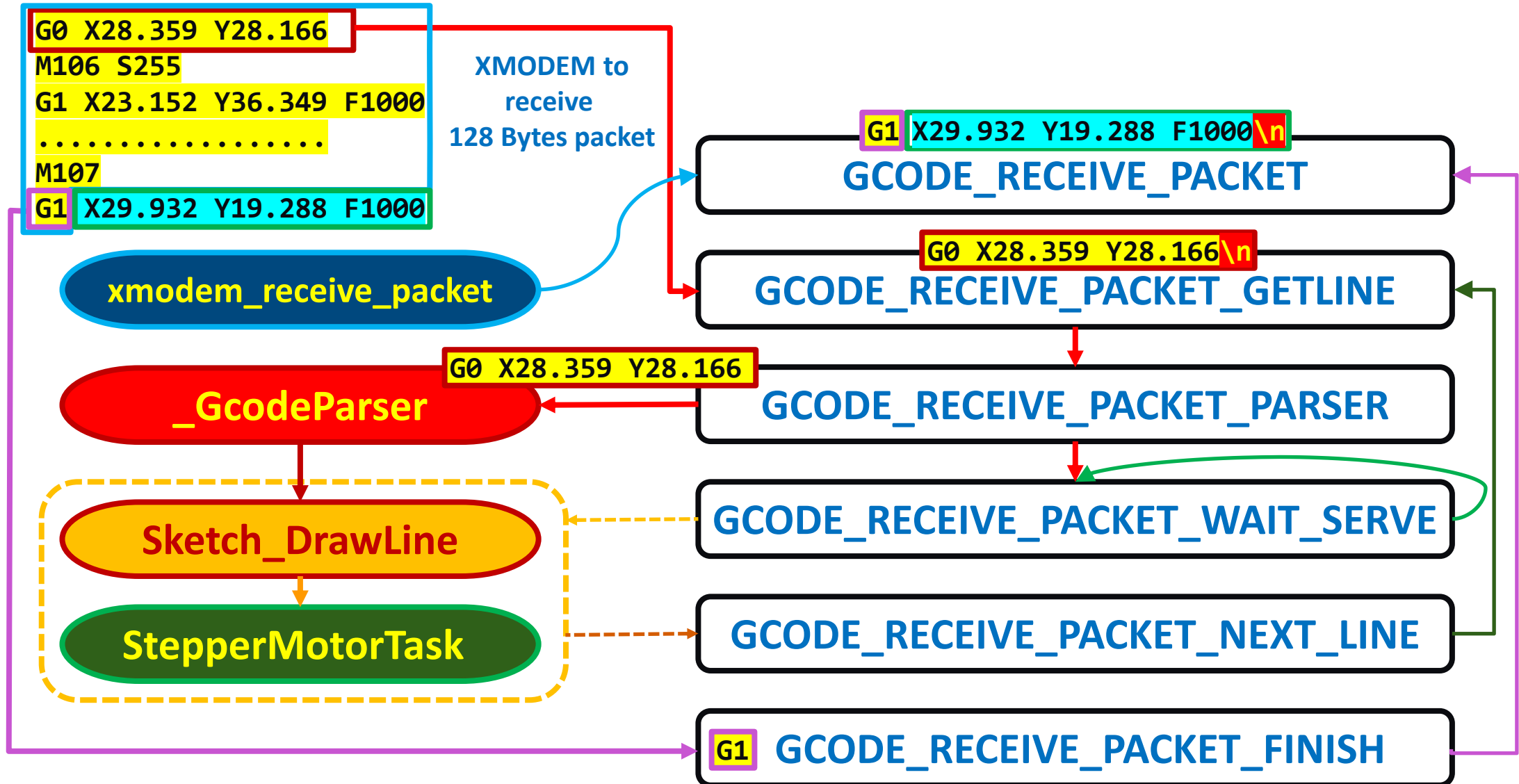
GcodeStreamTask state machine (GCodeParser.c / .h)



_GcodeStreamReceive state machine (GCodeParser.c / .h)



GcodeStreamReceive Get G-code Line (GCodeParser.c / .h)



_GcodeStreamReceive Get G-code Line (GCodeParser.c / .h)

```
G0 X28.359 Y28.166
M106 S255
G1 X23.152 Y36.349 F1000
G1 X28.426 Y28.076
; Put your Comment Here
M107
G4 X29.816 Y19.279 F30
G1 X29.932 Y19.288 F1000
G1 X29.523 Y19.158
G1 X29.614 Y19.213
G1 X29.712 Y19.254
G1 X29.816 Y19.279
M107
G0 X24.923 Y19.249
M106 S255
G1 X24.972 Y19.227 F1000
G1 X25.015 Y19.196
G1 X25.053 Y19.158
G1 X25.084 Y19.114
G1 X25.107 Y19.066
G1 X25.121 Y19.014
```

PacketBuffer[128]

```
G0 X28.359 Y28.166\n
M106 S255\n
G1 X23.152 Y36.349 F1000\n
G1 X28.426 Y28.076\n
; Put your Comment Here\n
M107\n
G4 X29.816 Y19.279 F30\n
G1
```

GCODE_RECEIVE_PACKET

GcodeLine[128]

```
G0 X28.359 Y28.166
M106 S255
G1 X23.152 Y36.349 F1000
G1 X28.426 Y28.076
; Put your Comment Here
M107
G4 X29.816 Y19.279 F30
G1
G1 X29.932 Y19.288 F1000
.....
```

PACKET_GETLINE

PACKET_PARSER

PACKET_PARSER

PACKET_PARSER

PACKET_PARSER

PACKET_NEXT_LINE

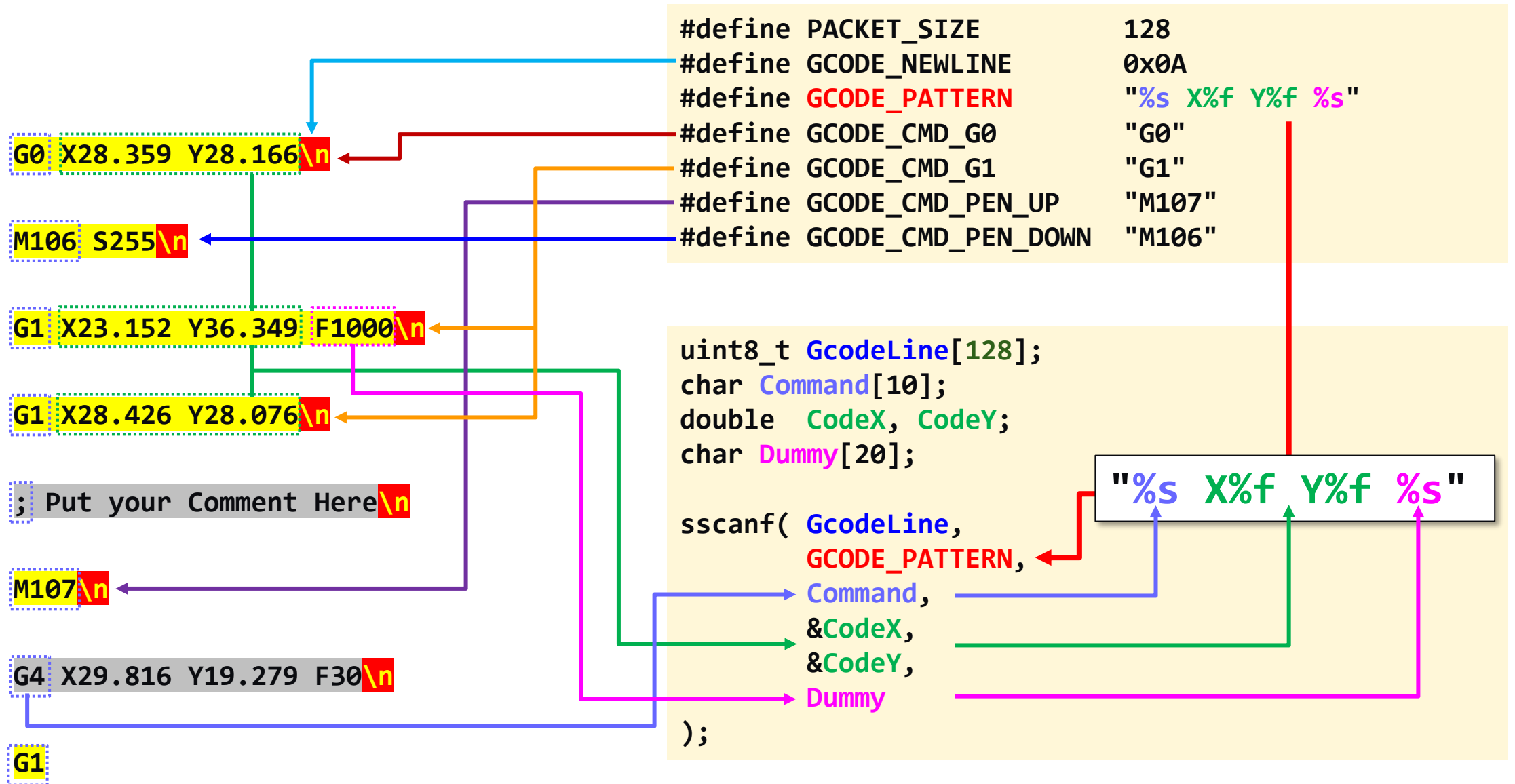
PACKET_PARSER

PACKET_NEXT_LINE

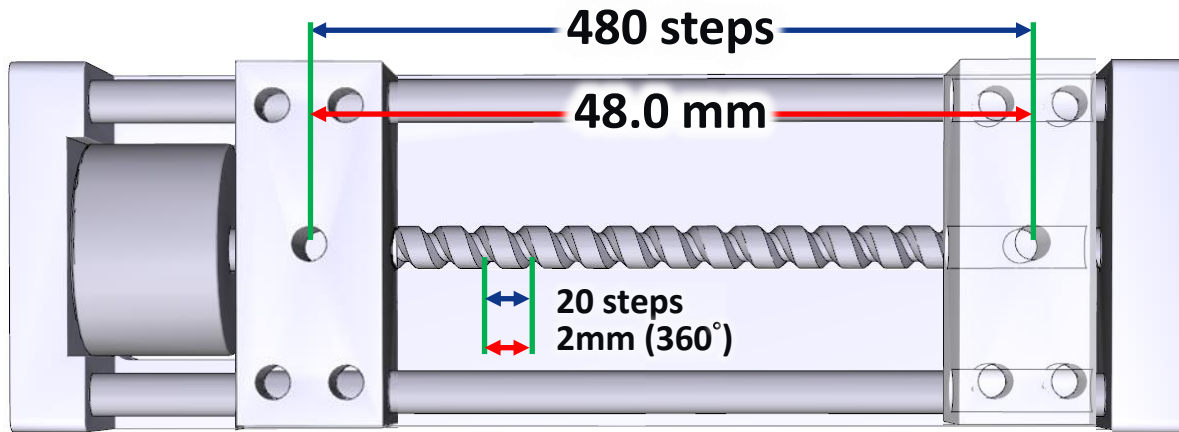
PACKET_FINISH

GCODE_RECEIVE_PACKET

_GcodeParser to Parser G-code Line (GCodeParser.c / .h)



GcodeMmToStep for mm to Step (GCodeParser.c / .h)



```
G1 X23.152 Y36.349\n
```

$X = 23.152, Y = 36.349$ (mm)

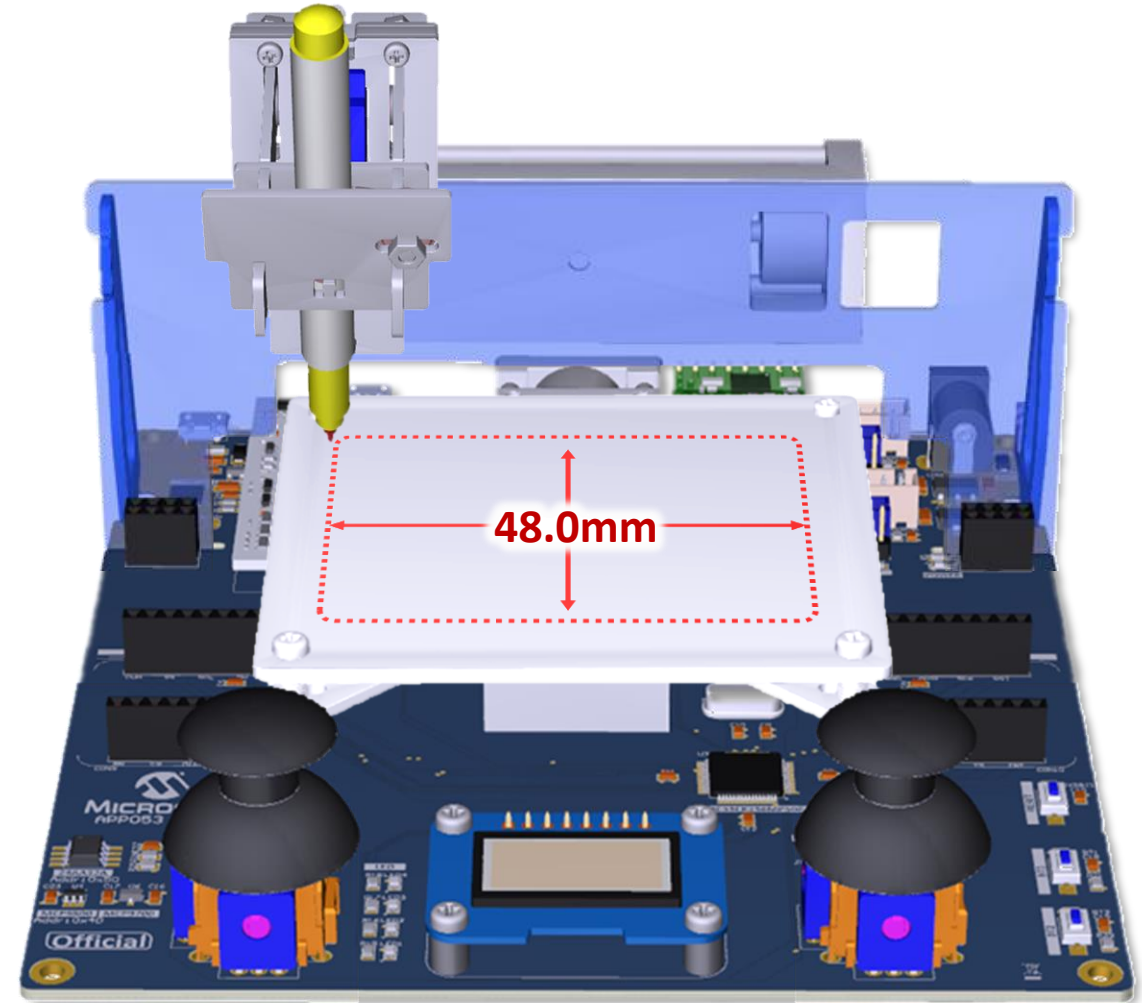
$\text{StepX} = X * 480 / 48.0 = 231$ (steps)

$\text{StepY} = Y * 480 / 48.0 = 363$ (steps)

or

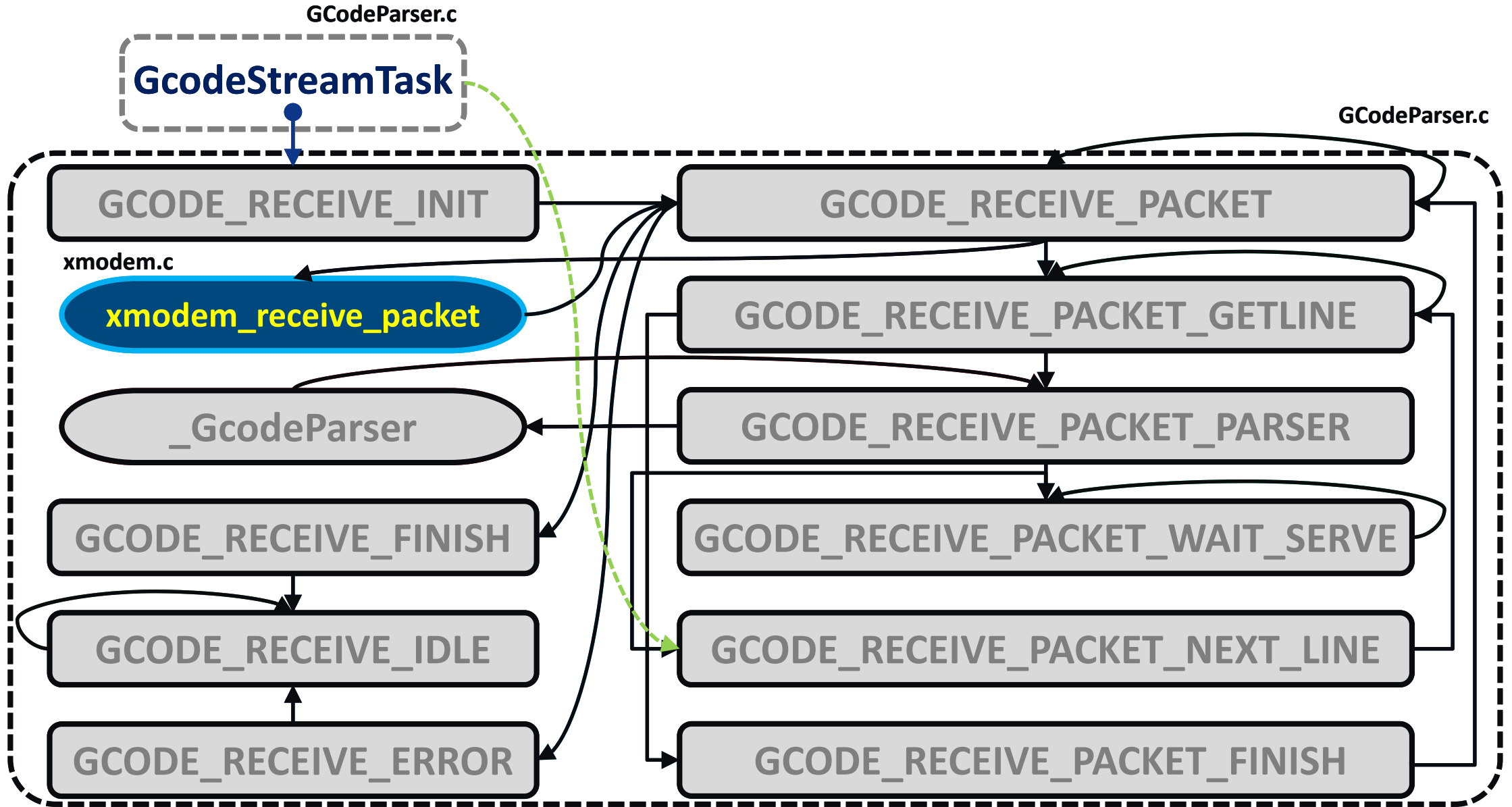
$\text{StepX} = X * 20 / 2 = 231$ (steps)

$\text{StepY} = Y * 20 / 2 = 363$ (steps)



XMODEM (G-code Receive)

xmodem_receive_packet implement (xmodem.c /.h)

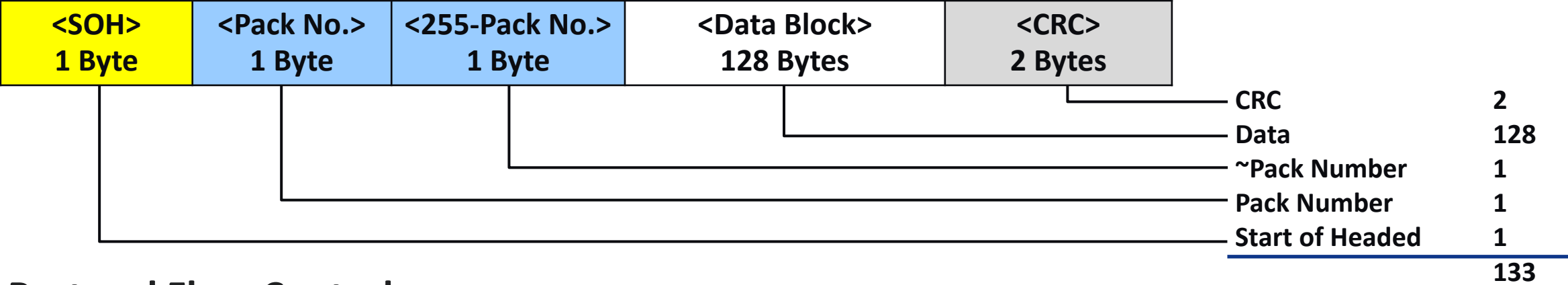


XMODEM

- **XMODEM** is a simple **file transfer** protocol developed as a quick **hack** by **Ward Christensen** for use in his 1977 **MODEM.ASM terminal program**. It allowed users to transmit files between their computers when both sides used MODEM.
- XMODEM, like most file transfer protocols, breaks up the original data into a series of "**packets**" that are sent to the receiver, along with additional information allowing the receiver to determine whether that packet was correctly received.
- If an error is detected, the receiver requests that the packet be re-sent.
- A string of bad packets causes the transfer to abort.

XMODEM

- Packet Format



133

- Protocol Flow Control

Symbol	Description	Value
SOH	Start of Header	0x01
EOT	End of Transmission	0x04
ACK	Acknowledge	0x06
NAK	Not Acknowledge	0x15
ETB	End of Transmission Block	0x17
CAN	Cancel (Force receiver to start sending C's)	0x18
C	ASCII 'C'	0x43

XMODEM

- **Synchronization**

The receiver starts by sending an **ASCII "C" (0x43)** character to the sender indicating it wishes to use the **CRC method of block validating**. After sending the initial "C" the receiver waits for either a **3 second time out** or until a buffer full flag is set. If the receiver is timed out then another "C" is sent to the sender and the 3 second time out starts again. This process continues until the receiver receives a complete **133-byte packet**.

- **Receiver Considerations**

This protocol NAKs the following conditions:

- | | |
|------------------------------|--|
| 1. Framing error on any byte | (Serious hardware failures. Verify baud rate, start bits and stop bits.) |
| 2. Overrun error on any byte | (Serious hardware failures. Verify baud rate, start bits and stop bits.) |
| 3. Duplicate packet | (Sender getting an ACK garbled and re-transmitting the packet) |
| 4. CRC error | (In noisy environments) |
| 5. Receiver timed out | (Didn't receive packet within 1 second) |

On any NAK, the sender will re-transmit the last packet.

XMODEM

Sender						Receiver
					<---	"C"
						Times Out after 3 Seconds
					<---	"C"
SOH	0x01	0xFE	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x02	0xFD	Data	CRC	--->	(Line Hit during Data Transmission)
					<---	NACK
SOH	0x02	0xFD	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x03	0xFC	Data	CRC	--->	Packet OK
(ACK Gets Garbled)					<---	ACK
					<---	ACK
SOH	0x04	0xFB	Data	CRC	--->	(UART Framing Error on Any Byte)
					<---	NACK
SOH	0x04	0xFB	Data	CRC	--->	Packet OK
					<---	ACK
SOH	0x05	0xFA	Data	CRC	--->	(UART Overrun Error on Any Byte)
					<---	NACK
SOH	0x05	0xFA	Data	CRC	--->	Packet OK
					<---	ACK
EOT					--->	Packet OK
					<---	ACK
ETB					--->	Finished
Finished					<---	ACK

SketchLib (2D Graphic Drawing)

2D Sketch Library (SketchLib.c /.h)

- About the Sketch Library.

For Stepper Motor control, we build an easy Sketch Library for 2D Drawing purpose.

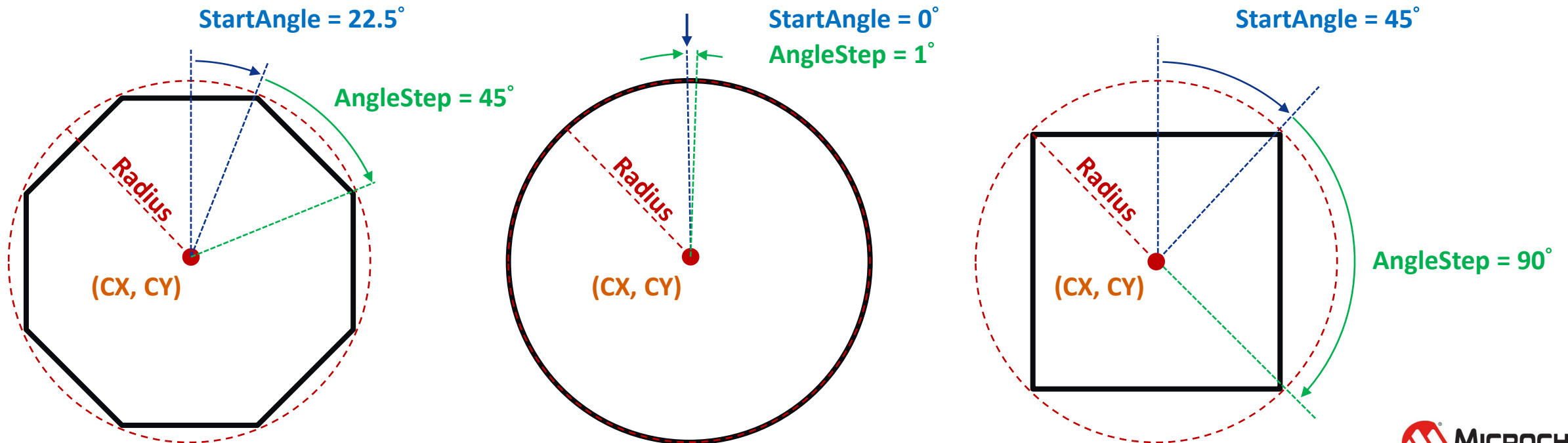
bool **Sketch_DrawLine**(const int32_t **X1**, const int32_t **Y1**, const int32_t **X2**, const int32_t **Y2**)

(**X1**, **Y1**) ● ————— ● (**X2**, **Y2**)

bool **Sketch_DrawPolygon**(const int32_t **CX**, const int32_t **CY**, const int32_t **Radius**, const int32_t **StartAngle**, const int32_t **AngleStep**)

bool **Sketch_DrawCircle**(const int32_t **CX**, const int32_t **CY**, const int32_t **Radius**)

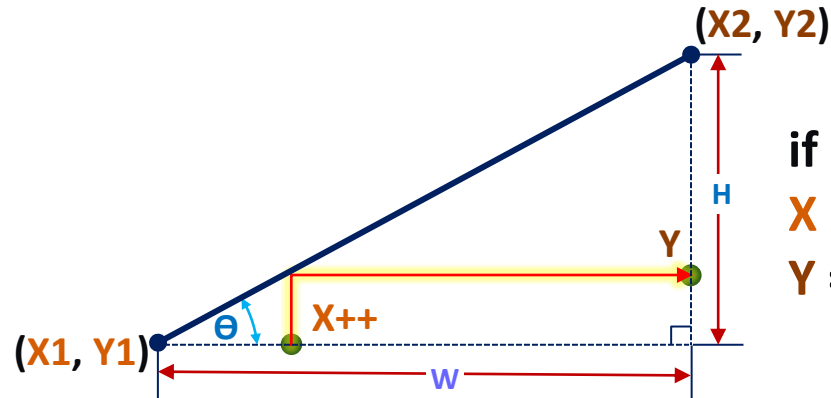
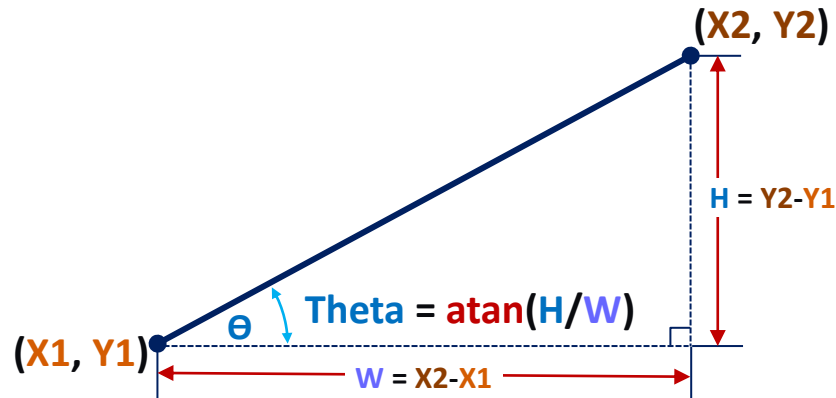
bool **Sketch_DrawRectangle**(const int32_t **CX**, const int32_t **CY**, const int32_t **Radius**)



2D Sketch Library (SketchLib.c /.h)

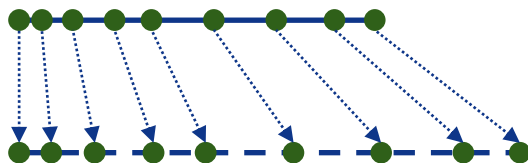
Sketch_DrawLine(const int32_t **X1**, const int32_t **Y1**, const int32_t **X2**, const int32_t **Y2**)

Given two point (**X1**, **Y1**) and (**X2**, **Y2**) to calculate angle **Theta**.

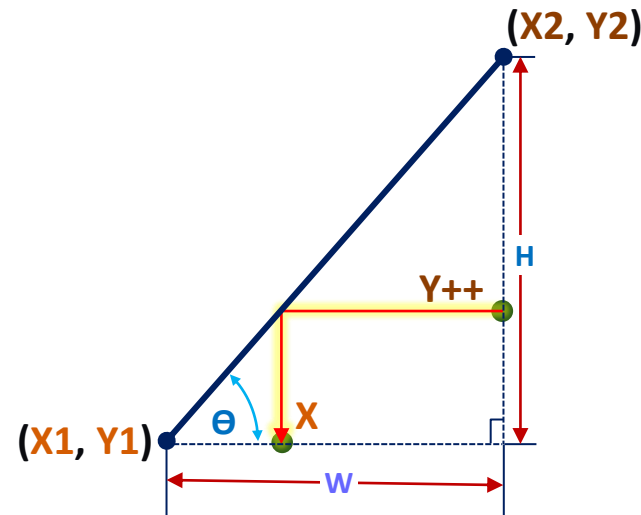
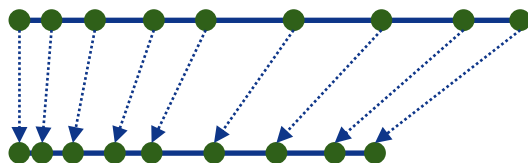


if $W \geq H$, Move **X** to calculate **Y**
 $X = X1 ++$
 $Y = Y1 + (\tan(\Theta) * X - X1)$

Projection
with hole



Projection
eliminate hole

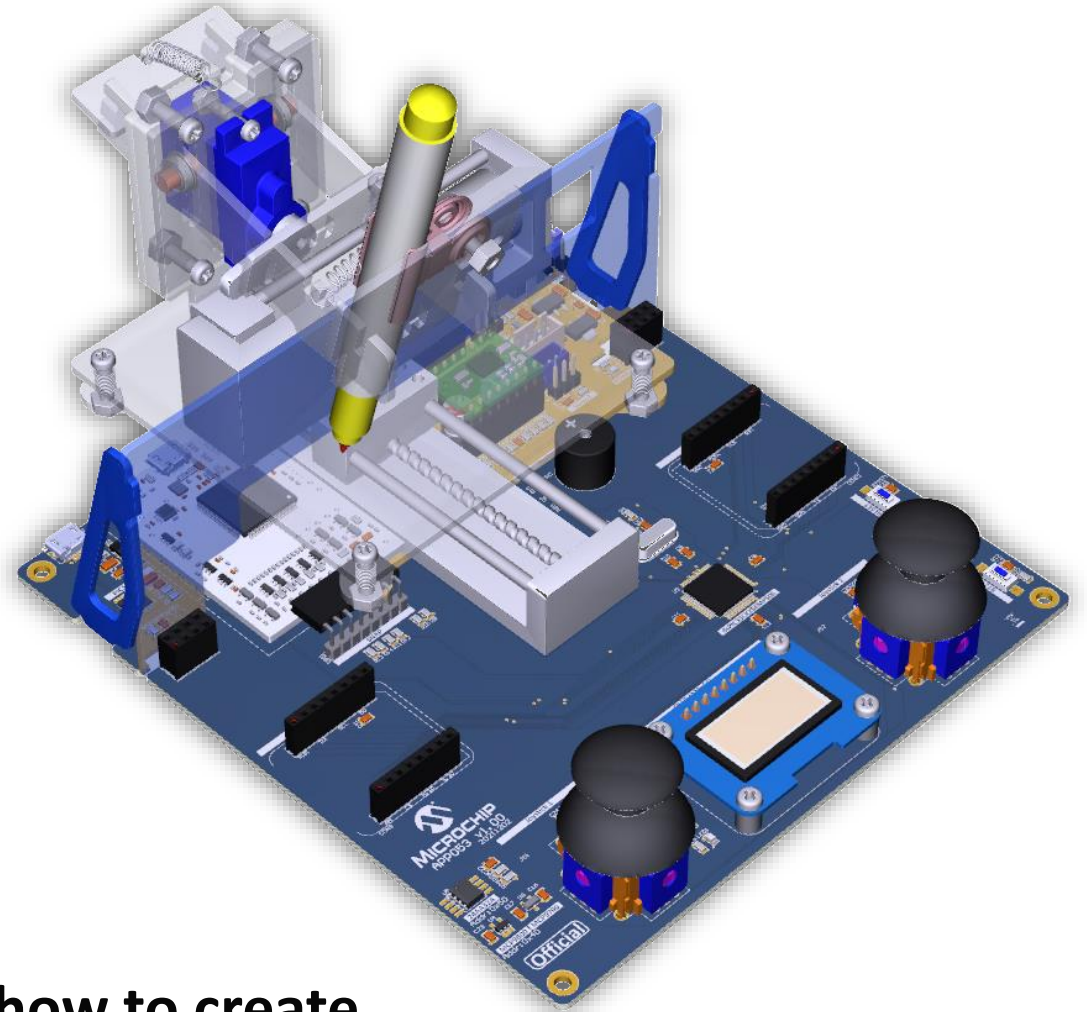


if $W < H$, Move **Y** to calculate **X**
 $Y = Y1 ++$
 $X = X1 + (Y - Y1 / \tan(\Theta))$

G-code, 3D Model and Slice Introduction

Introduction

- This is an application section.
At previous slice, you already now that how to build a “**2 Axis Tiny Working Station**” and finished the firmware for “**2D Sketch Drawing Machine**”.
You can control it very well also, I guessed. :P
- So, at this section, you will know that the knowledge about,
 - information of G-code
 - G-code Parser for APP053
 - Slice Tool
 - 3D Modeling Tool
- We have three exercise at here and teach you how to create your patterns to draw it by our “**2D Sketch Drawing Machine**”.



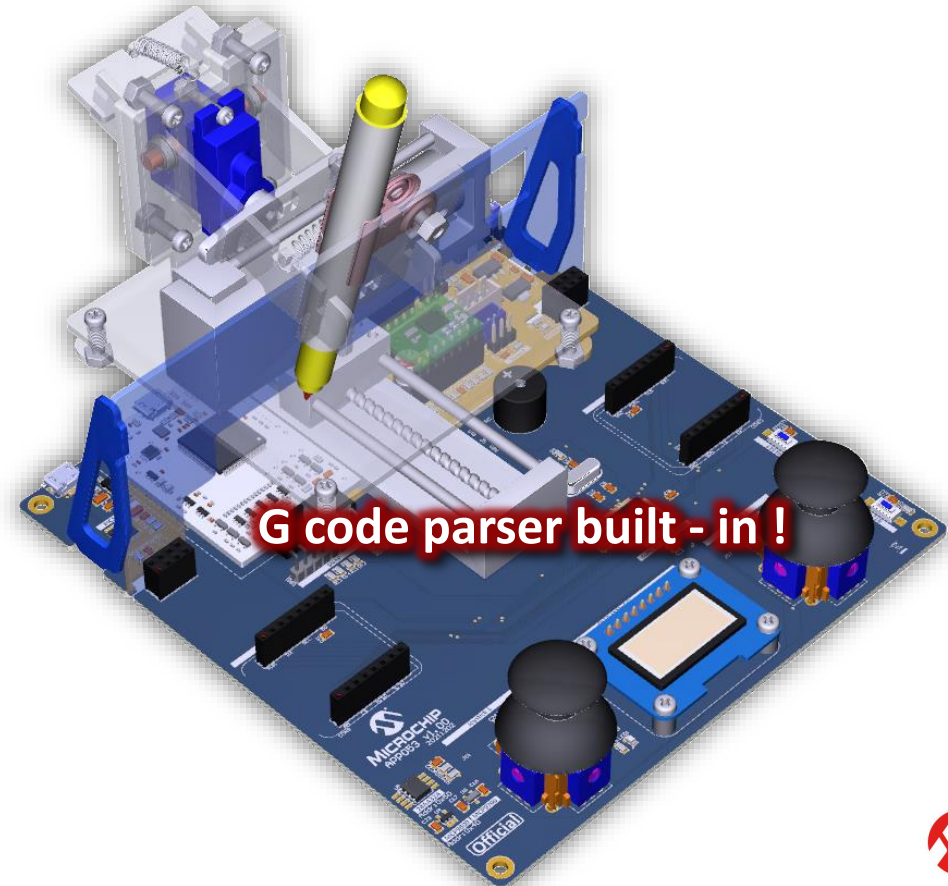
What's G-code ?

G-code Introduction

- G-code or named G-language means “Geometric code” is the operational language for CNC machining.
- It use for numerically controlled lathes(車床) and machining centers(加工中心) how to move tools in order to perform various cutting(切削) operations.

www.thomasnet.com/articles/custom-manufacturing-fabricating/introduction-gcode

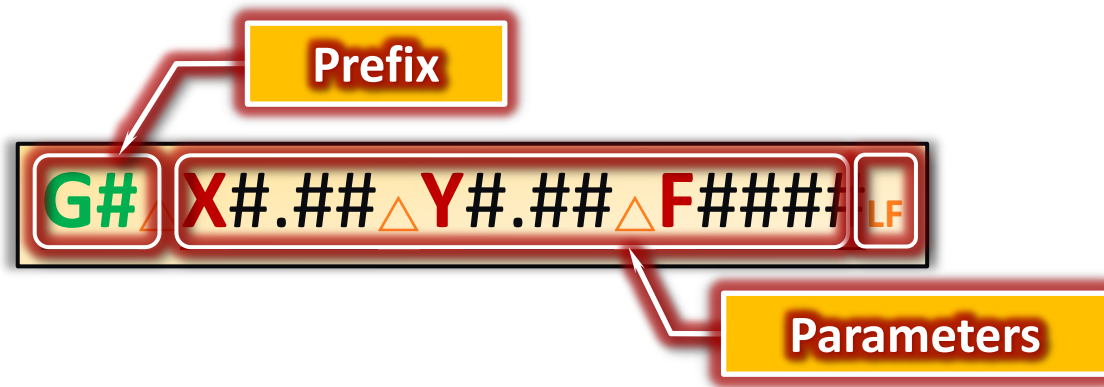
- We apply a G-code parser named “**LADK parser**” to our platform named “**2D Sketch Drawing Machine**” already. So, we can send the G-code to control our platform follow needs, right now.



Format of G-code

- G-code is known as RS-274. G-code is text format files, the command line look like below.

www.thomasnet.com/articles/custom-manufacturing-fabricating/introduction-gcode



- A Prefix figure out G#, # is number means command number and follow by parameter depend on command type that end by line Feed (LF).

G-code Types

- **G-code is not a strongly defined format.**
- **there are compatibility issues with different versions or types of machines.**
- Different formats, commands and parameters are needed for different machines.
- Various commands,
 - G codes(commands) for preparatory command.
 - M codes(commands) for miscellaneous function.
 - For CNC machines that provide like F code, S code and T code.
 - For 3D Printers(FDM) that provide like F code, E code.

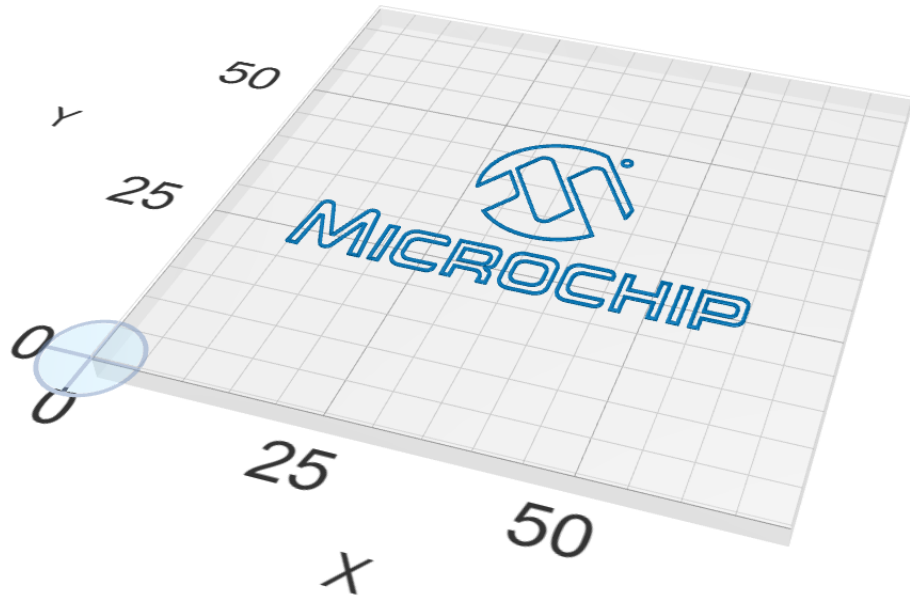
G-code Example

- Please refer the G-code example. This is part of G-code that be generated by standard 3D Printer slicer tool named “**Ultimaker Cura**”. Let me look at this.
- There are specific features (but not limited) on the example,
 - It's text format files and use “**CR, (0x0D), \r**” & “**LF, (0x0A), \n**” to Separate each command.
 - Shows two different prefix command type G & M.
G1 means line move.
G28 means home.
M105 and **M109** is special features for 3D printer.
 - etc. .
- For detail information please refer On-line G-code document, <https://marlinfw.org/meta/gcode>

```
...  
M109 S180 \r\n; Wait for Hotend  
M105 \r\n; Report Temperatures  
G28 \r\n; Home  
G1 Z15.0 F6000 \r\n; Linear Move  
G92 E0 \r\n; Set Position  
G1 F200 E3 \r\n; Linear Move  
G92 E0 \r\n; Set Position  
G1 F600 Z0.25 \r\n; Linear Move  
G1 F3000 E0 \r\n; Linear Move  
G1 F1800 X3.958 Y-64.82 E0.09356 \r\n; Linear Move  
G1 X6.626 Y-64.971 E0.18689 \r\n; Linear Move  
G1 X9.282 Y-65.197 E0.27998 \r\n; Linear Move  
G1 X31.295 Y-62.144 E1.06191 \r\n; Linear Move  
G1 X33.965 Y-61.86 E1.15568 \r\n; Linear Move  
G1 X34.36 Y-61.827 E1.16952 \r\n; Linear Move  
...
```

Format of G-code for Here

- We aligned that a specific g-code format that be generated by On-line slice tool named “**Kiri:Moto**” based on pre-defined machine and slice setting.
- Please refer the G-code. It’s a very different with G-code of previous slide.

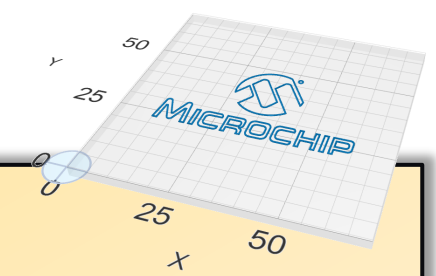


Kiri:Moto

```
...  
G0 X28.359 Y28.166 \n; Rapid Move  
M106 S255 \n; Laser Trun On  
G1 X23.152 Y36.349 F1000 \n; Linear Move with Move Rate  
G1 X24.924 Y38.849 \n; Linear Move  
G1 X24.995 Y38.935 \n; Linear Move  
G1 X28.359 Y28.166 \n; Linear Move  
M107 \n; Laser Trun Off  
G0 X29.816 Y19.279 \n; Rapid Move  
M106 S255 \n; Laser Trun On  
G1 X29.932 Y19.288 F1000 \n; Linear Move with Move Rate  
G1 X31.416 Y19.288 \n; Linear Move  
G1 X31.533 Y19.279 \n; Linear Move  
M107 \n; Laser Trun Off  
...
```

Format of G-code for Here *cont.1*

- There are specific features (but not limited) on the example,
 - It's text format files and use “**LF, (0x0A),\n**” to Separate each command, only.
 - Shows two different prefix command type G & M.
 - **G0** means rapid move.
 - **G1** means line move.
 - **M106** and **M107** is special features for Laser cutting means is turn On and Off Leaser header.
 - etc. .



```
...  
G0 X28.359 Y28.166 \n; Rapid Move  
M106 S255 \n; Laser Trun On  
G1 X23.152 Y36.349 F1000 \n; Linear Move with Move Rate  
G1 X24.924 Y38.849 \n; Linear Move  
G1 X24.995 Y38.935 \n; Linear Move  
G1 X28.359 Y28.166 \n; Linear Move  
M107 \n; Laser Trun Off  
G0 X29.816 Y19.279 \n; Rapid Move  
M106 S255 \n; Laser Trun On  
G1 X29.932 Y19.288 F1000 \n; Linear Move with Move Rate  
G1 X31.416 Y19.288 \n; Linear Move  
G1 X31.533 Y19.279 \n; Linear Move  
M107 \n; Laser Trun Off  
...
```

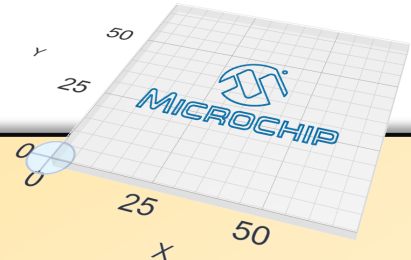
Customized G-code Parser

LADK Parser Introduction

Customized G-code Parser for Laser Cutter Mode

LADK Parser : Customized G-code Parser

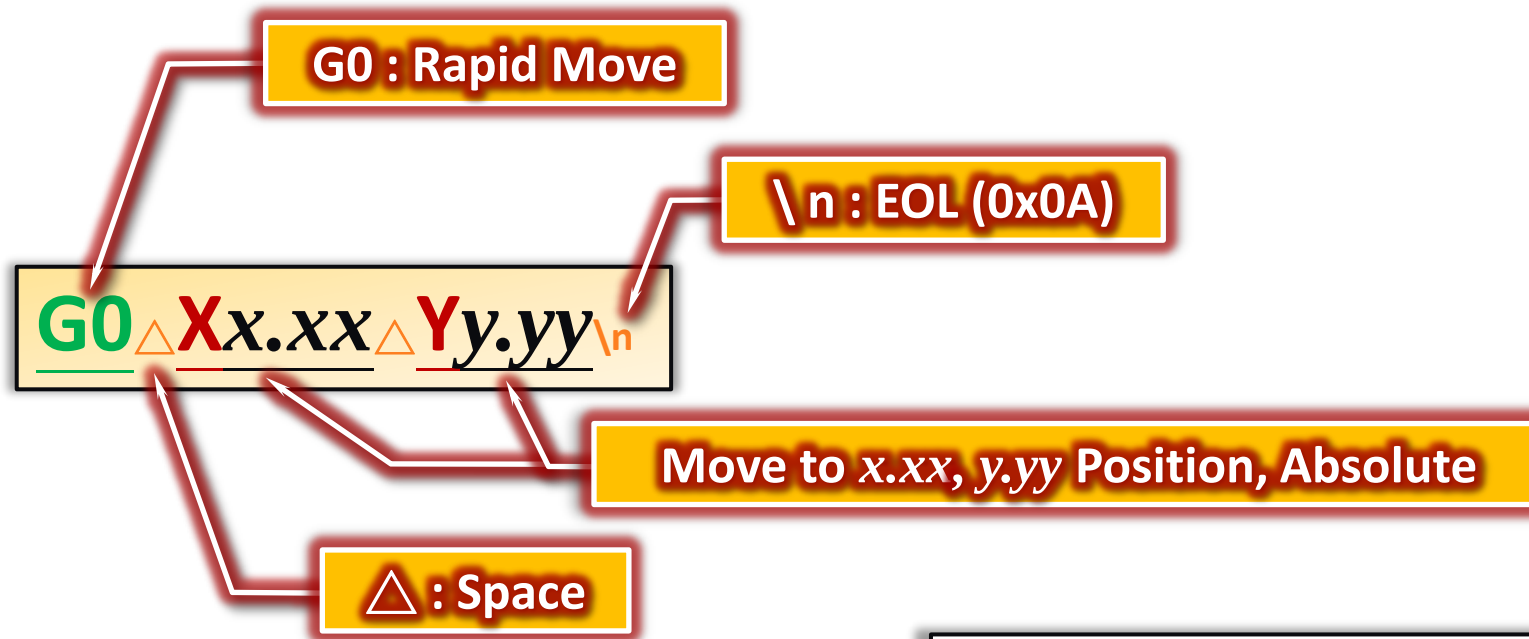
- For this training. We create a customized G-code parser named “LADK Parser” that created by Libra Chien.
- LADK parser are compatible with that the G-code format generated by Kiri:Moto Online Slice Tool based at pre-defined machine and slice setting. We will talk about Kiri:Moto later. Let me look more detail for the G-code.
- There are 5 different patterns of G-code we want/can parse it and ignore others,
 - M106 S### \n; Laser Trun On
 - M107 \n; Laser Trun Off
 - G1 X#.### Y#.### F#### \n; Linear Move
 - G1 X#.### Y#.### \n; Linear Move
 - G0 X#.### Y#.### \n; Rapid Move
- Each G-code command consist by prefix, space, commands, parameters and EOL (End of Line) symbol.



```
...
G0 X28.359 Y28.166 \n; Rapid Move
M106 S255 \n; Laser Trun On
G1 X23.152 Y36.349 F1000 \n; Linear Move with Move Rate
G1 X24.924 Y38.849 \n; Linear Move
G1 X24.995 Y38.935 \n; Linear Move
G1 X28.359 Y28.166 \n; Linear Move
M107 \n; Laser Trun Off
G0 X29.816 Y19.279 \n; Rapid Move
M106 S255 \n; Laser Trun On
G1 X29.932 Y19.288 F1000 \n; Linear Move with Move Rate
G1 X31.416 Y19.288 \n; Linear Move
G1 X31.533 Y19.279 \n; Linear Move
M107 \n; Laser Trun Off
...
```

G0 Command decoder

- G0 Command is rapid move command.



e.g.:

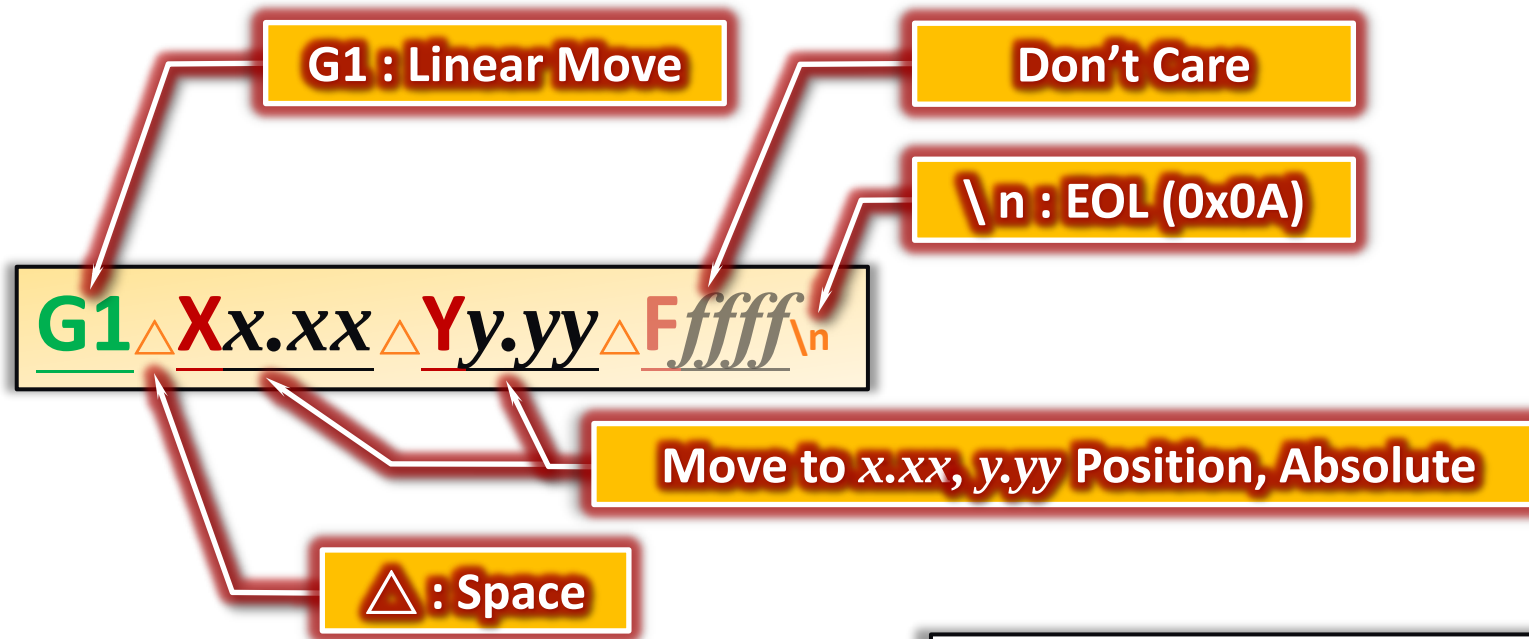
`G0 X28.359 Y28.166 \n`; Rapid Move

`G0 X29.816 Y19.279 \n`; Rapid Move

...

G1 Command decoder

- G1 Command is linear move command.

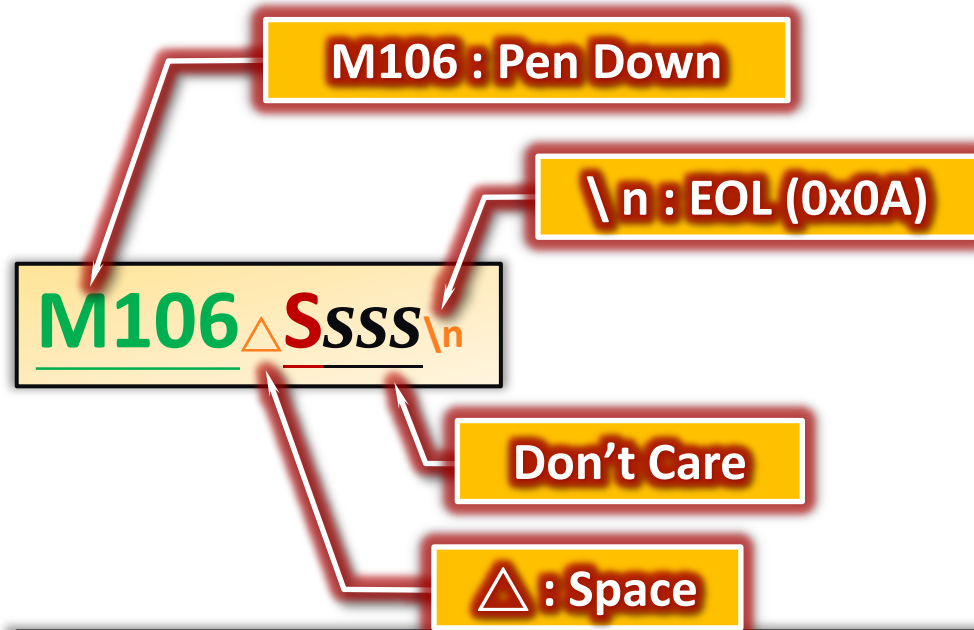


e.g.:

- `G1 X23.152 Y36.349 F1000 \n`; Linear Move with Move Rate
- `G1 X24.924 Y38.849 \n`; Linear Move
- `G1 X24.995 Y38.935 \n`; Linear Move
- `G1 X28.359 Y28.166 \n`; Linear Move
- ...

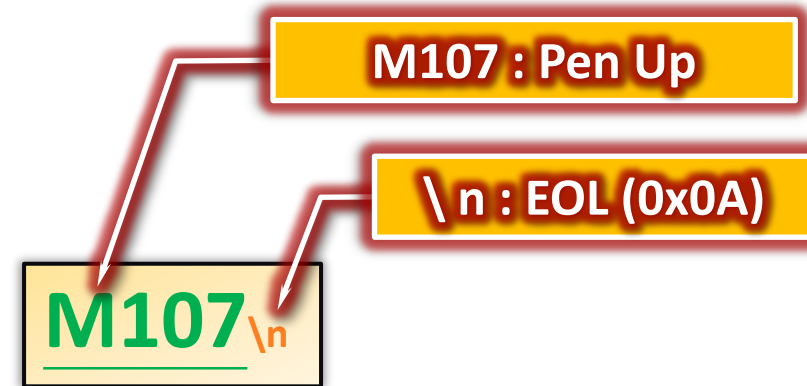
M106 & M107 Command decoder

- M106 & M107 Command are Laser ON/OFF (Pen Down / Up) commands.



e.g.:
M106 Δ **S255** $\backslash n$; Laser Trun On

...

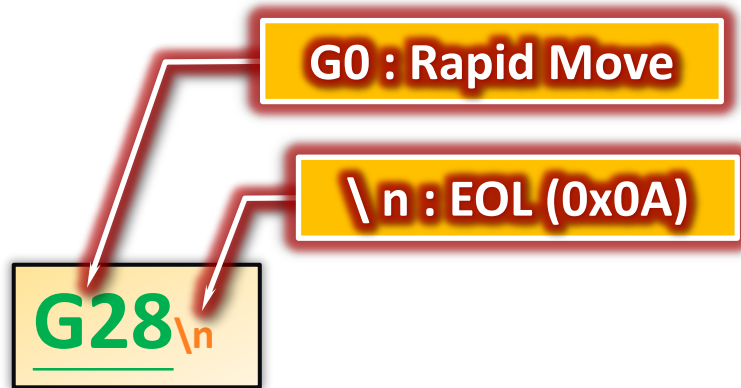


e.g.:
M107 $\backslash n$; Laser Trun Off

...

G28 Command decoder

- G28 is “Home” Command, that require move to (0,0).



e.g.:
G28 \n; move to home (0,0)
...

What's exercises at this section ?

Exercises Introduction

- We will finish below application exercises at this section,
 - Exercise A : Generate a G-code file at slice tool directly **by pre-saved workspace**.



- Exercise B : Generate a G-code file at slice tool directly by **import 3D Object (.stl)**.



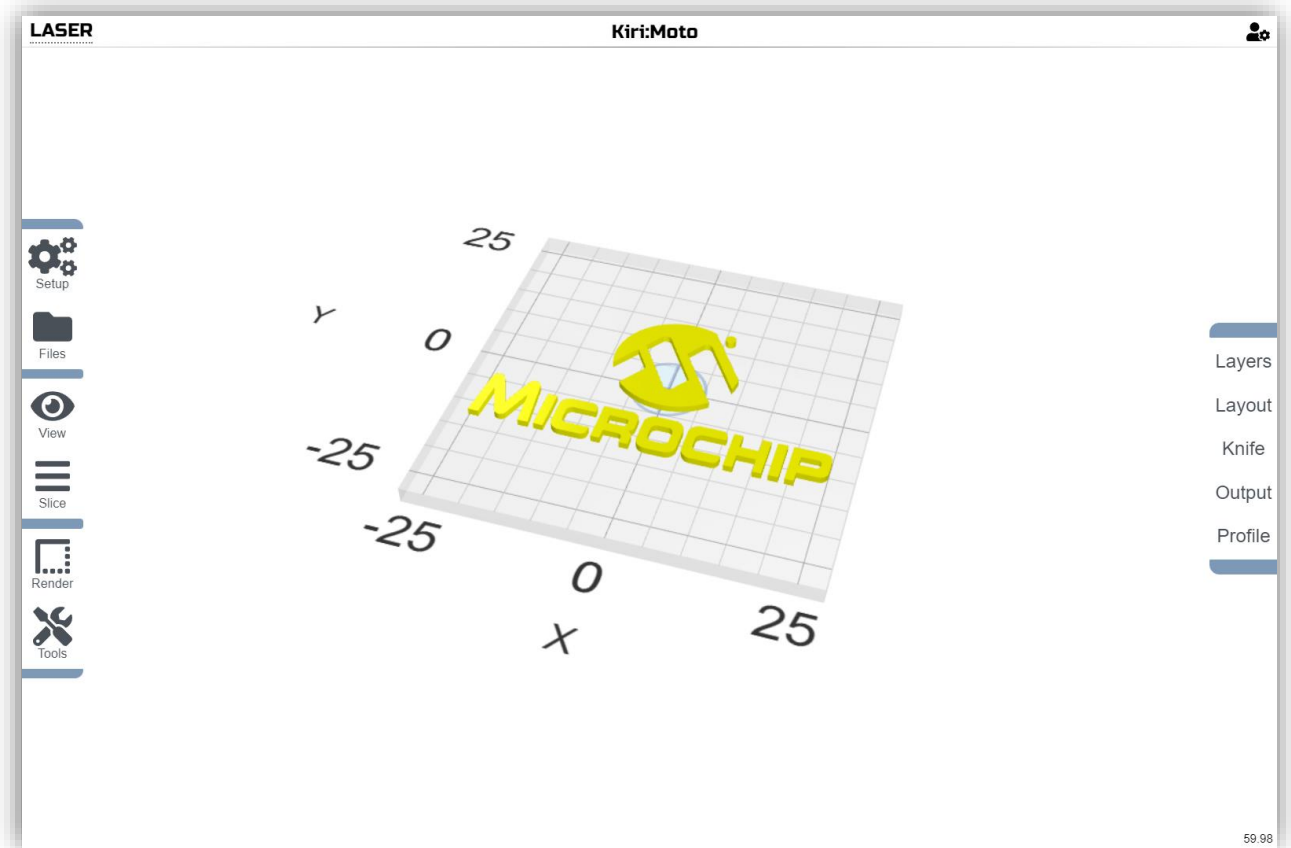
- Exercise C : **Create a 3D object and Generate the G-code file at slice tool.**



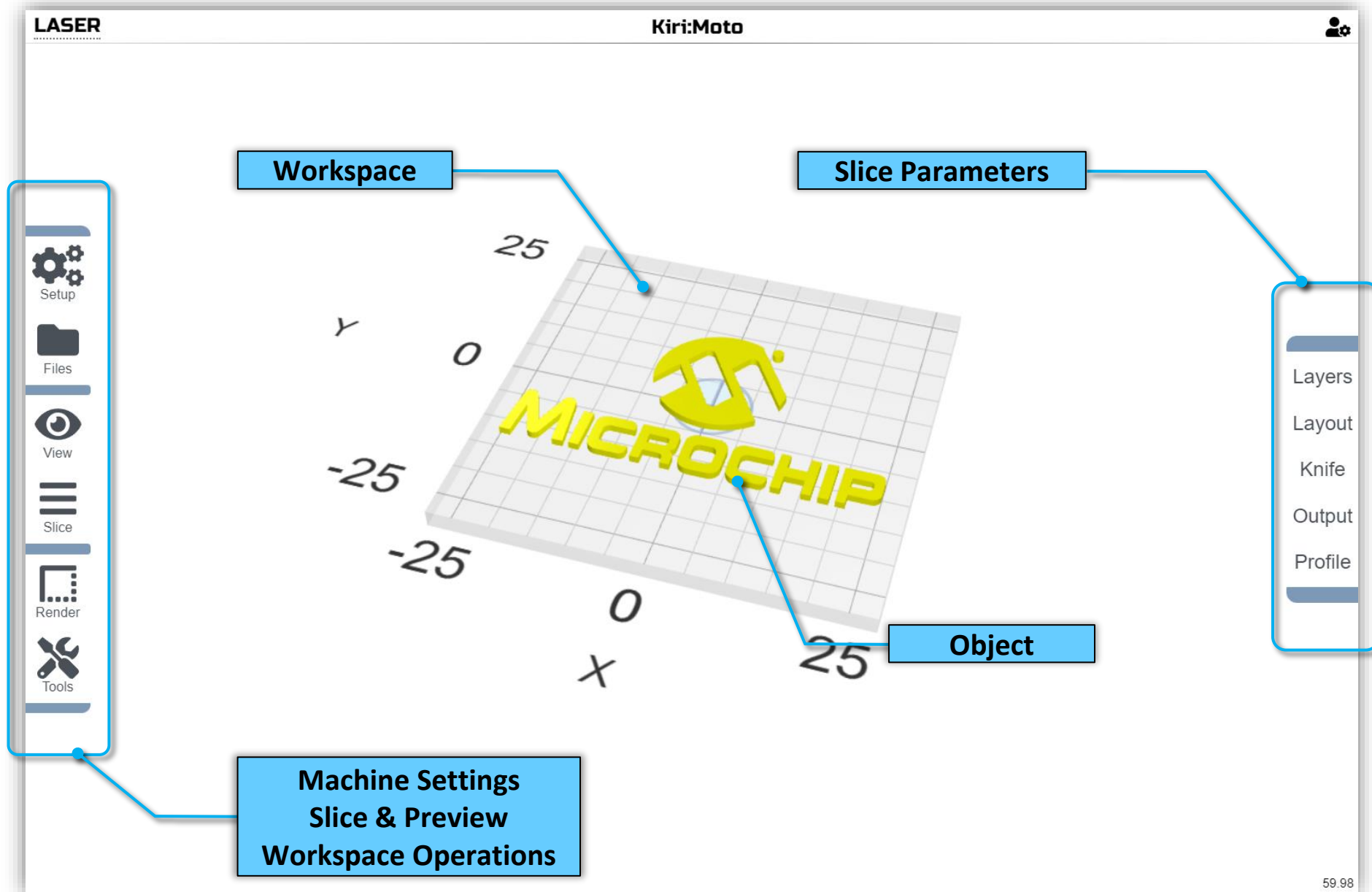
Online Slice Tool Kiri:Moto Introduction

Kiri:Moto - Online Slice Tool

- Kiri:Moto (grid.space/kiri) is a Browser-Based Slicer for 3D FDM and SLA printing, CNC Milling, and Laser Cutter.
- It's included below features,
 - It's Browser-Based.
 - Support FDM/SLA/CNC/LASER.
 - Support .STL (3D) and .SVG (2D).
 - Support profile & setting import / export.
 - Machine setting & Slice setting.
 - Workspace.
- You can find a lot on-line training and courses. e.g.,
https://www.youtube.com/watch?v=bZmCg98OD_8
https://www.youtube.com/watch?v=luJ_-3oEXx4
https://www.youtube.com/watch?v=s9OD_1hacwU
<https://www.youtube.com/watch?v=08795Sj22QE&list=PLRoVgyRoWZpumQIKWm7oldx3bjDivqP24>
etc..



Kiri:Moto Introduction



Kiri:Moto Introduction *cont.1*

LAZER

Kiri:Moto

Setup

Machine

Prefs

Files

View

Slice

Render

Tools

Machine Settings

Select A Device Type

FDM

SLA

CNC

LAZER

Standard Devices

Any Generic Laser

Glowforge

Snapmaker Original

My Devices

Microchip APP053+ V1.00

Copy Delete Export

Selected Device

Name

Microchip APP053+ v1.00

Volume X

50

Y

50

Z

Origin Center

☐

Output

Token Spacer

☒

File Extension

gcode

Gcode Macros

Header

Footer

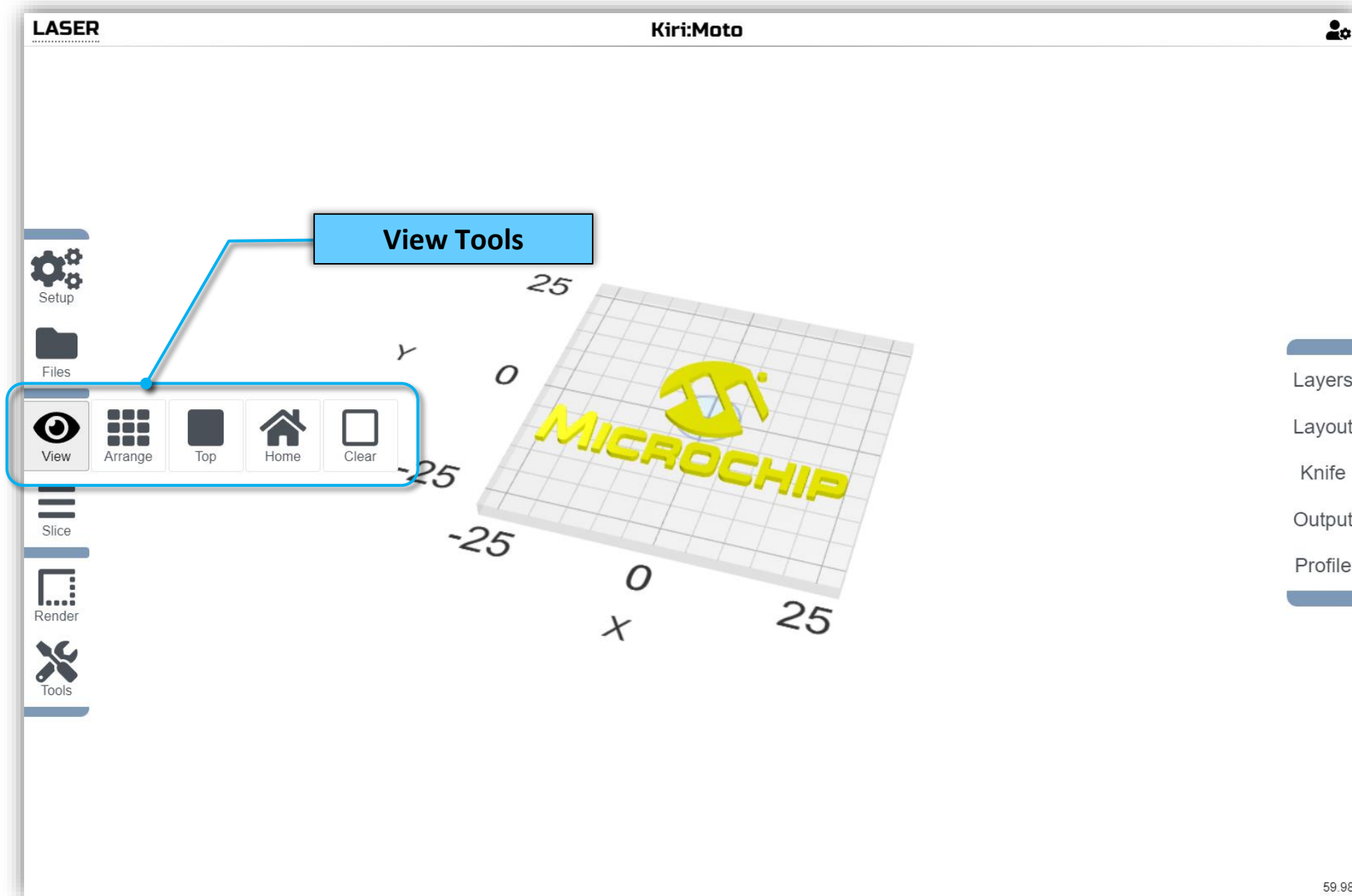
Laser On

Laser Off

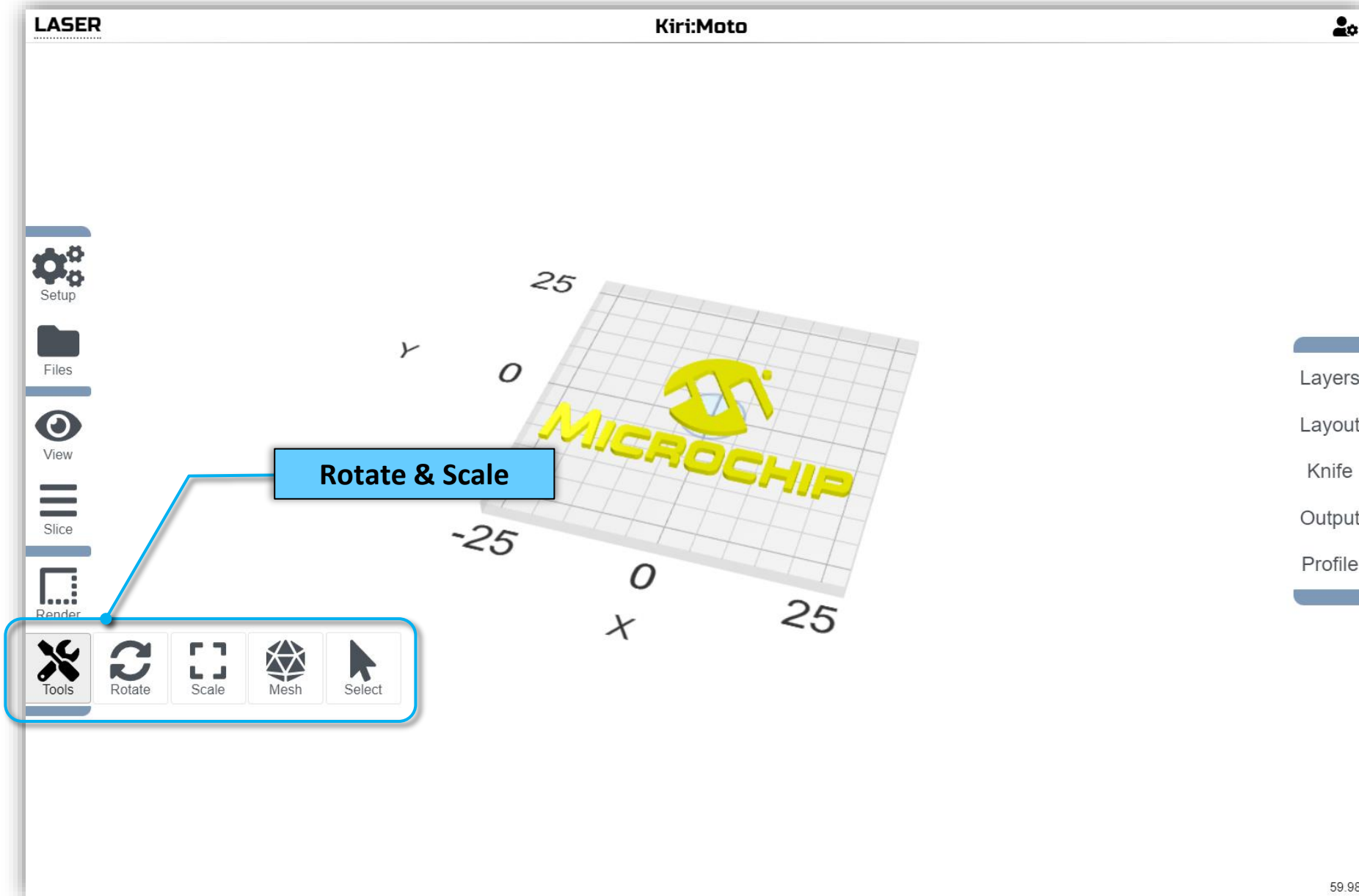
Save

Machine Settings
for
APP053+ v1.00

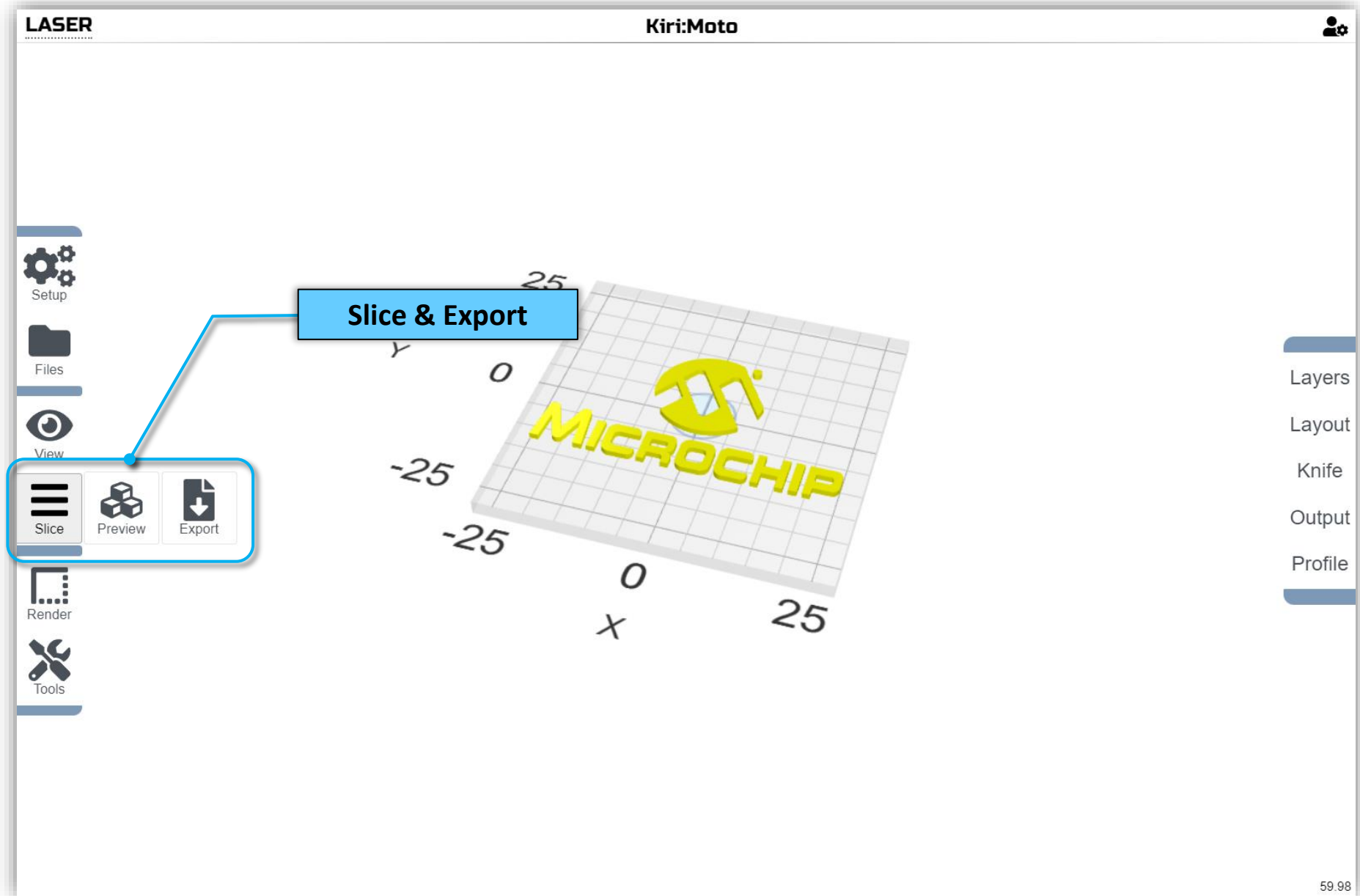
Kiri:Moto Introduction *cont.2*



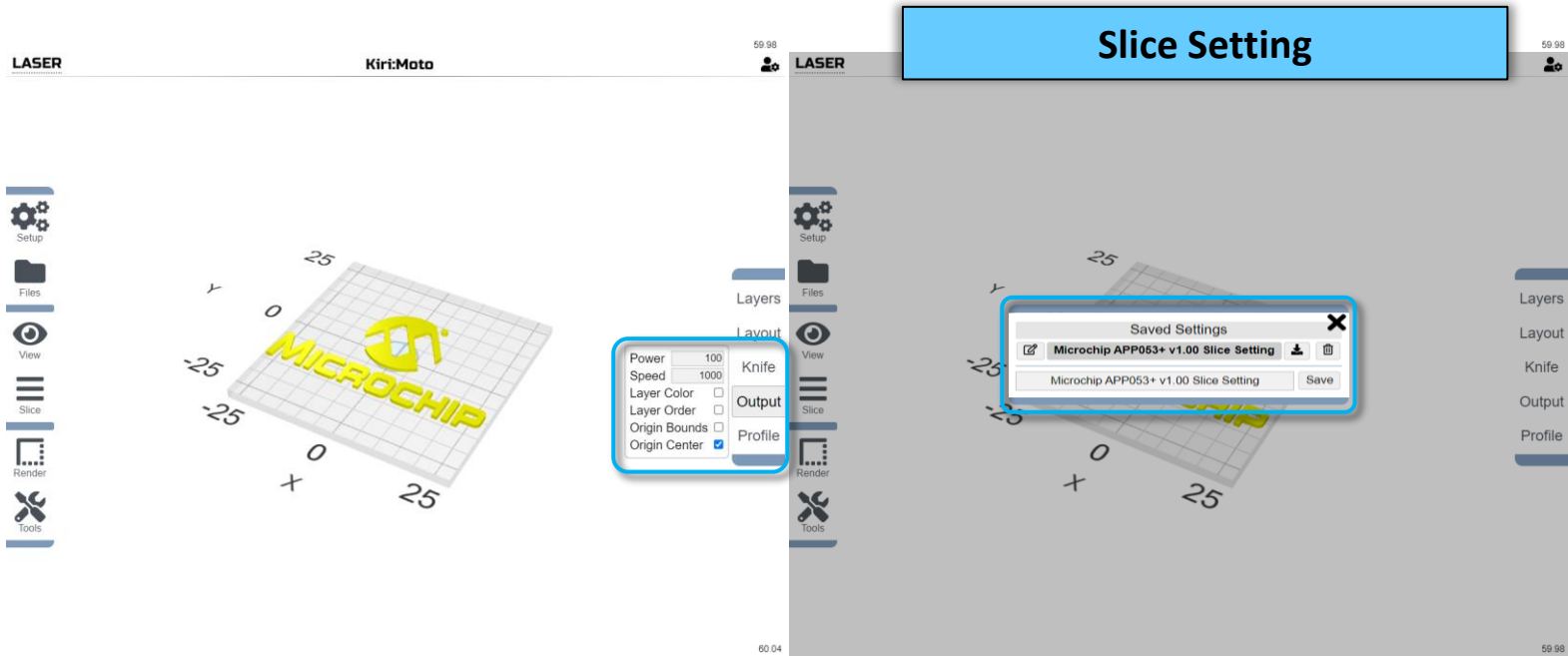
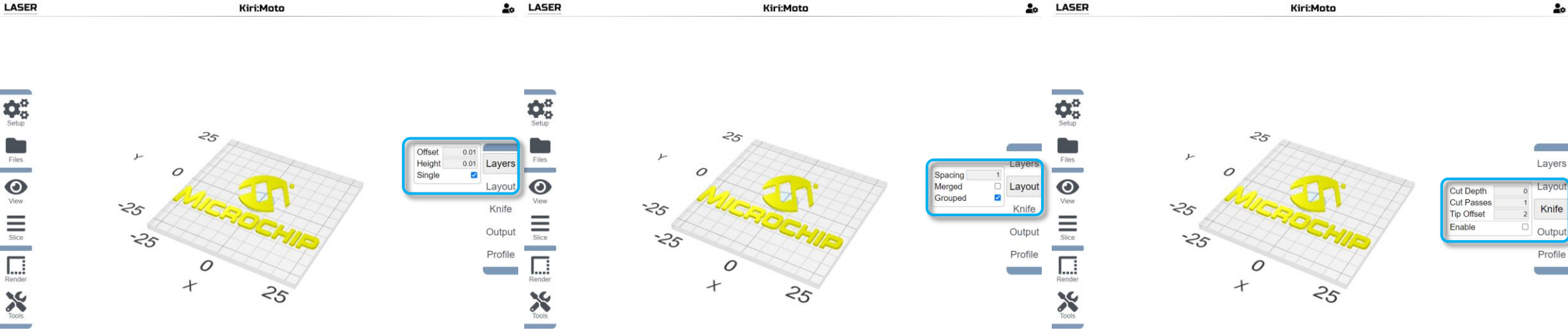
Kiri:Moto Introduction *cont.3*



Kiri:Moto Introduction *cont.4*



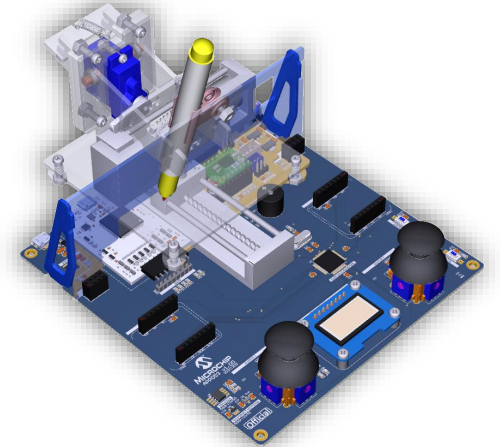
Kiri:Moto Introduction *cont.5*



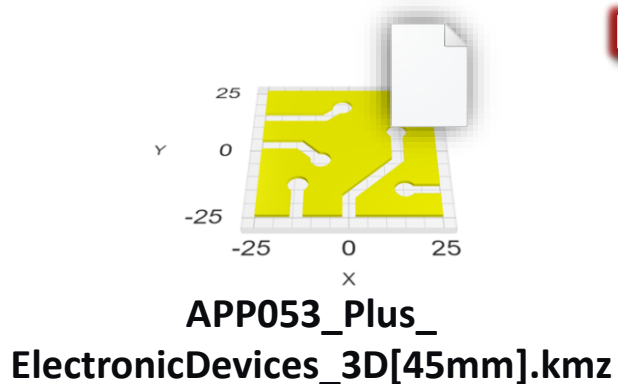
Exercise A

**Generate a G-code file at slice tool directly
by pre-saved workspace.**

Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.



XMODEM



Import

Slicing

Export



e.g.:
M106 S255 \n; Laser Trun On
G1 X23.152 Y36.349 F1000 \n
G1 X24.924 Y38.849 \n
M107 \n
G0 X29.816 Y19.279 \n
M106 S255 \n
G1 X29.932 Y19.288 F1000 \n
M107 \n
...

ElectronicDevices_3D.gcode

Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

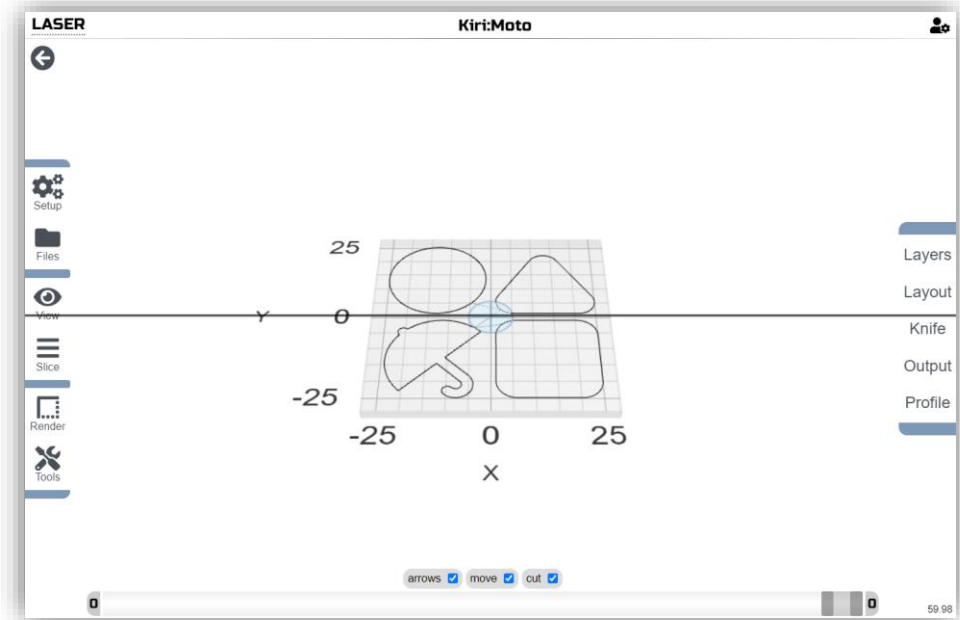
Before Exercises

- Kiri:Moto is browser-based application.
- you will get some problems sometimes from internet connection issues.

So, Always remember, re-fresh your browser.

you can solve most issues after re-fresh your browser.

F5 key



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Before Exercises

- I known that Some attender pre-study our slide already.
- **To prevent unexpected issues and non-standard step was occurred.
Please clear your Cookie before exercises.**

Clear Cookie

Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Before Exercises

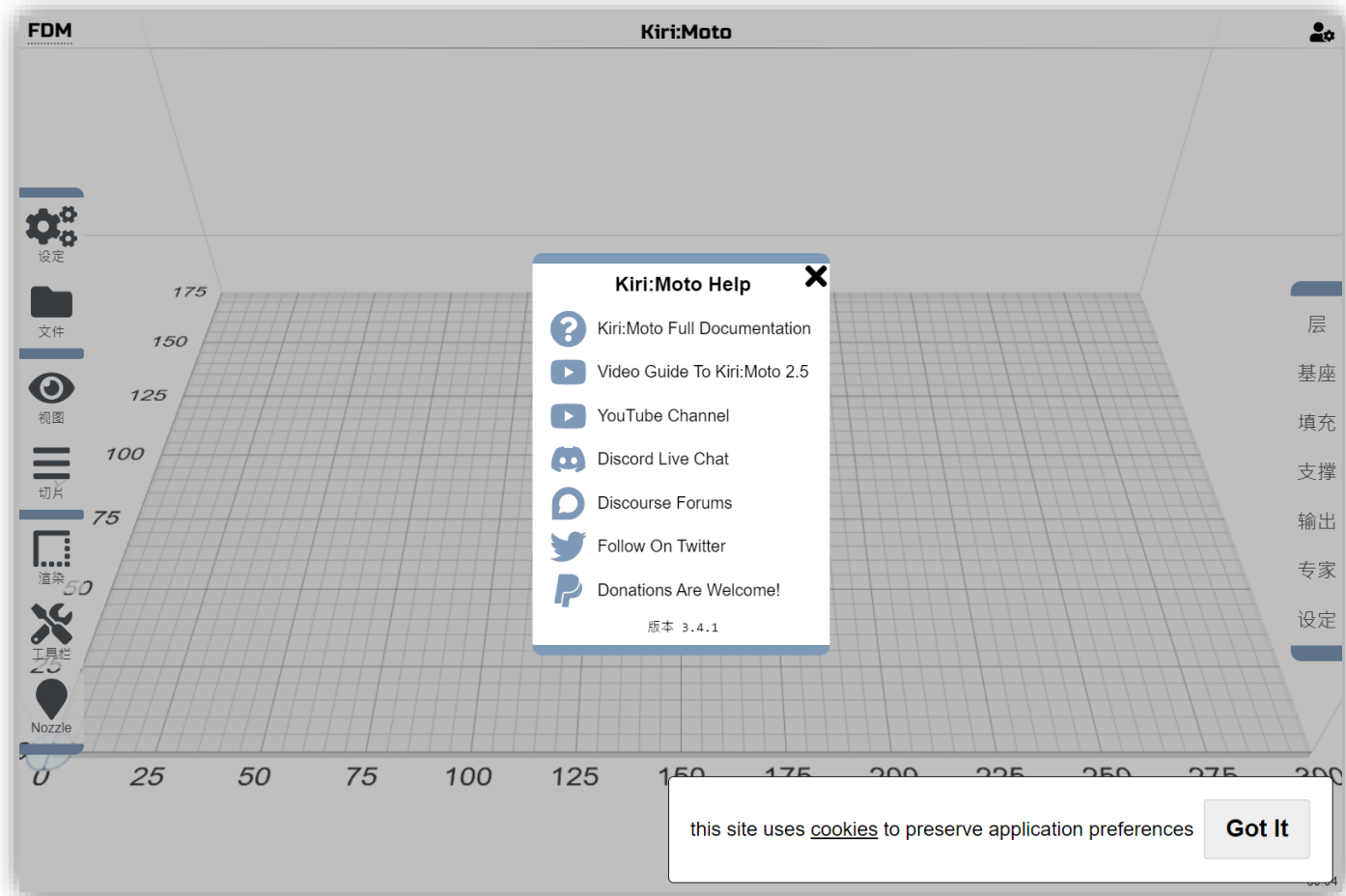
- e.g., clear cookie at Chrome
- Kiri:Moto
<chrome://settings/cookies/detail?site=grid.space>
- THINKCAD
<chrome://settings/cookies/detail?site=tinkercad.com>

**Copy & Paste to Chrome
to Clear Cookie**

Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Getting Started

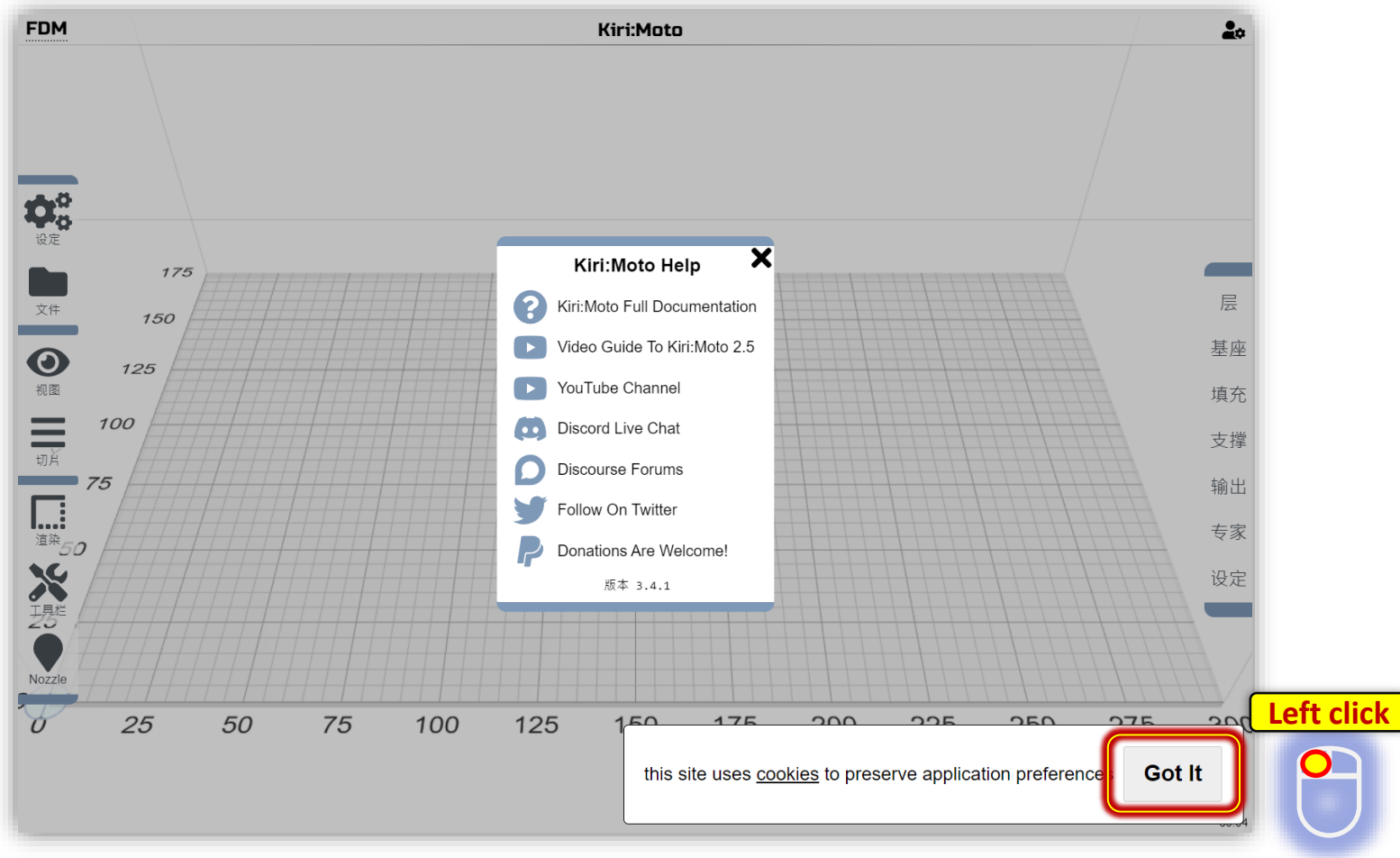
- Go to Kiri:Moto website (grid.space/kiri)



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Getting Started

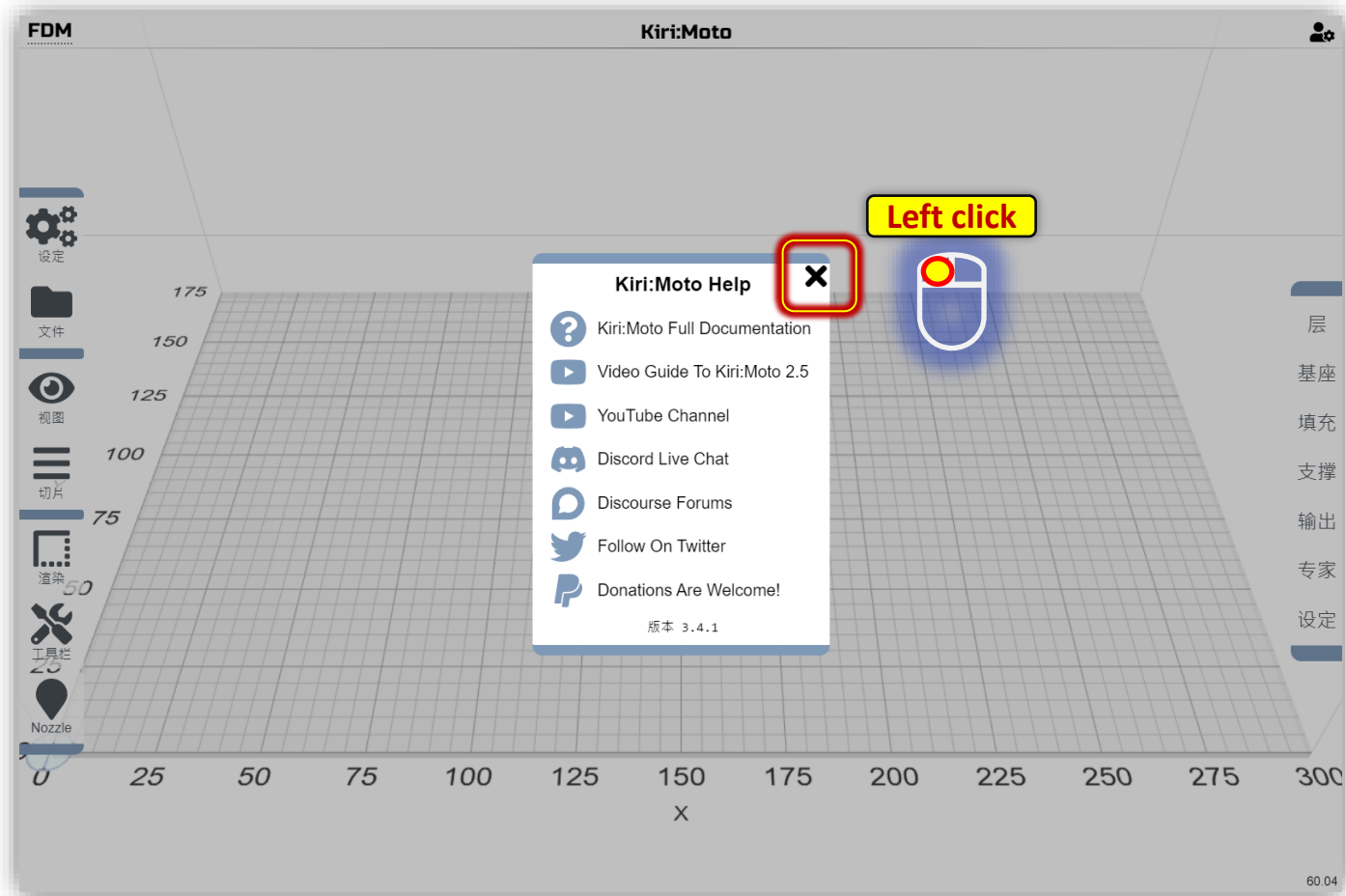
- Click “**Got it**” to start exercise



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Getting Started

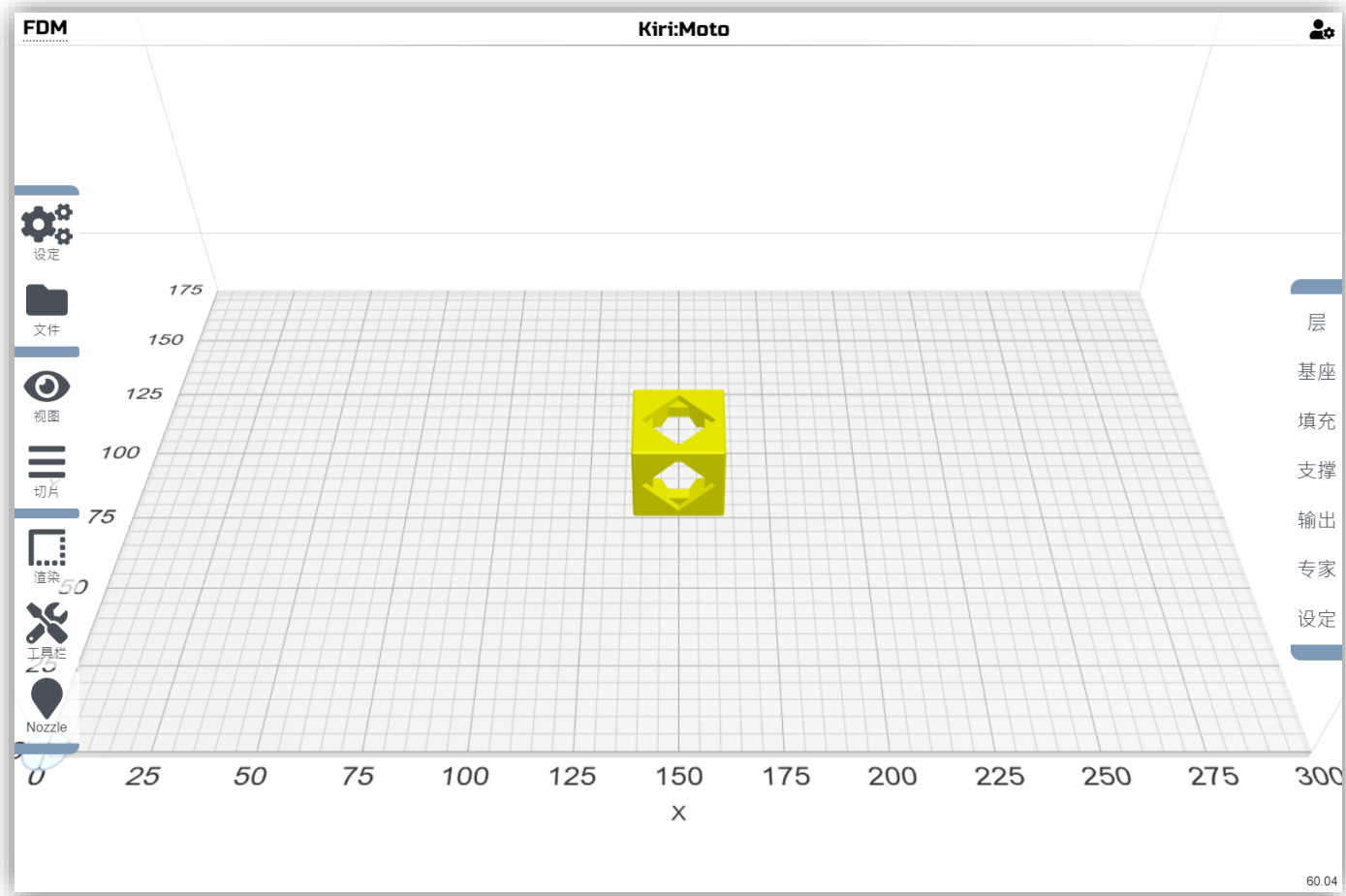
- Click “X” to start exercise



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Getting Started

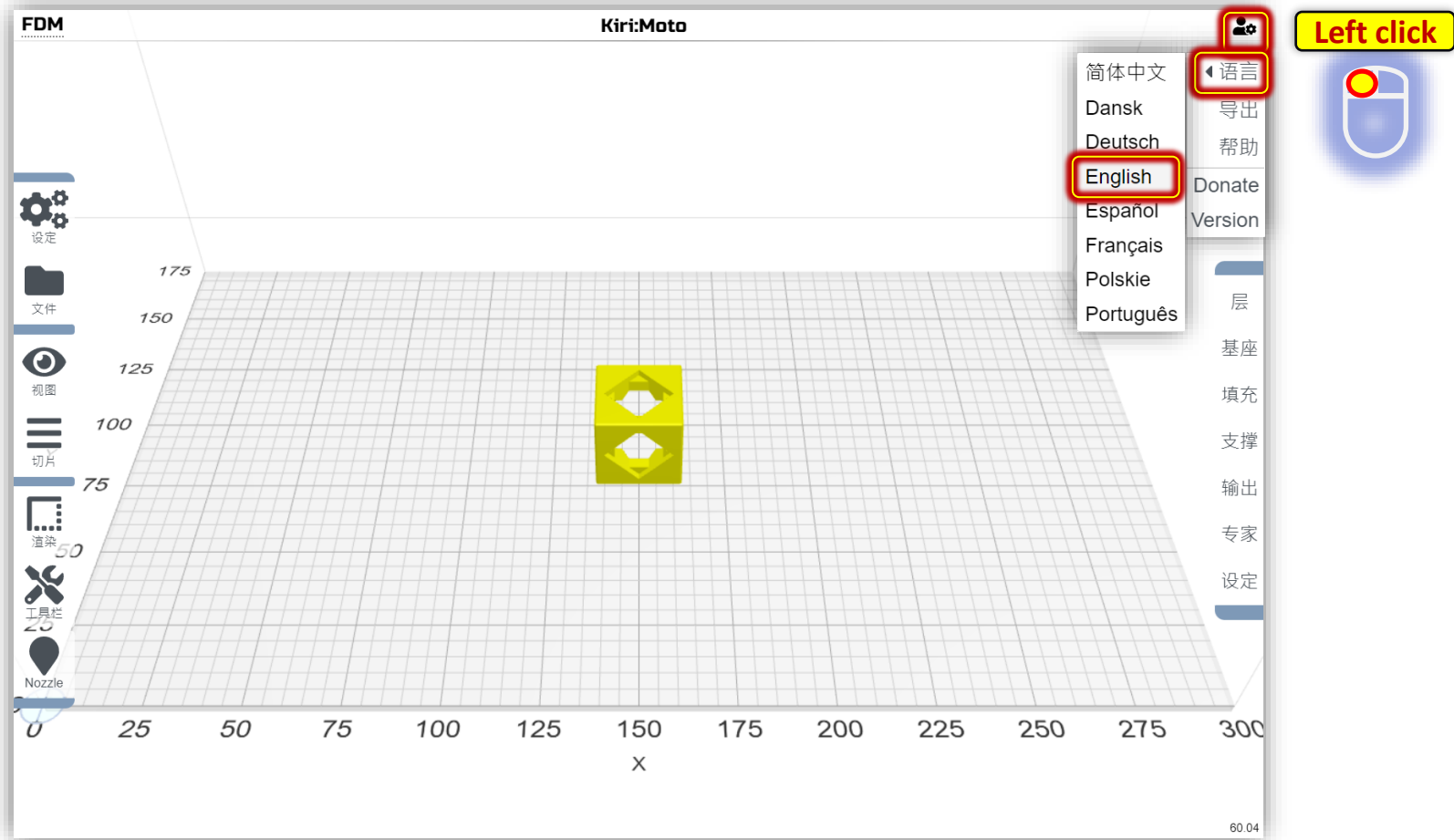
- Return to work panel.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 1 Set Language

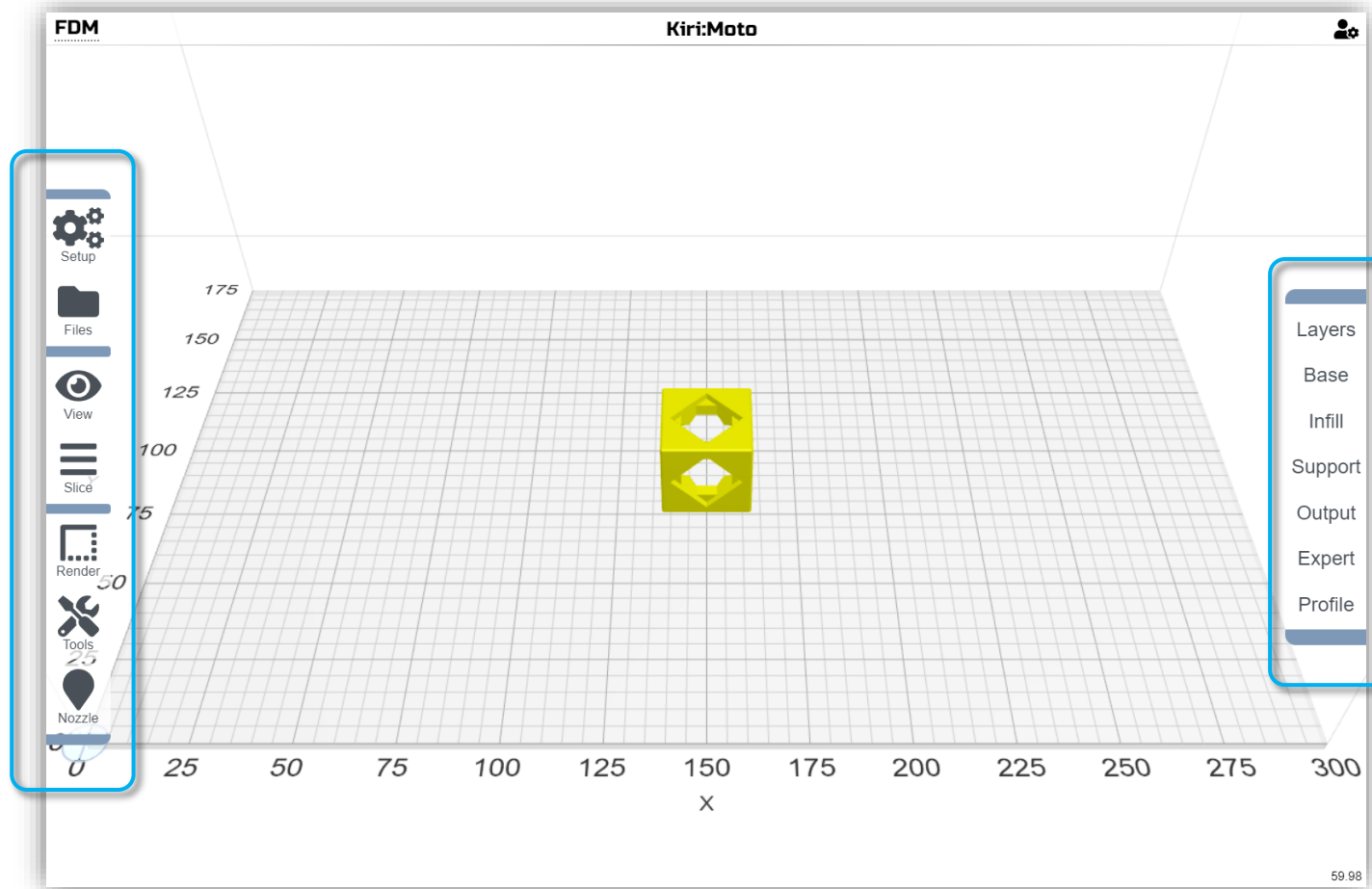
- Select **logo** > **Language** > **English**



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 1 Set Language

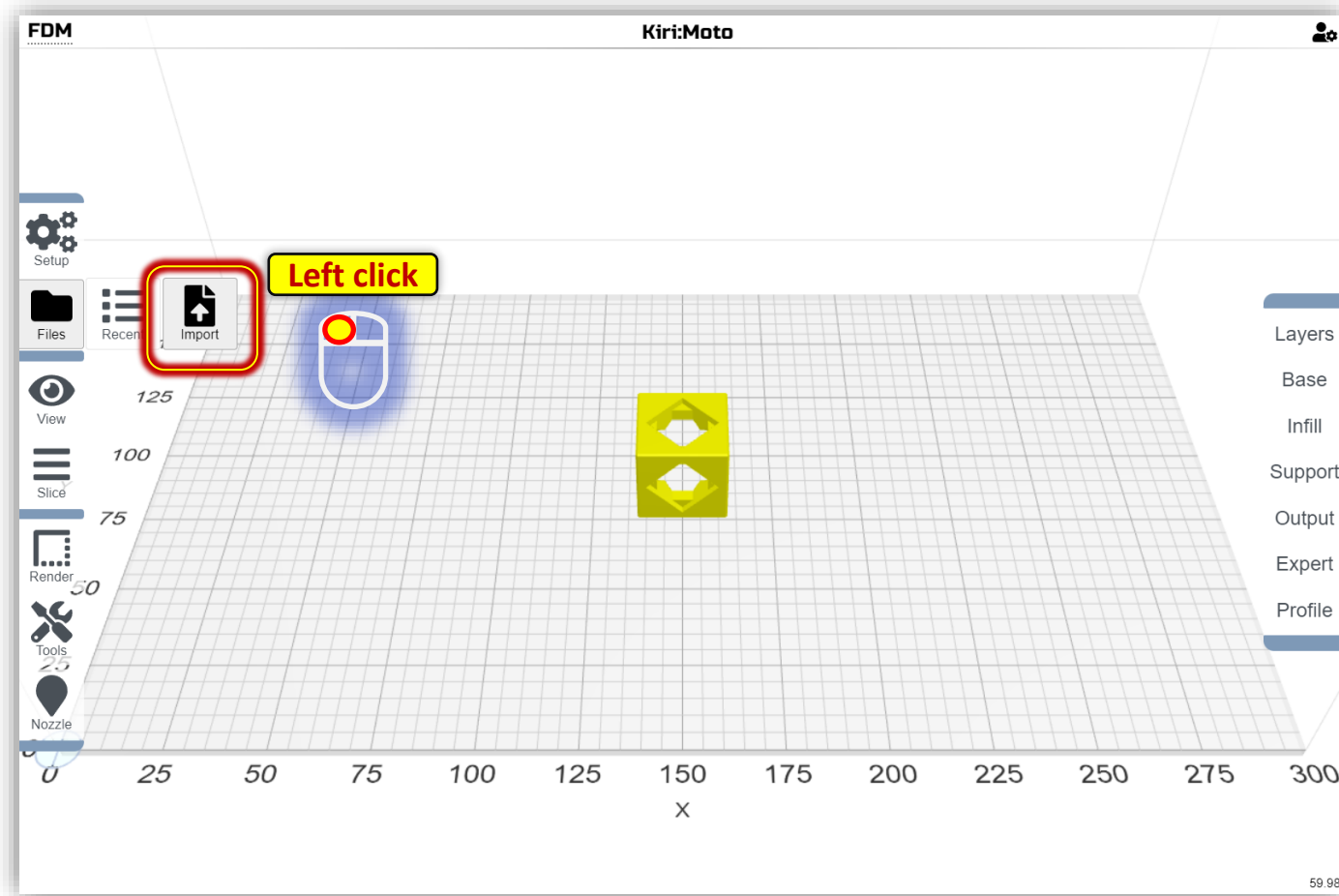
- Interface was changed to English.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 2 Import Exist Project

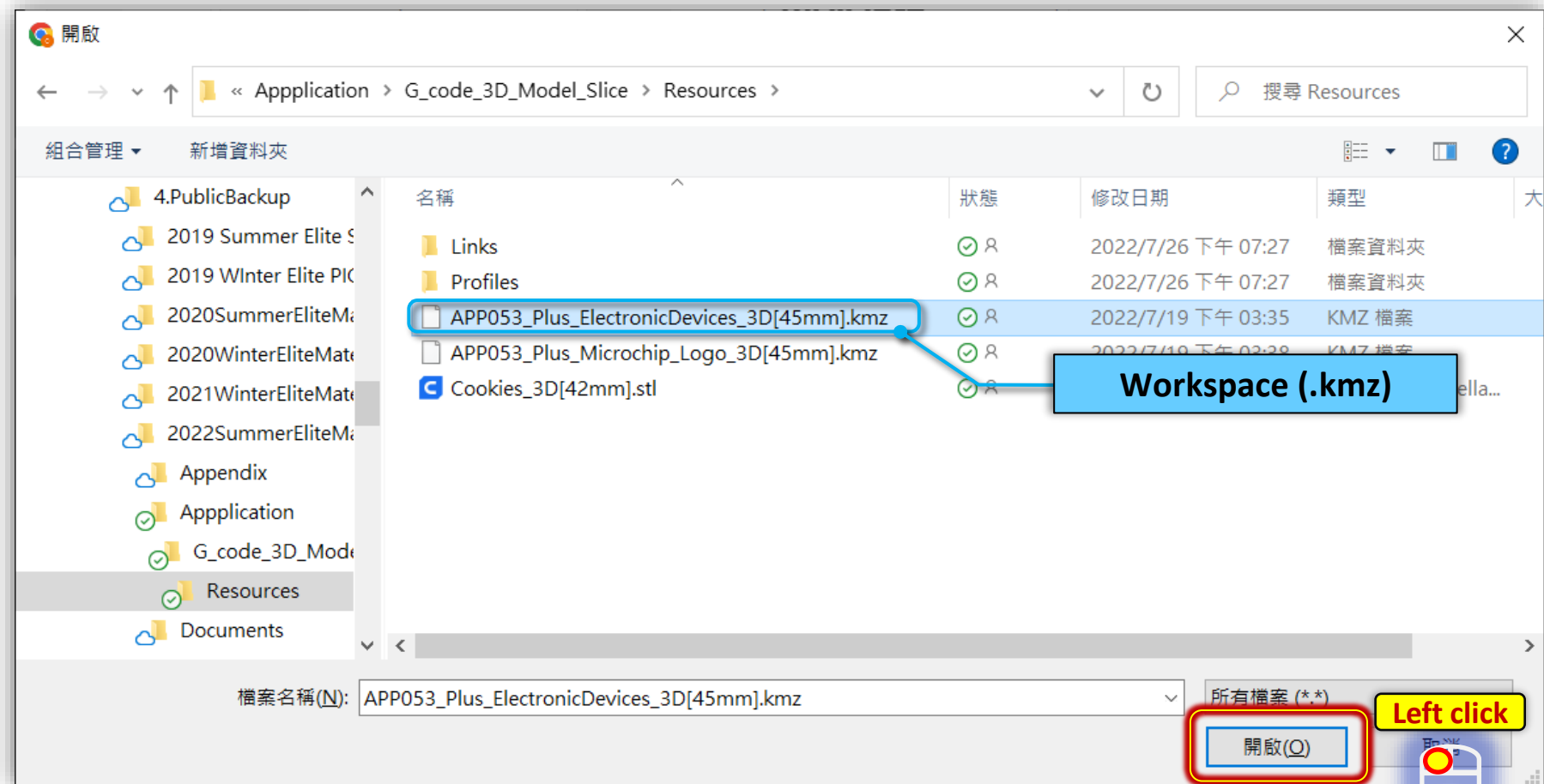
- Select **Files** > **Import**



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 2 Import Exist Project

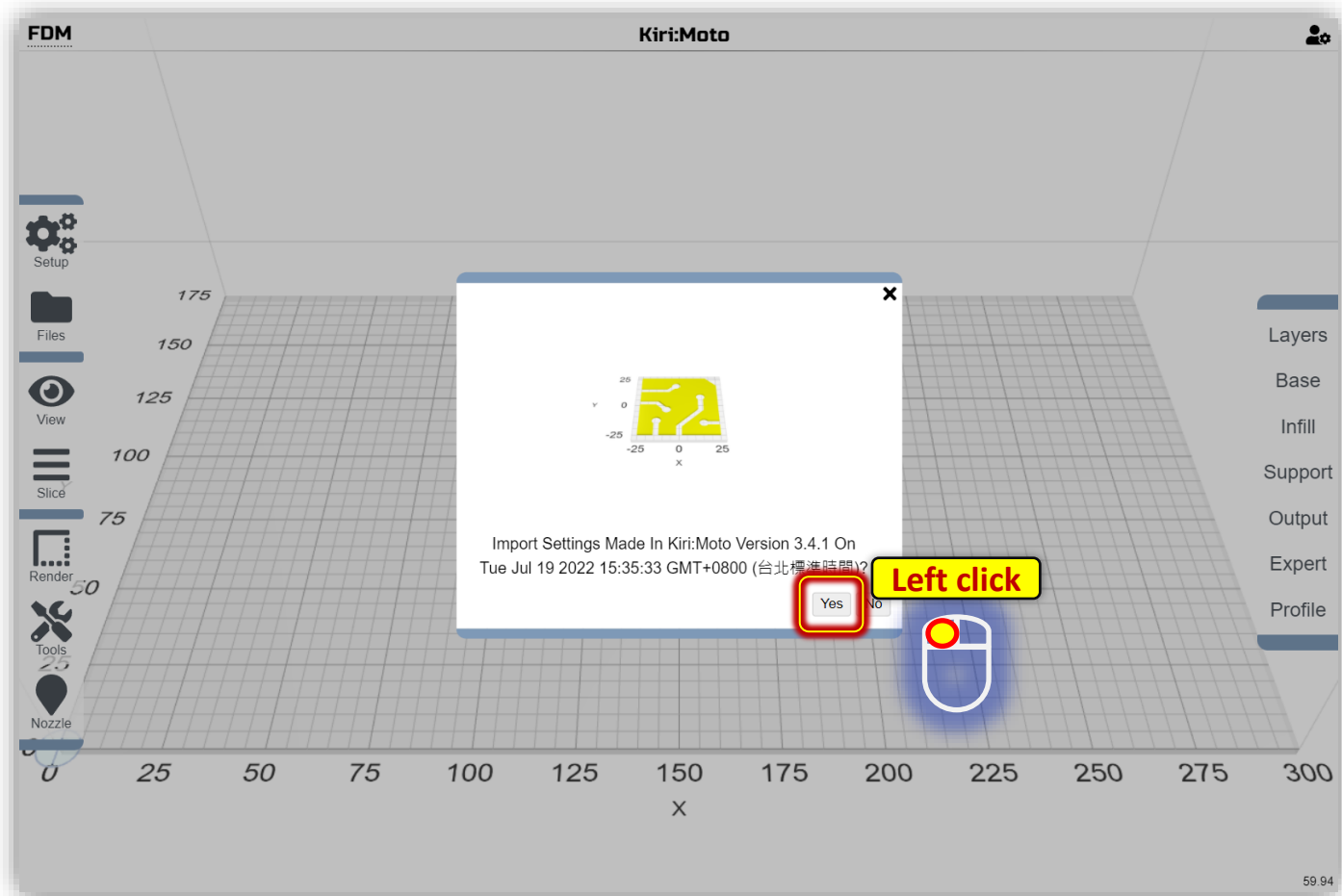
- Select “.\Resources\APP053_Plus_ElectronicDevices_3D[45mm].kmz” and click “開啟”



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 2 Import Exist Project

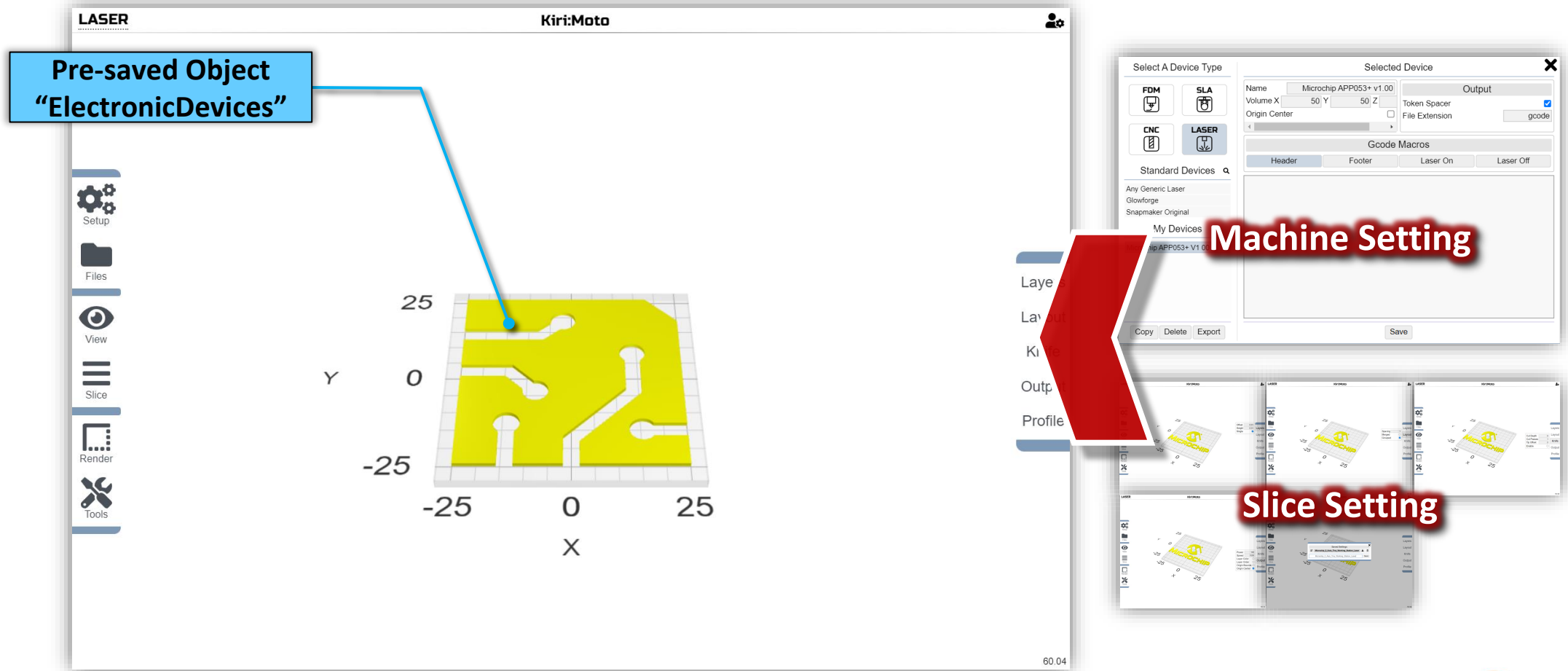
- Select “Yes” to load project file (.kmz)



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 2 Import Exist Project

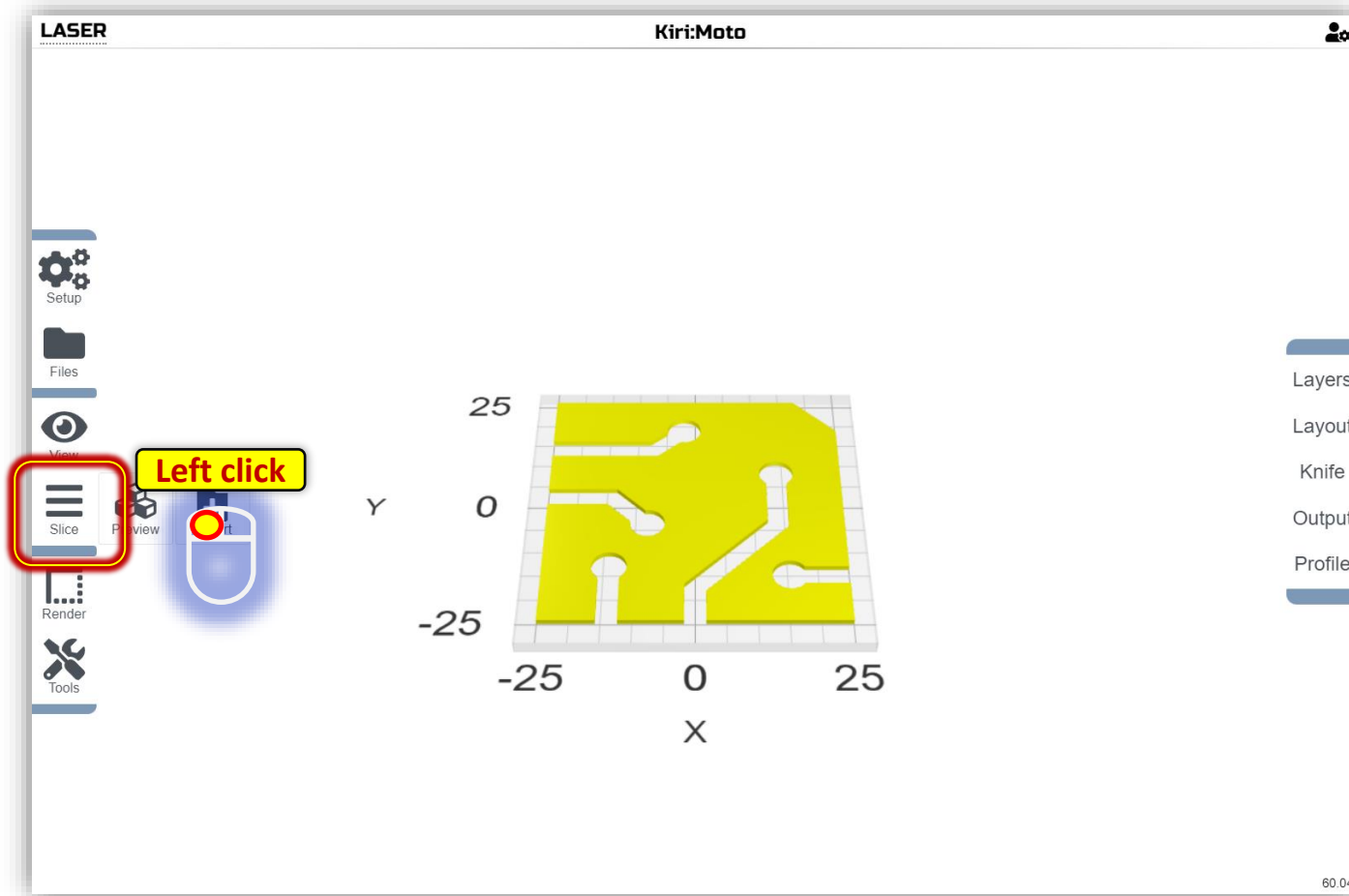
- Pre-saved project was loaded. The machine, slice setting, and object was load also.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 3 Slice the Object

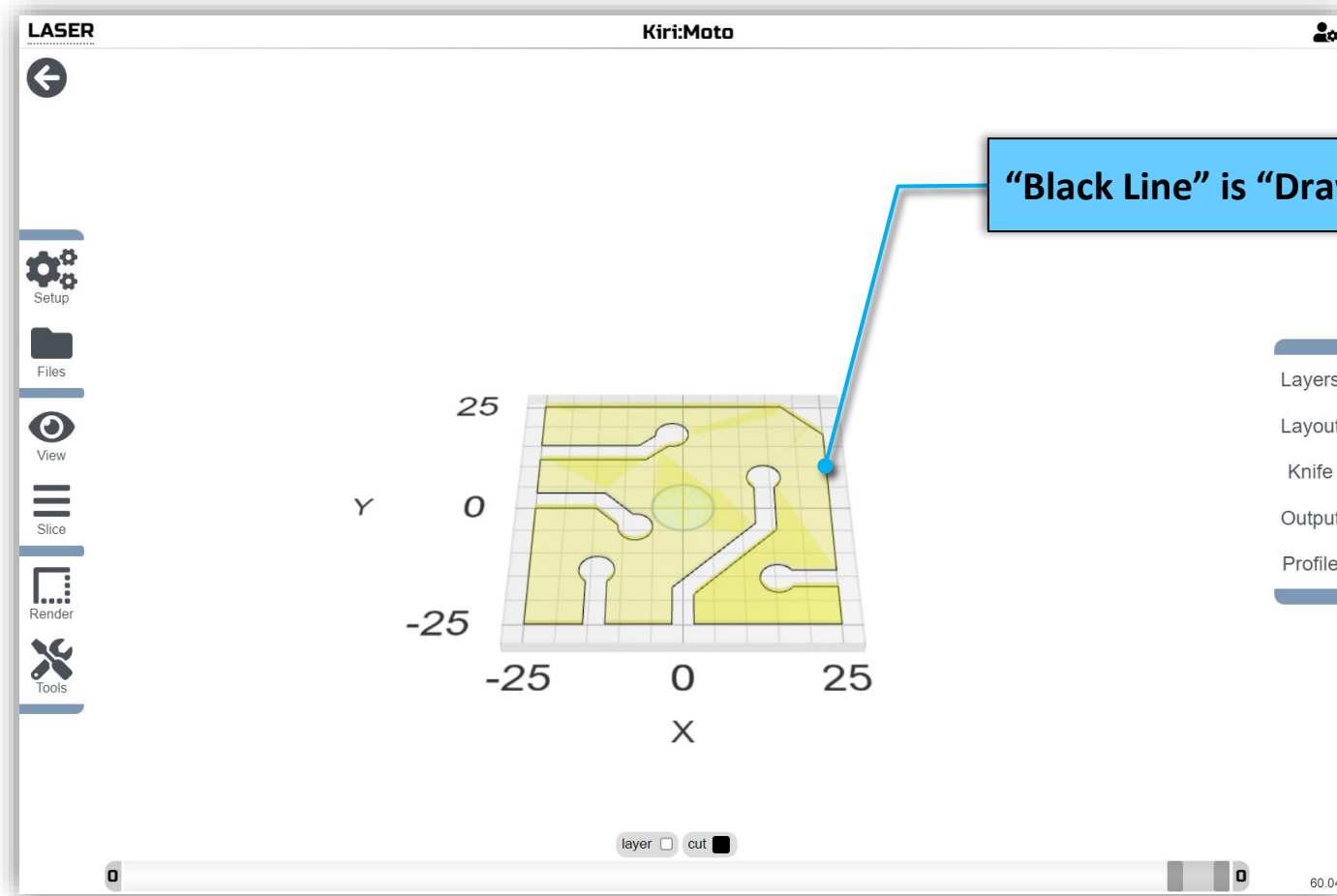
- Select **Slice**



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 3 Slice the Object

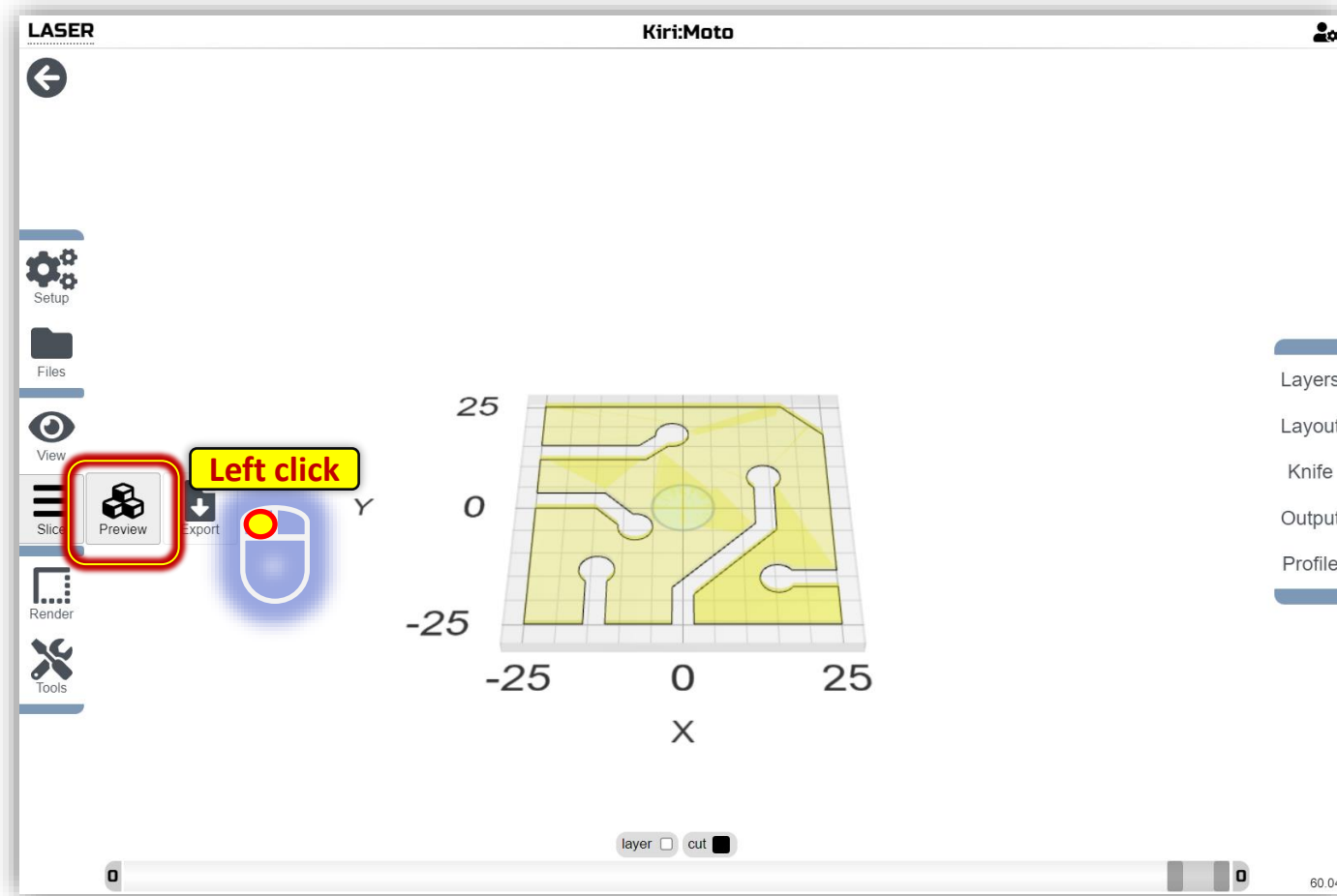
- Object was sliced.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 4 Preview the “path”

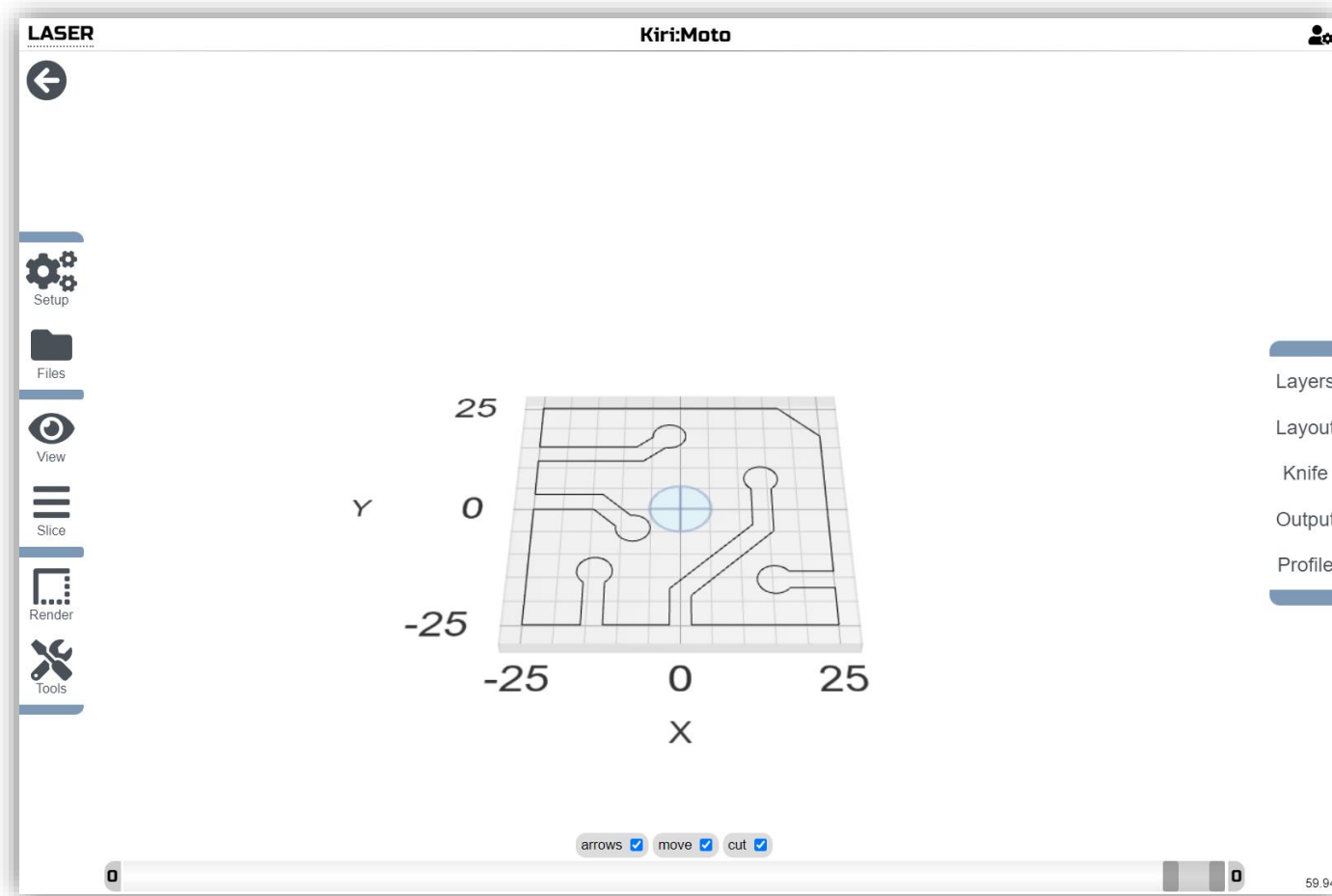
- Select **Slice** > **Preview**



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 4 Preview the “path”

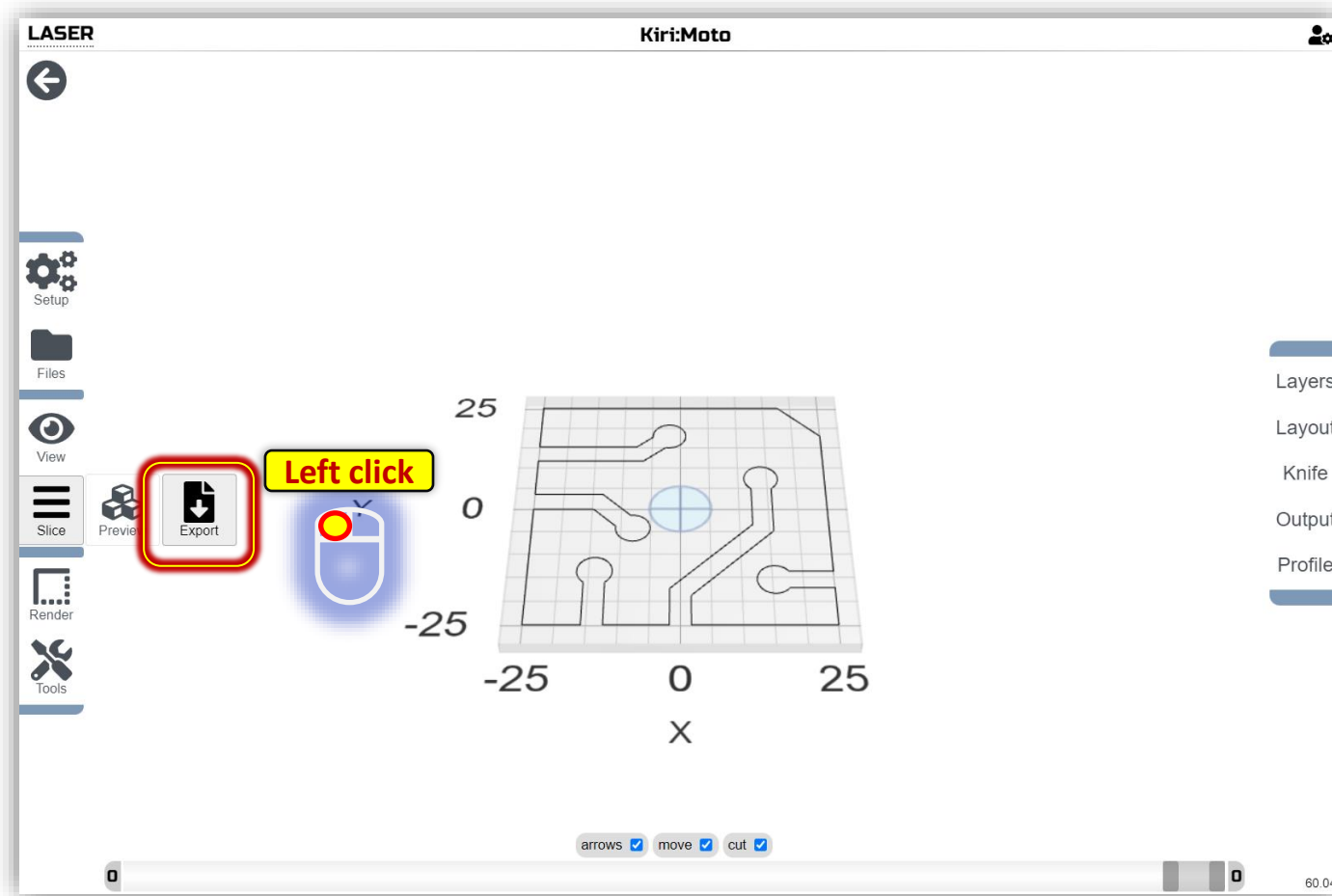
- Shows the “path” only.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 5 Export “path” to G-code

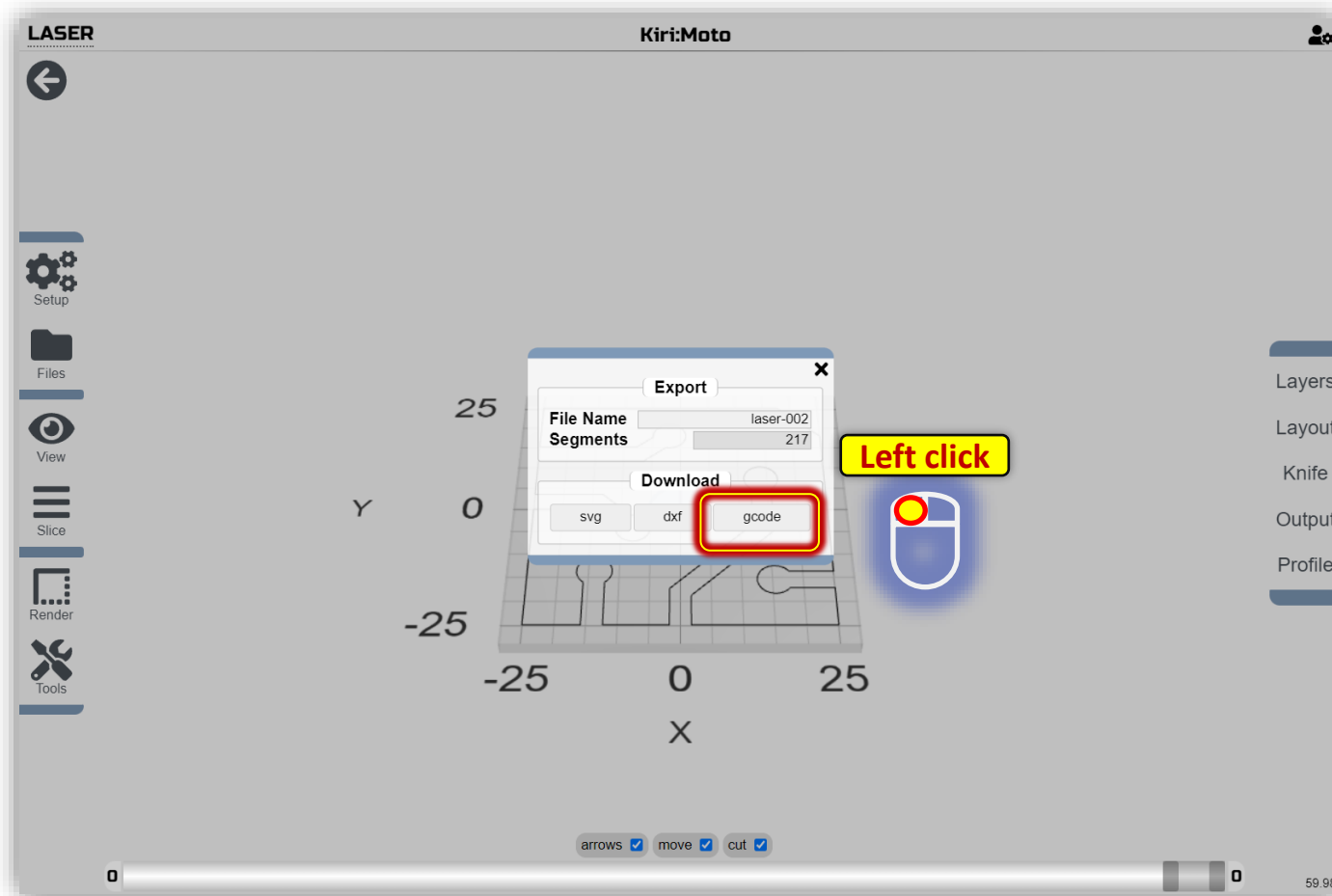
- Select **Slice** > **Export**



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 5 Export “path” to G-code

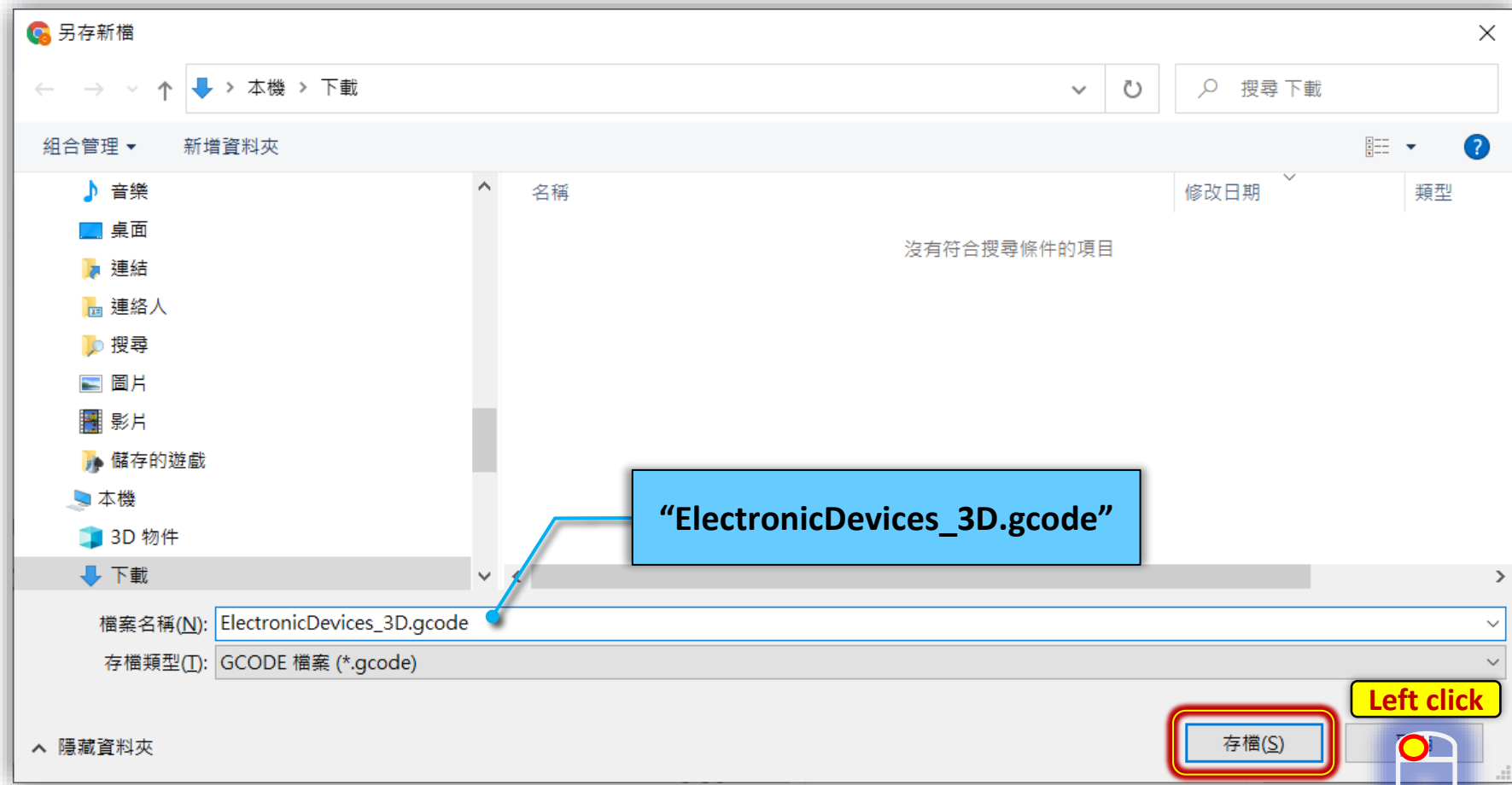
- Select “**gcode**” directly



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 5 Export “path” to G-code

- Select local **folder**, **rename** and **save** file(.gcode) then select “**確定**”.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 5 Export “path” to G-code

- The G-code file was downloaded. Then Select “X” directly



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Step 5 Export “path” to G-code

- The “ElectronicDevices_3D.gcode” file was saved.



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

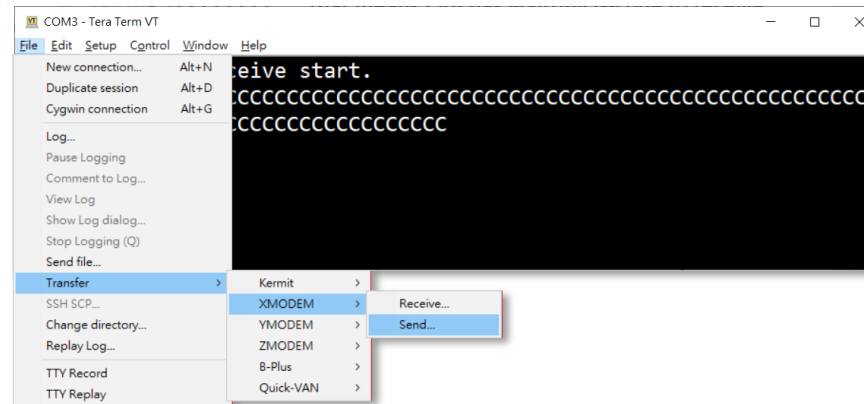
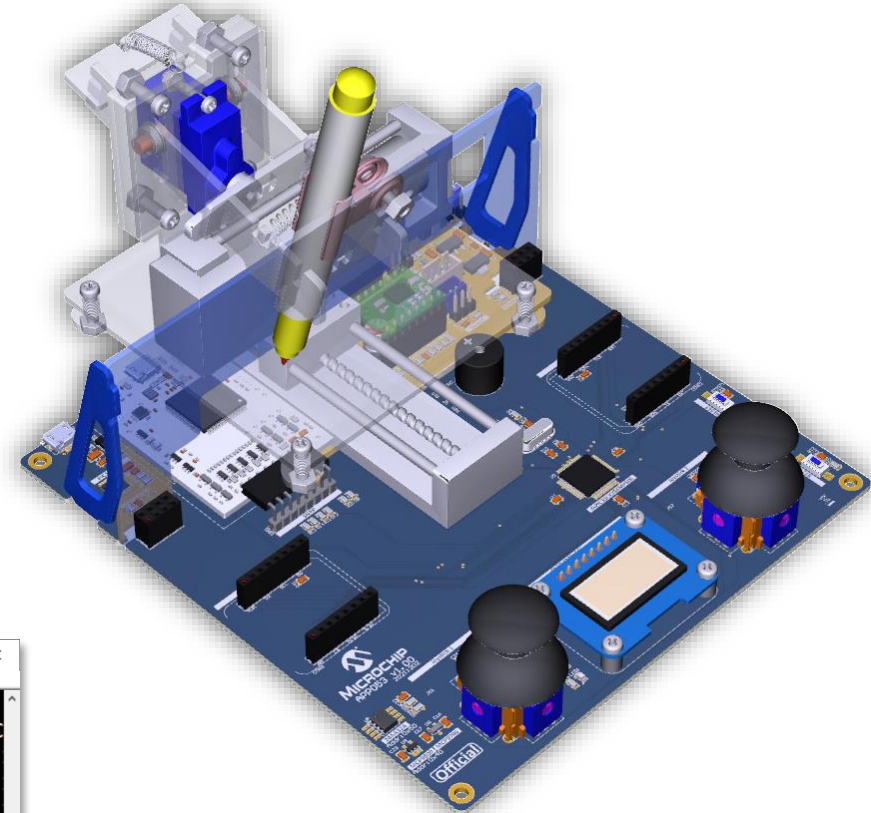
Finally, Draw the G-code

- Send the G-code to through XMODEM & enjoy.

e.g.:

```
M106 S255 \n; Laser Trun On
G1 X23.152 Y36.349 F1000 \n
G1 X24.924 Y38.849 \n
M107 \n
G0 X29.816 Y19.279 \n
M106 S255 \n
G1 X29.932 Y19.288 F1000 \n
M107 \n
...
```

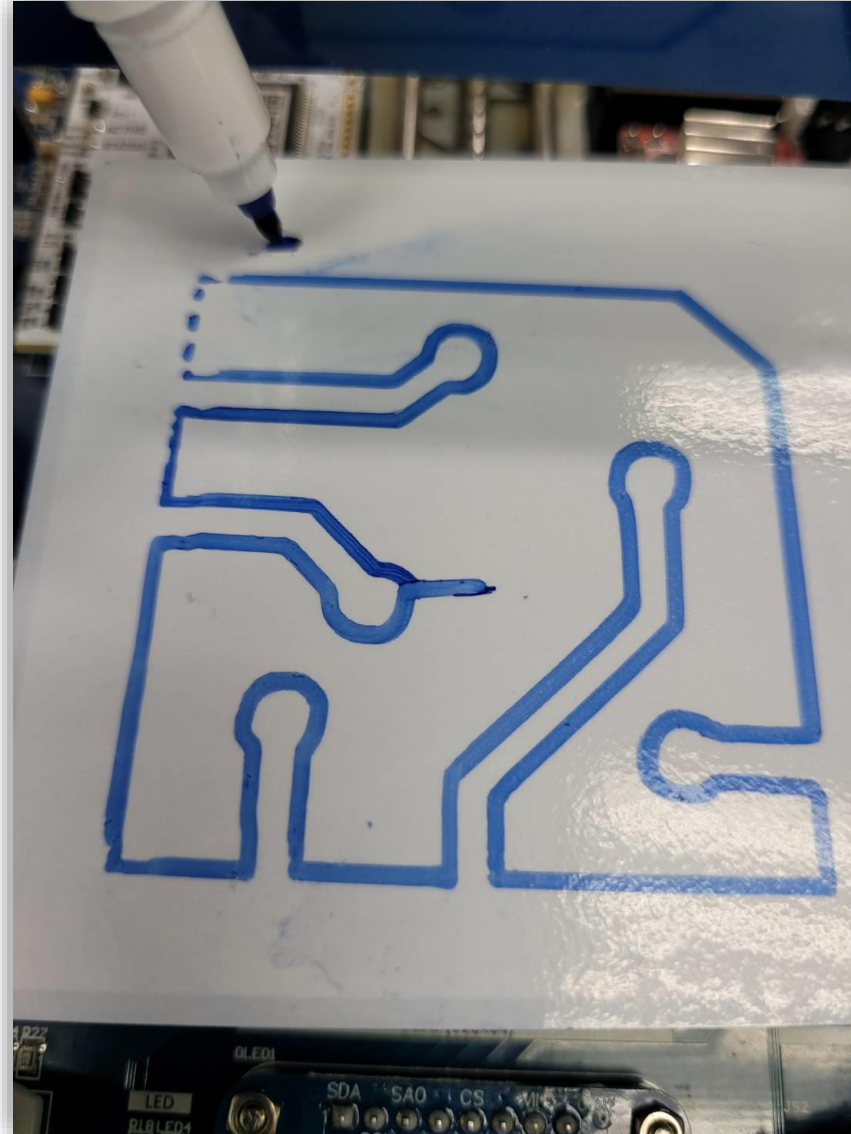
ElectronicDevices_3D.gcode



Exercise A Generate a G-code file at slice tool directly by pre-saved workspace.

Finally, Draw the G-code

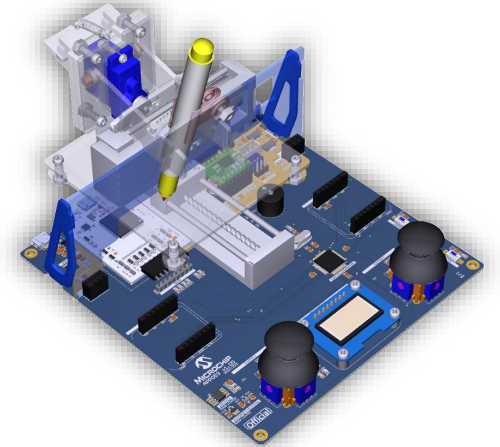
- Result



Exercise B

**Generate a G-code file at slice tool directly
by import 3D Object (.stl).**

Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)



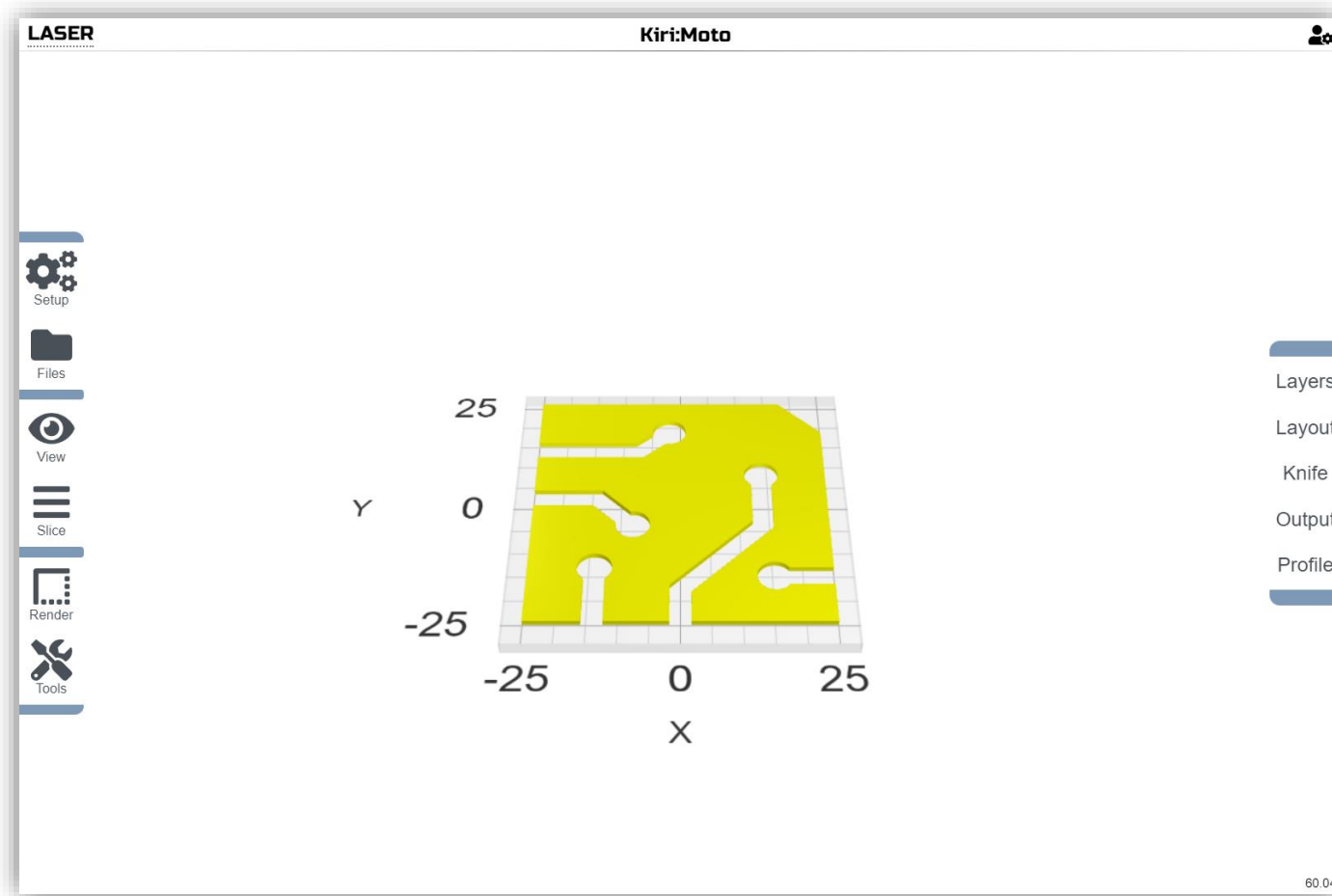
XMODEM



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Getting Started

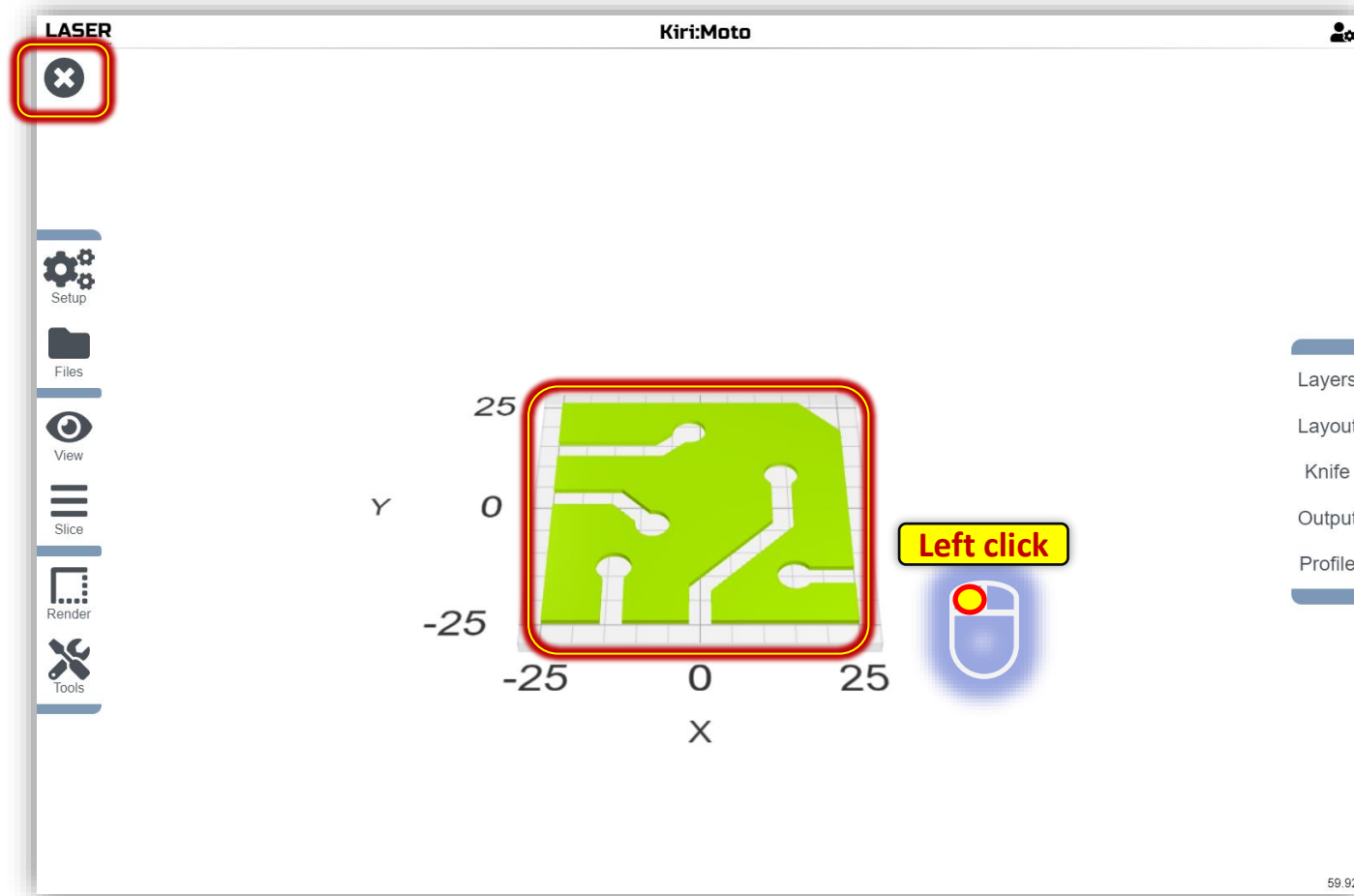
- Go backup to Kiri:Moto website (grid.space/kiri)



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 1 Remove Exist Object

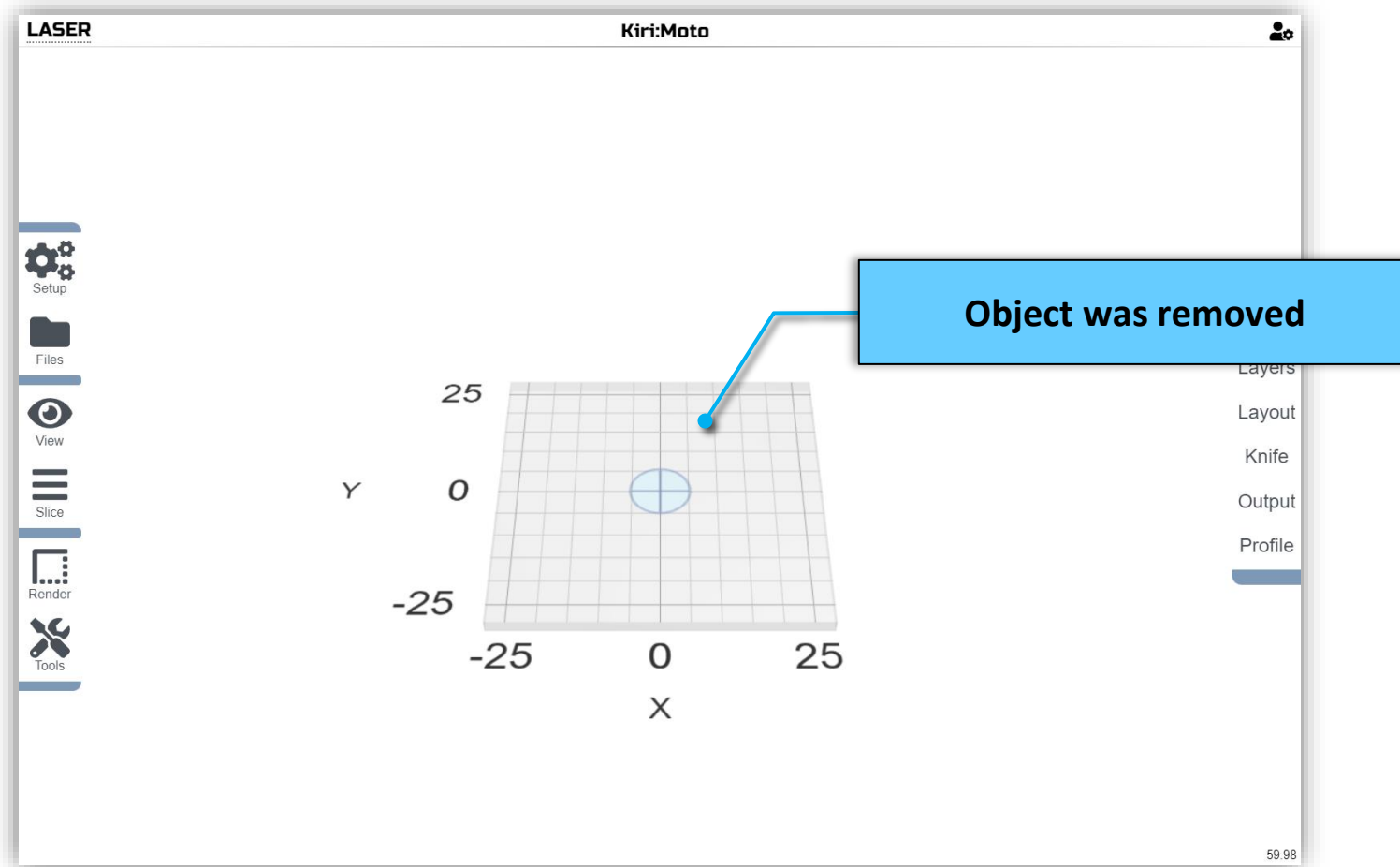
- Click the exist object then click “X”



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 1 Remove Exist Object

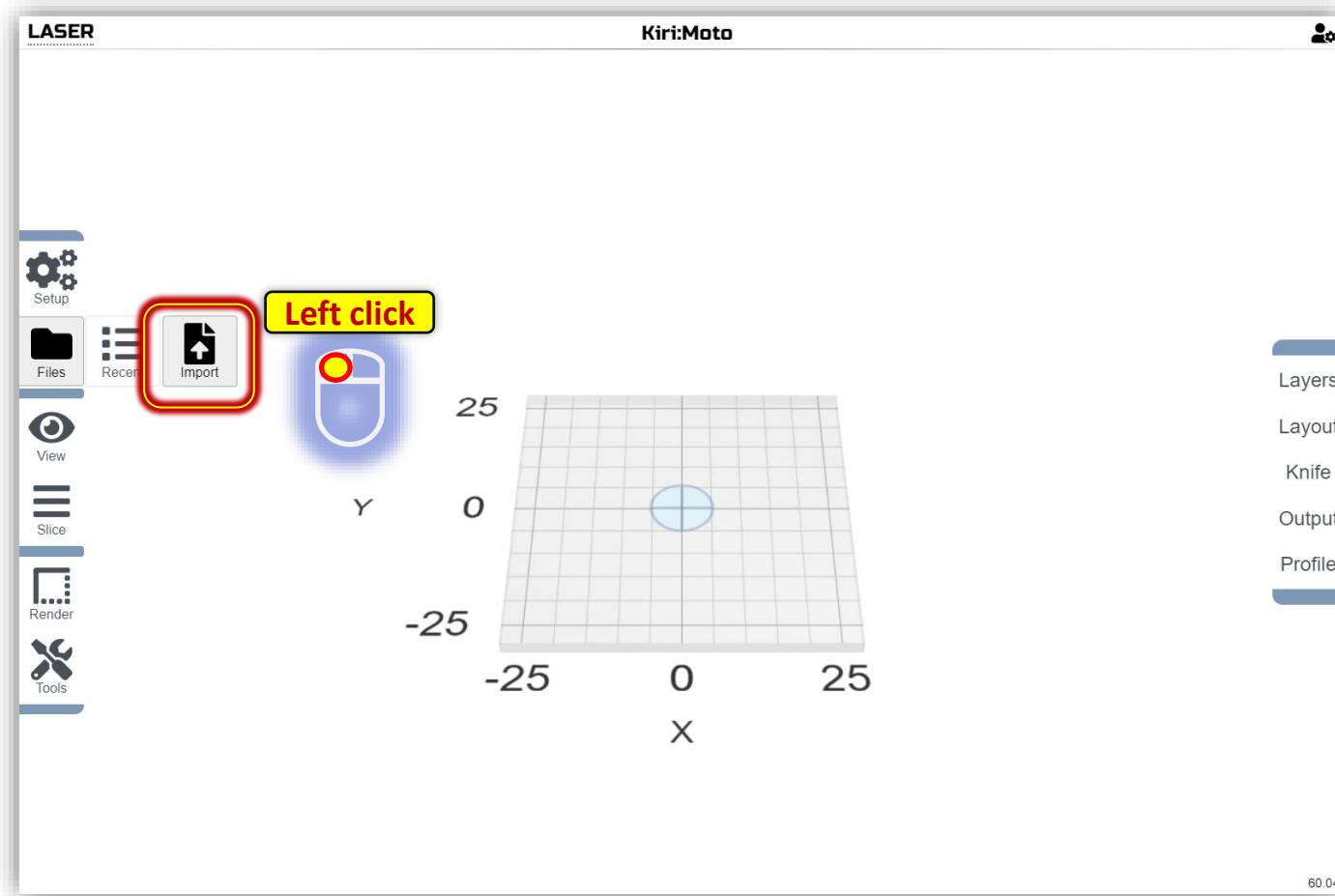
- The object was removed.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 2 Import Exist 3D Object

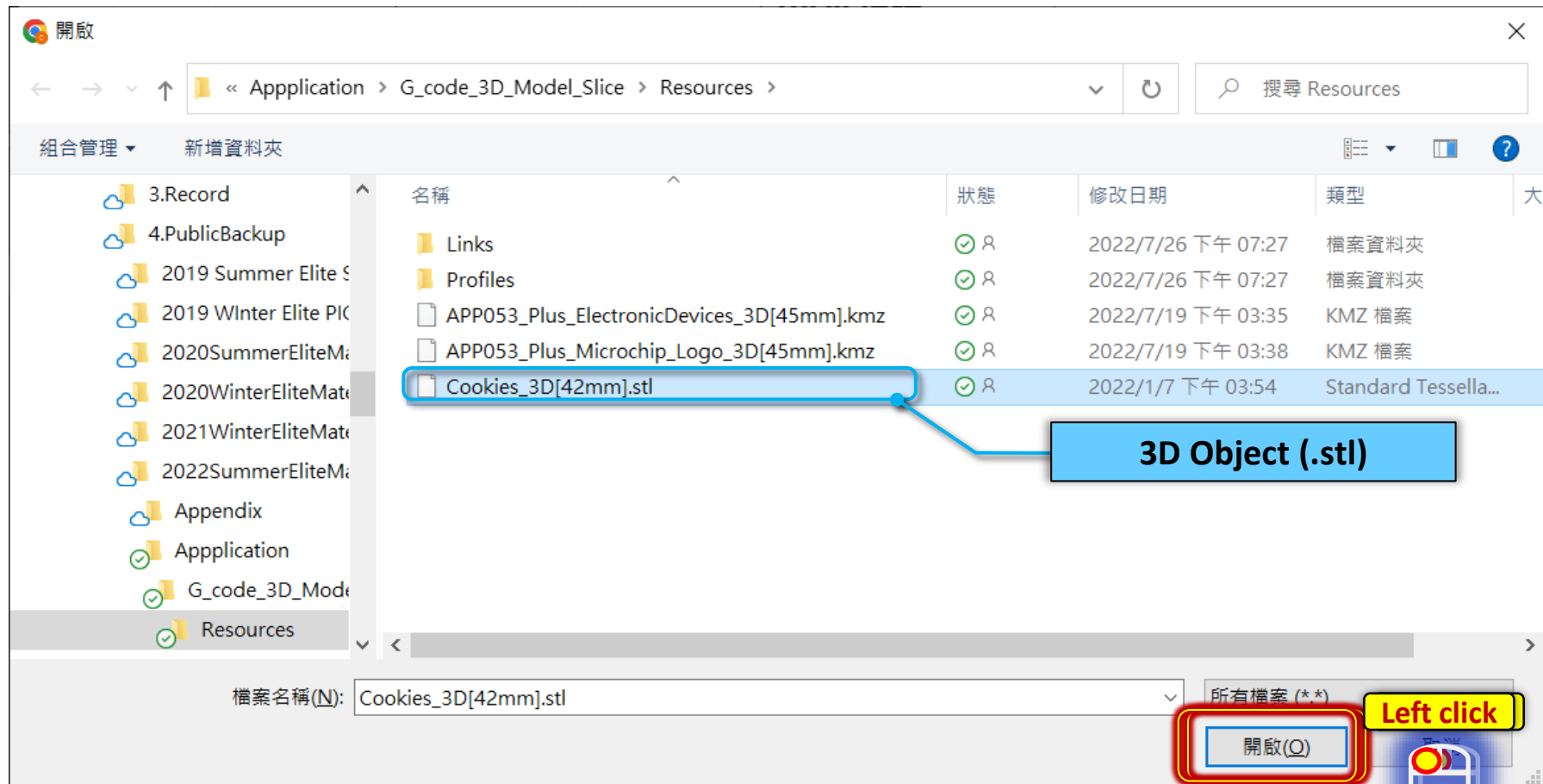
- Select **Files** > **Import**



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 2 Import Exist 3D Object

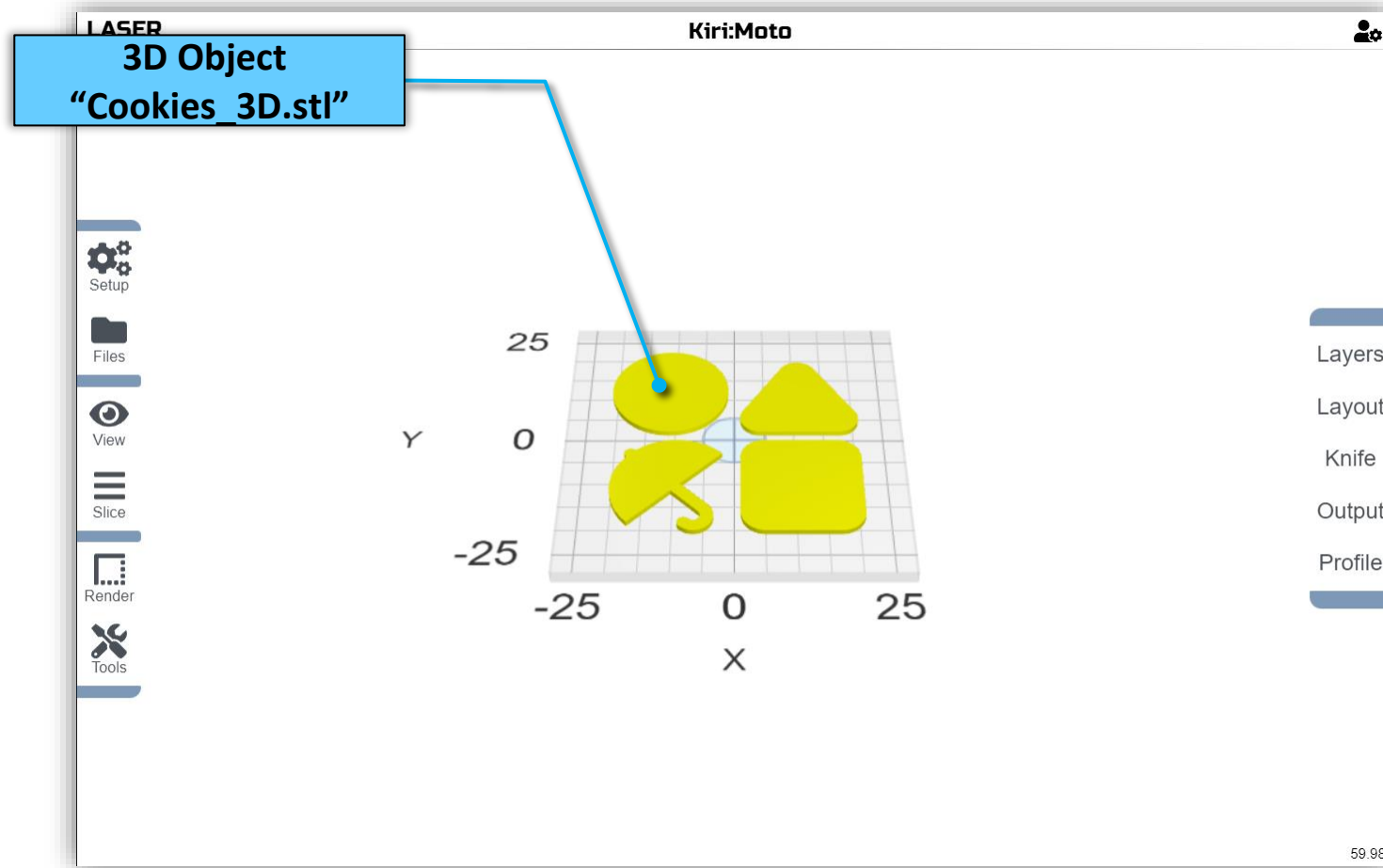
- Select “.\Resources\Cookies_3D[42mm].stl” and click “開啟”



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 2 Import Exist 3D Object

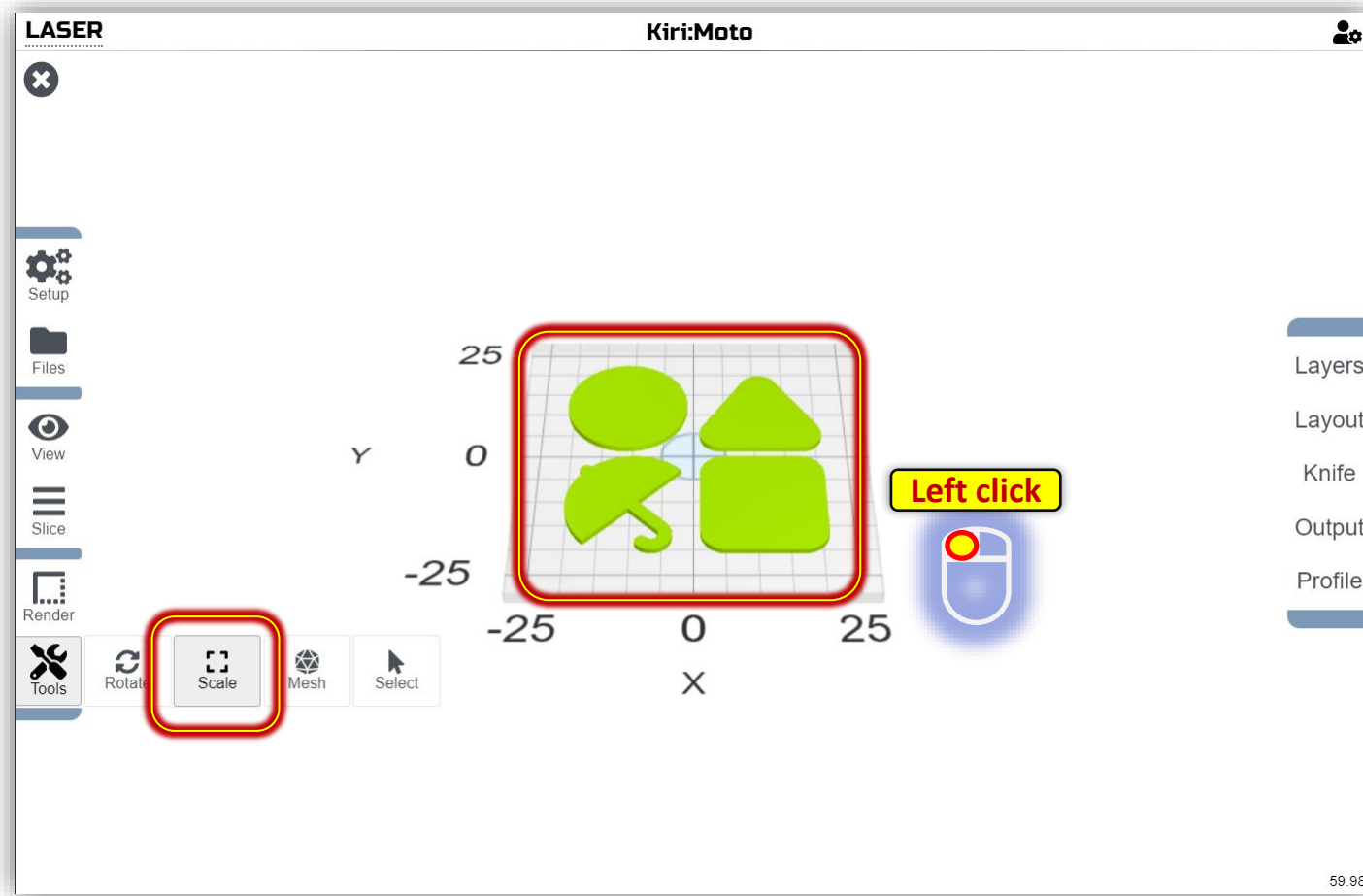
- 3D object was loaded.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 3 Scaling The Object

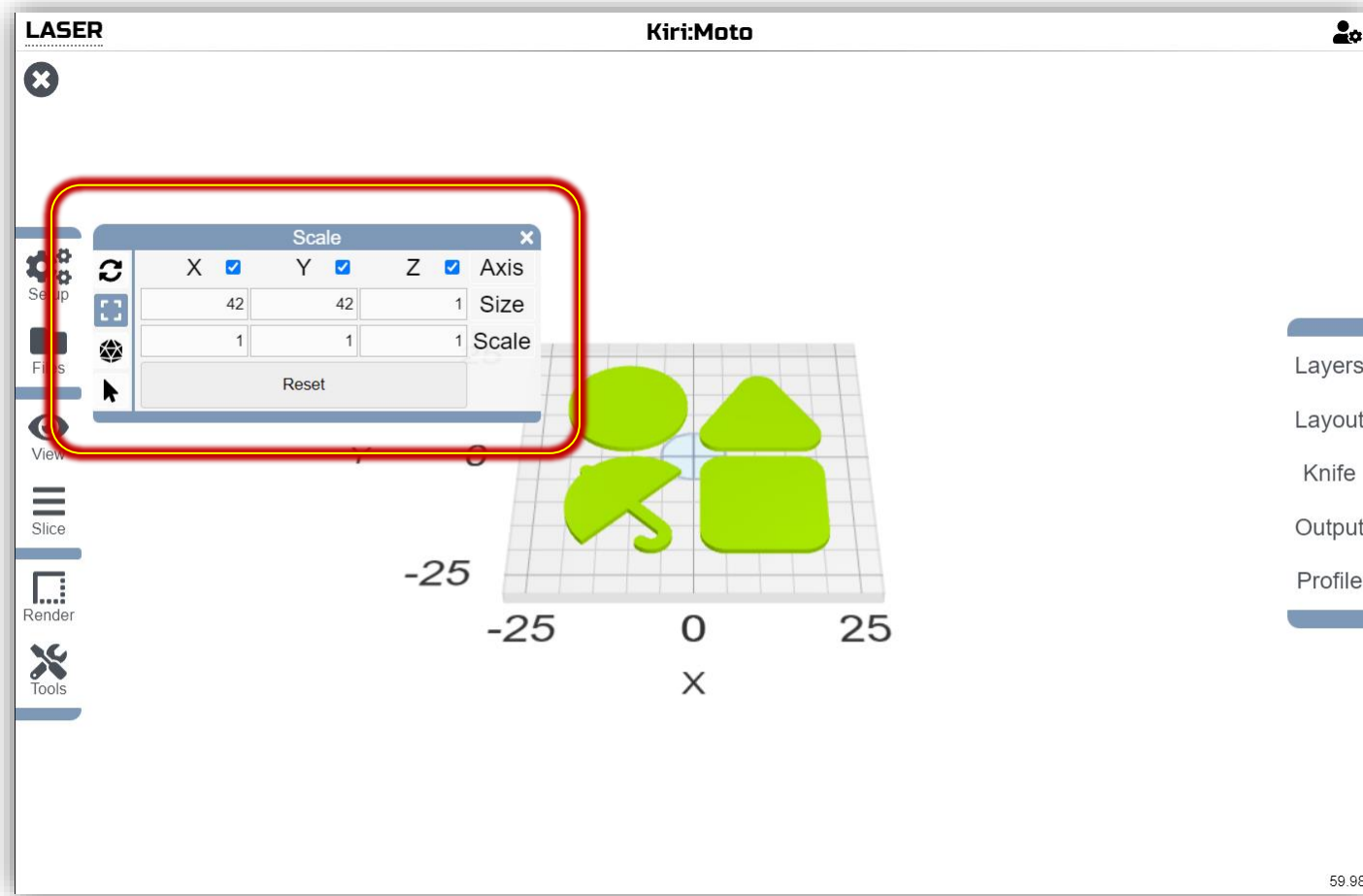
- Click the exist object , then Select **Tools > Scale**



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 3 Scaling The Object

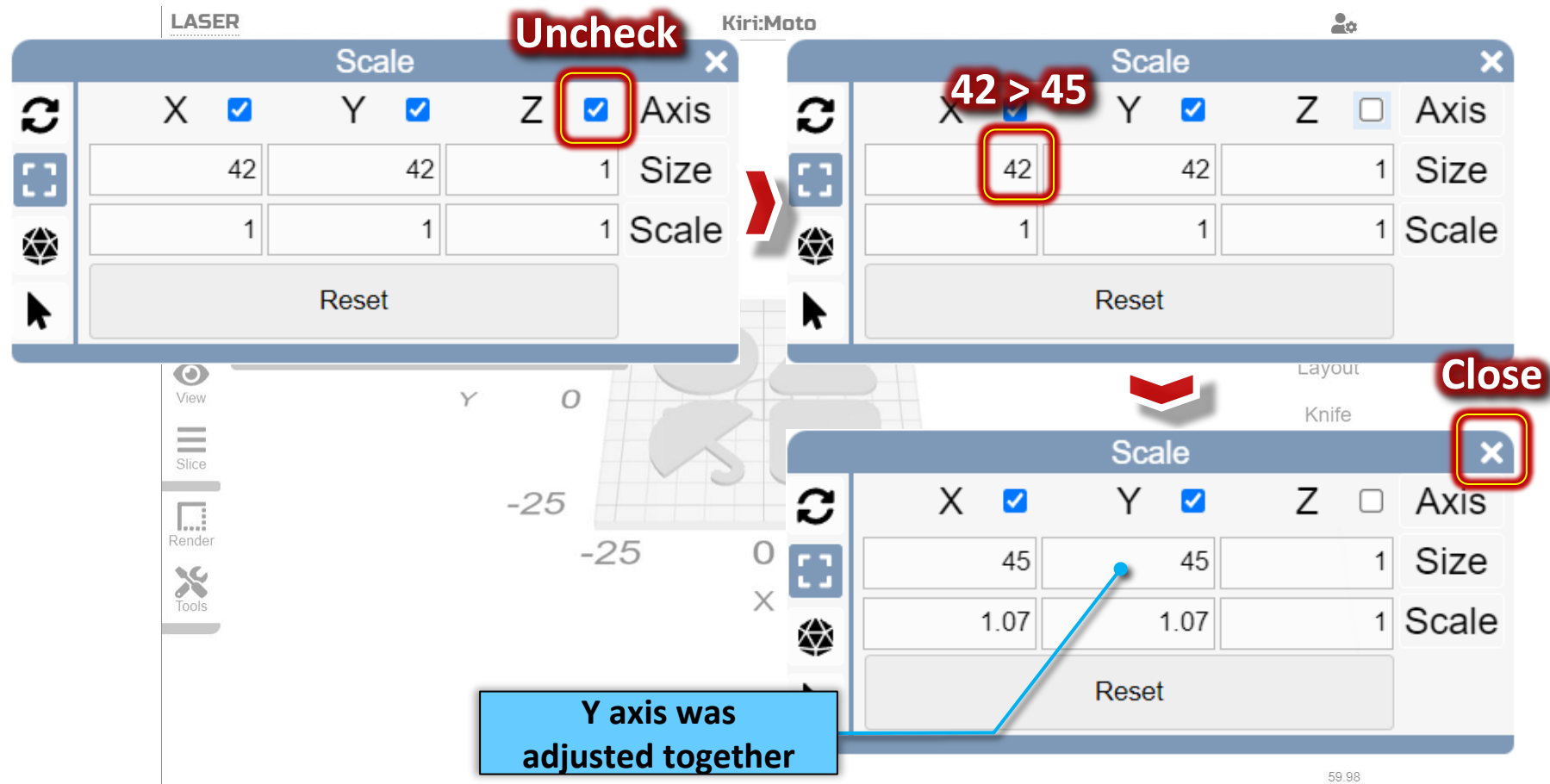
- Scale Toolbox was pop-up.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 3 Scaling The Object

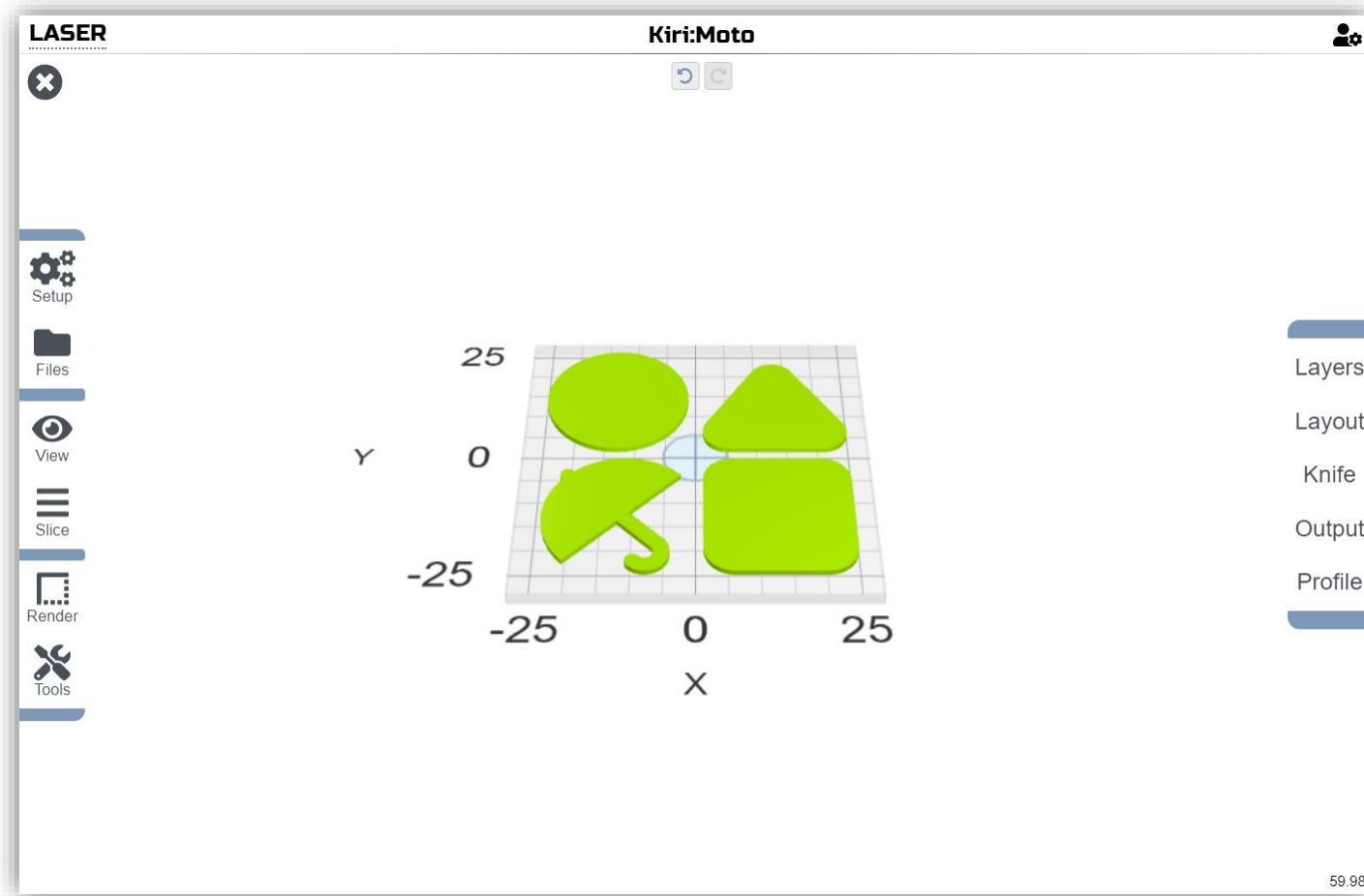
- **Uncheck Z** axis , then change **X** from **42** to **45**, Y axis was adjusted together.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 3 Scaling The Object

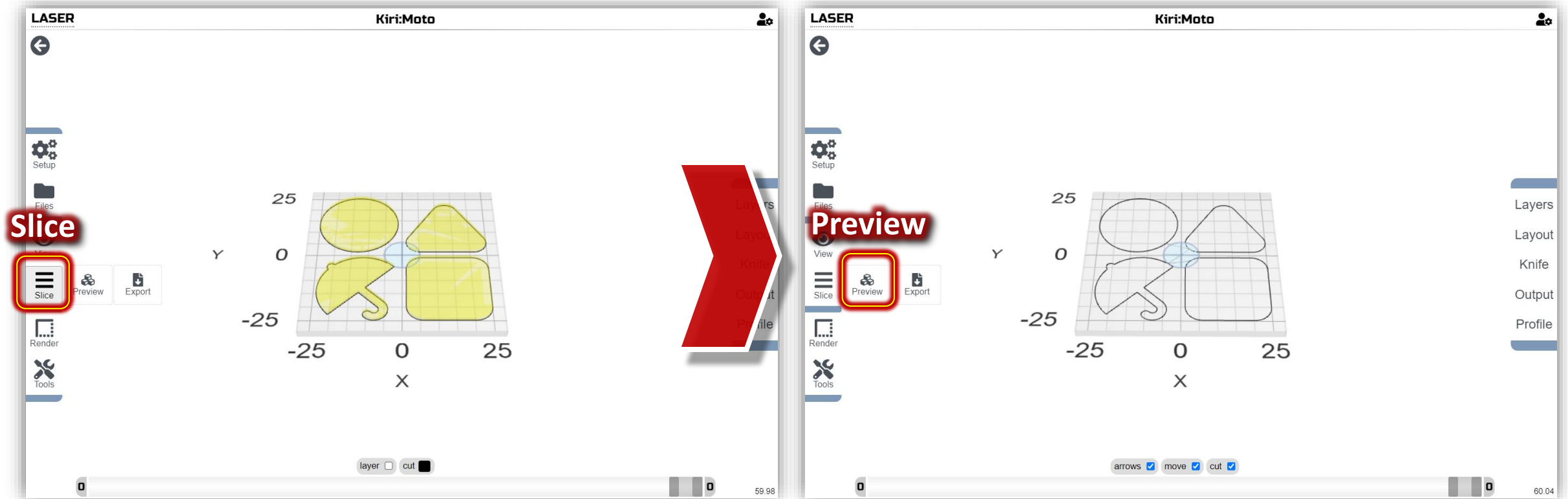
- The object was scaled (45 mm x 45 mm x 1 mm)



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 4 Slice the Object and Preview the “path”

- Refer Exercise A Step 3,4. Slice the object and preview the path after sliced.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 5 Export “path” to G-code

- Refer Exercise A Step 5. Export the path to g code file (Cookies_3D.stl)

The screenshot illustrates the process of exporting a 3D model to G-code. On the left, the Kiri:Moto software interface shows the 'Export' button highlighted in the sidebar. A red arrow points from this button to the 'Export' dialog box. In the dialog, the 'File Name' is 'Cookies_3D-007' and 'Segments' is '273'. The 'Download gcode' button is highlighted. Another red arrow points from this button to the 'Cookies_3D.gcode' file in the download list. A blue box highlights the filename 'Cookies_3D.gcode'.

Export

File Name: Cookies_3D-007
Segments: 273

Download **gcode**

svg dxf gcode

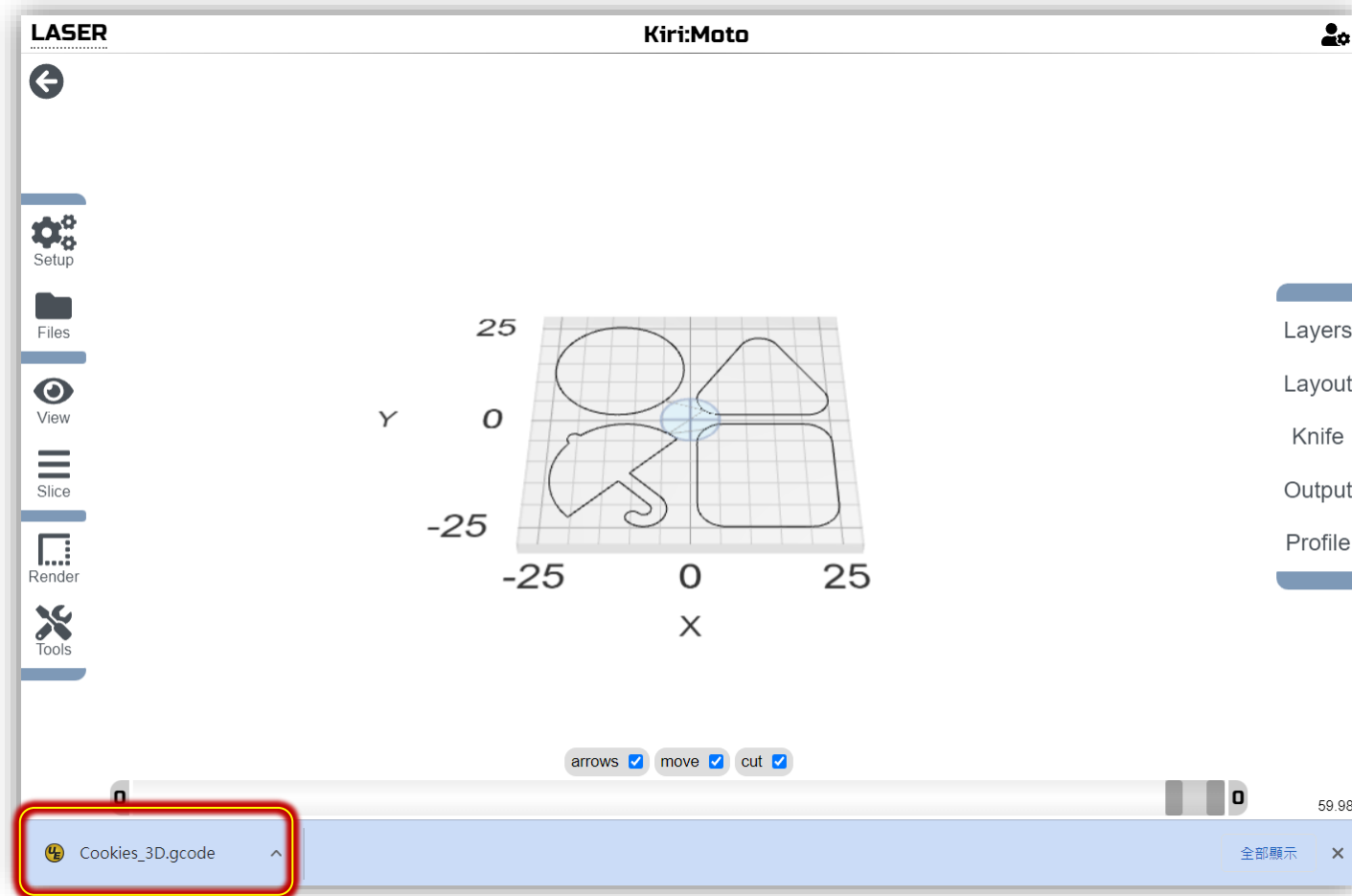
“Cookies_3D.gcode”

MICROCHIP

Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Step 5 Export “path” to G-code

- The G-code file was downloaded.



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

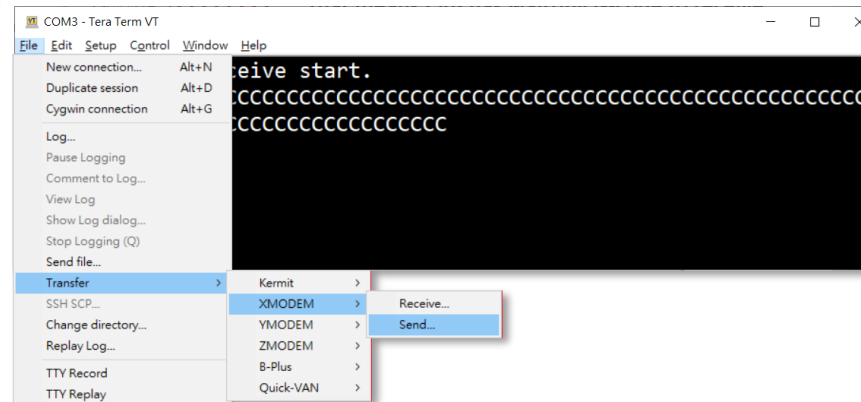
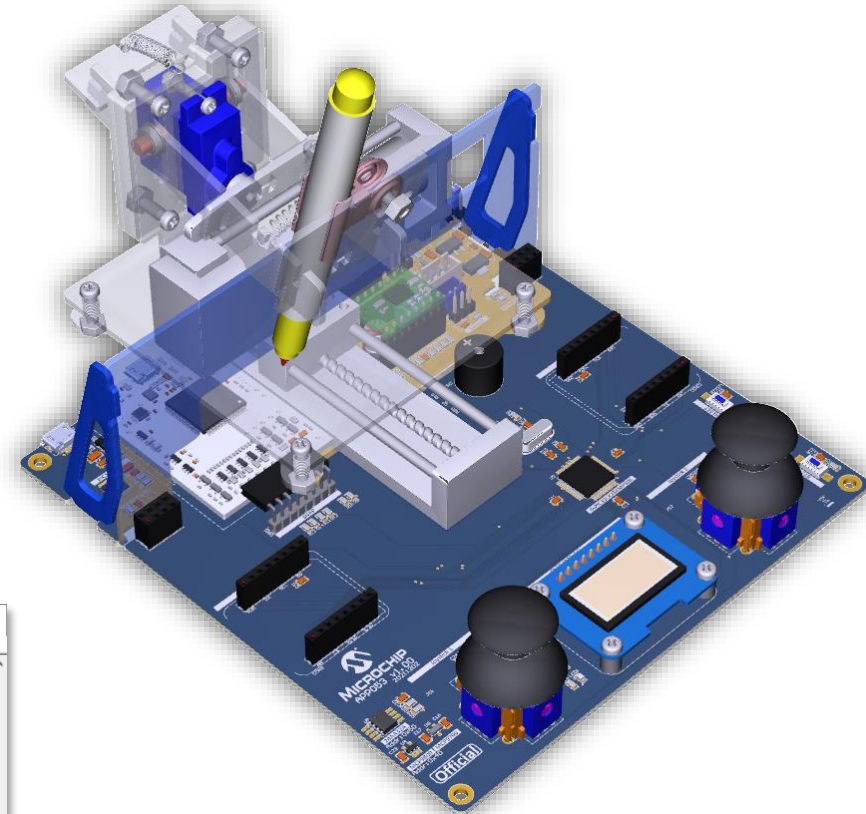
Finally, Draw the G-code

- Send the G-code to through XMODEM & enjoy.

e.g.:

```
M106 S255 \n; Laser Trun On
G1 X23.152 Y36.349 F1000 \n
G1 X24.924 Y38.849 \n
M107 \n
G0 X29.816 Y19.279 \n
M106 S255 \n
G1 X29.932 Y19.288 F1000 \n
M107 \n
...
```

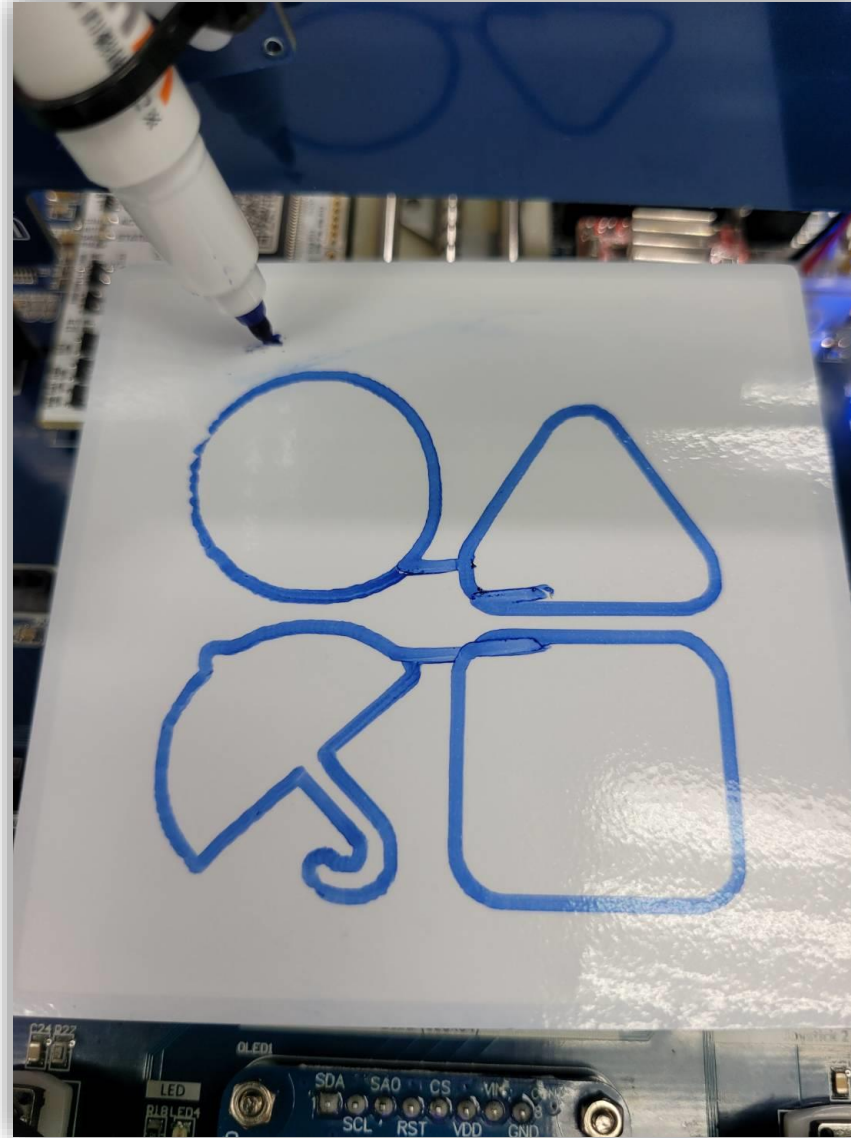
Cookies_3D.gcode



Exercise B Generate a G-code file at slice tool directly by import 3D Object (.stl)

Finally, Draw the G-code

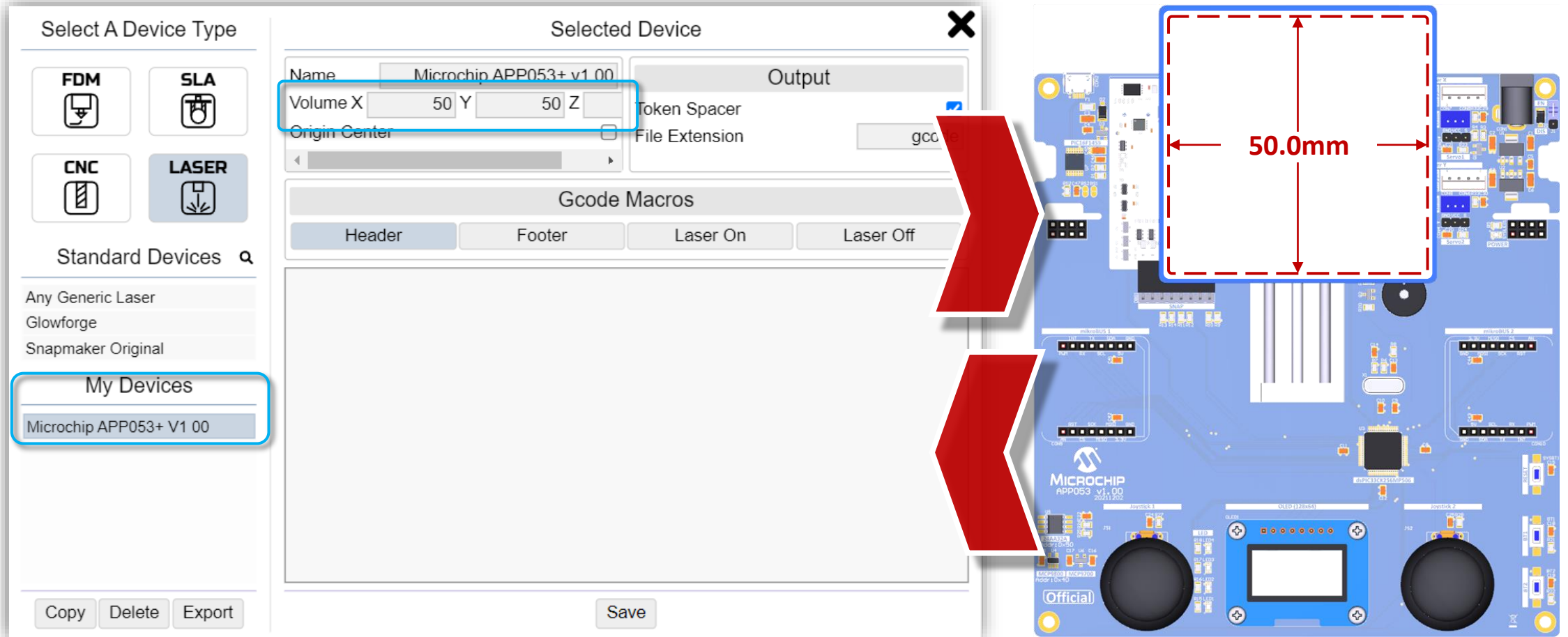
- Result



About Working Panel

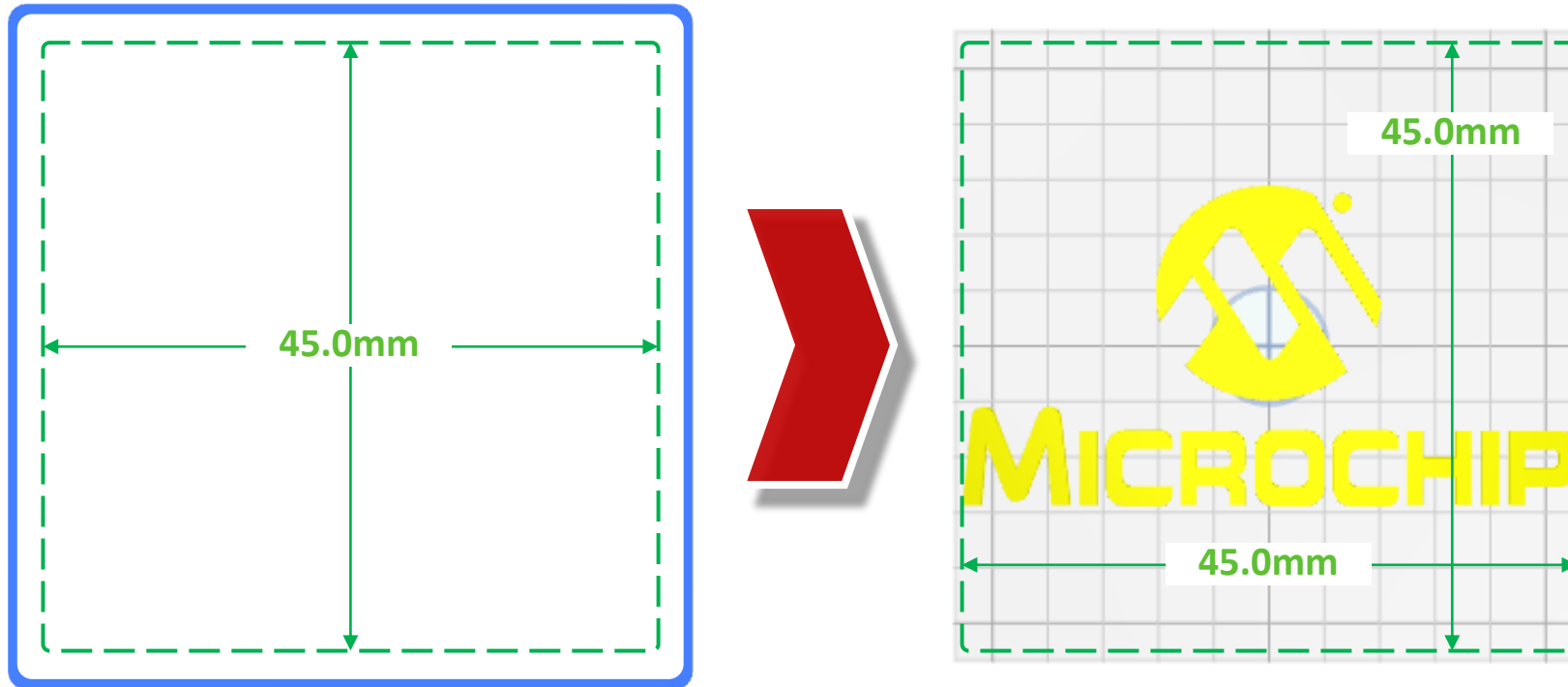
Absolute Maximum Dimension

- Absolute Maximum Dimension is 50mm x 50mm



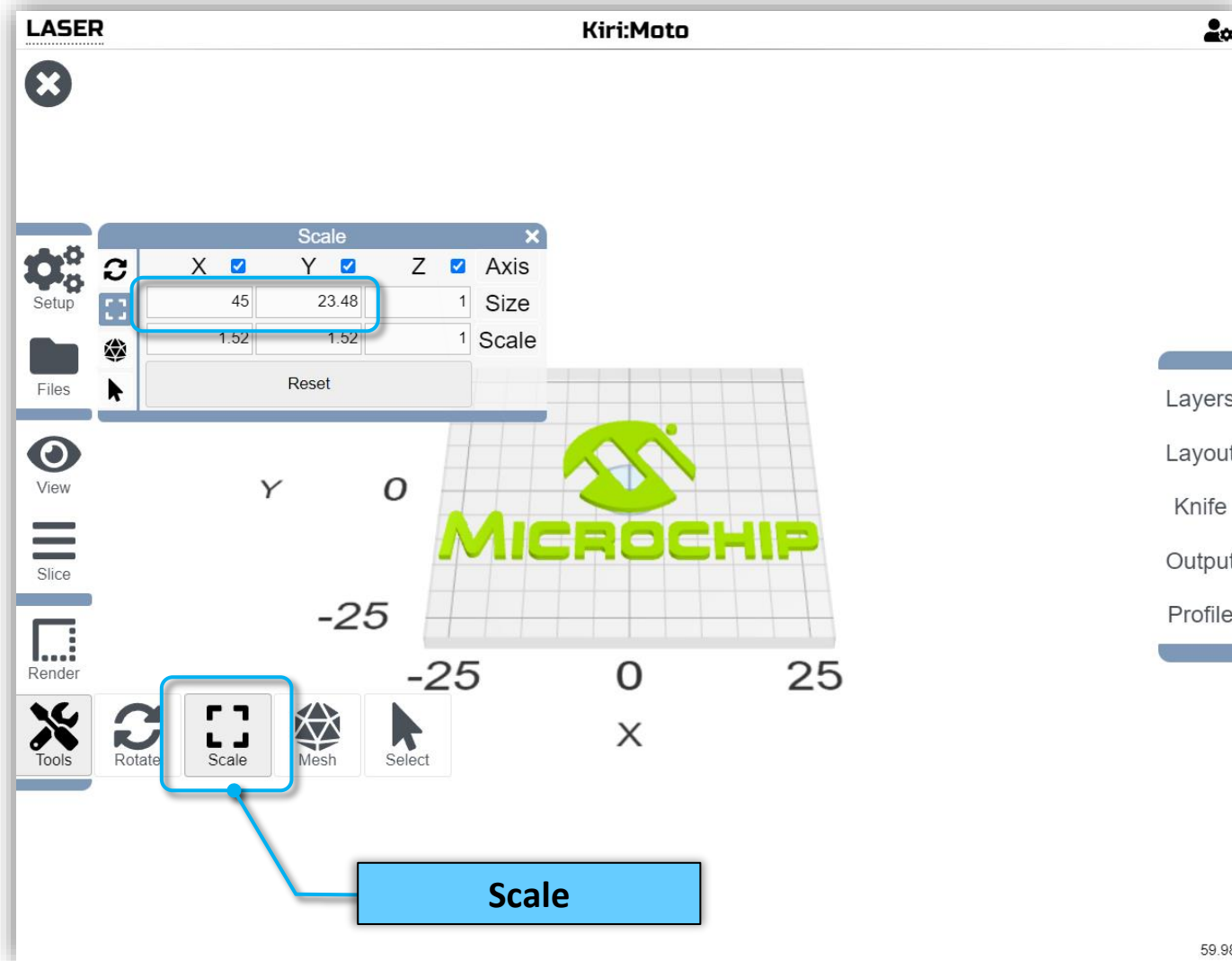
Workable Maximum Dimension

- Workable Maximum Dimension is 45.0mm x 45.0mm



Scaling at Kiri:Moto

- Maximum Height and length must < 45.0(mm)



Online 3D Modeling Tool TINKERCAD Introduction

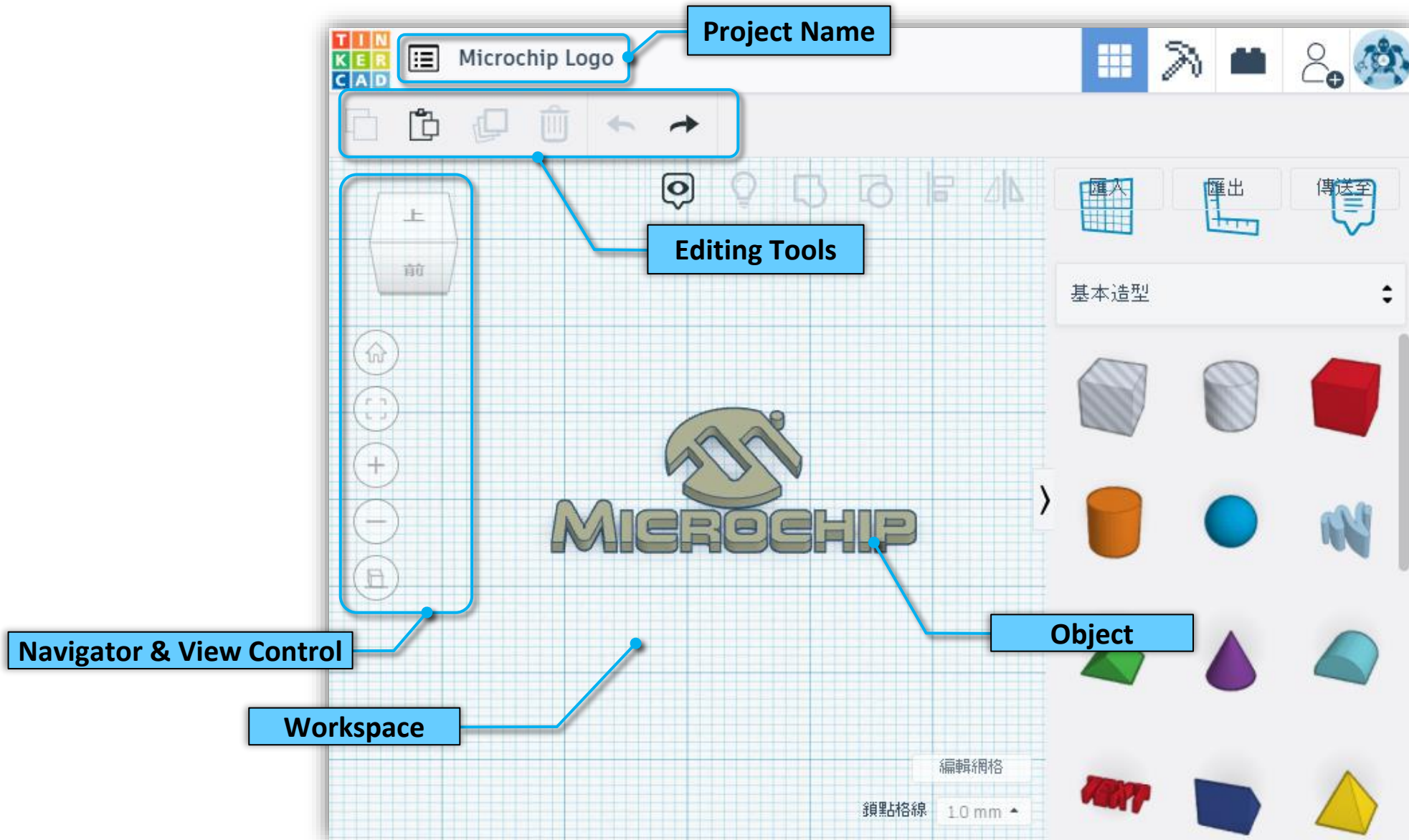
Online 3D Modeling Tool

TINKERCAD - Online 3D Modeling Tool

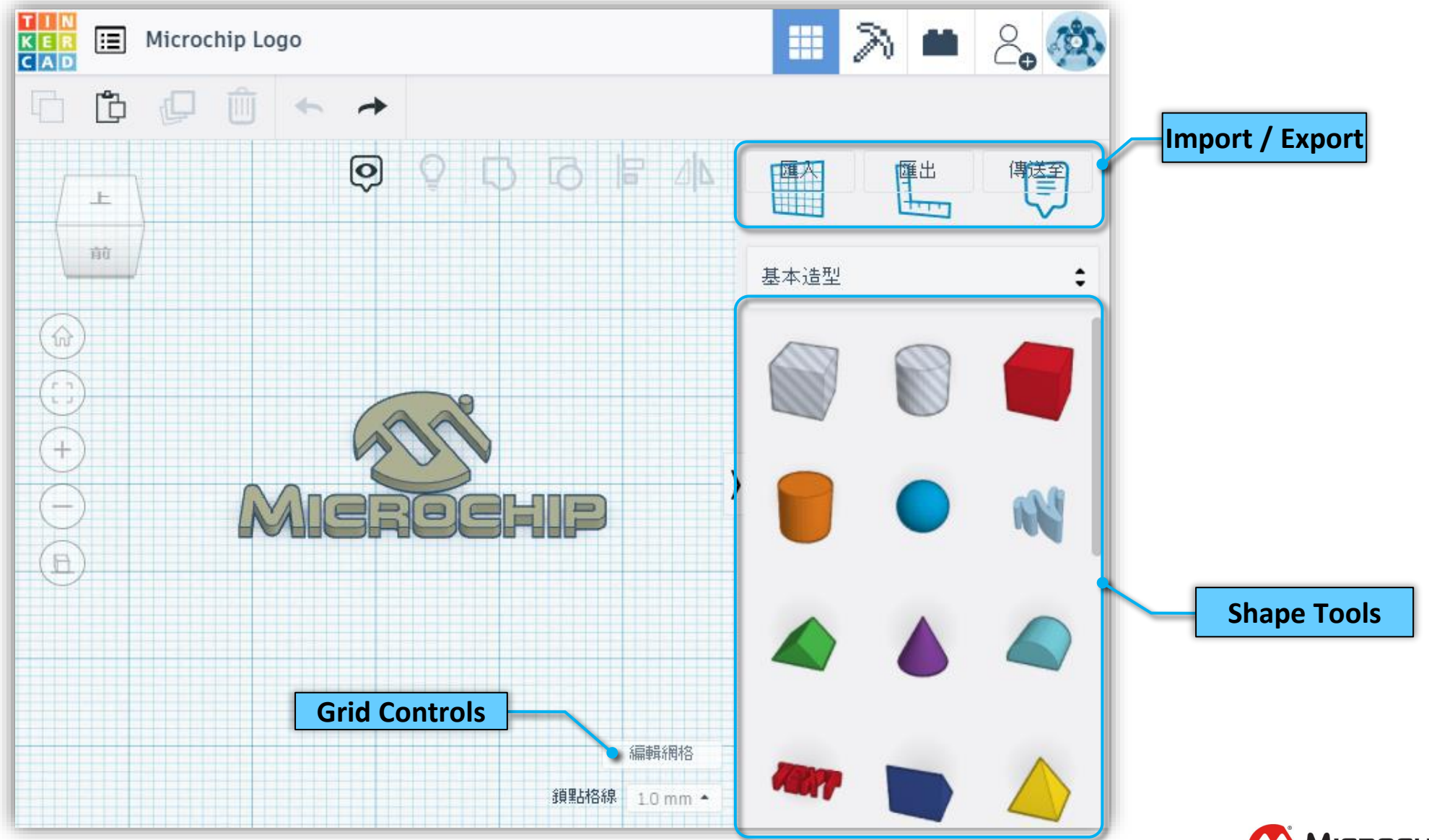
- TINKERCAD (www.tinkercad.com) is a Browser-Based 3D design and modeling program created to provide a way for a variety of users (beginners to experts) to create projects.
 - You must register an account before getting started.
 - TINKERCAD include below features,
 - It's Browser-Based.
 - Easy to use for beginner.
 - Support .stl and .svg import and .stl export.
 - You can find a lot on-line training and courses. e.g.,
 - <https://www.tinkercad.com/learn/designs/learning>
 - <https://libguides.uthscsa.edu/c.php?g=832157&p=8696241>
 - <https://maker.pro/custom/tutorial/introduction-to-tinkercad>
 - <https://www.youtube.com/playlist?list=PL59TW6BsbMMXO2WKzwkTsJqfgxeop6M1u>
- etc..



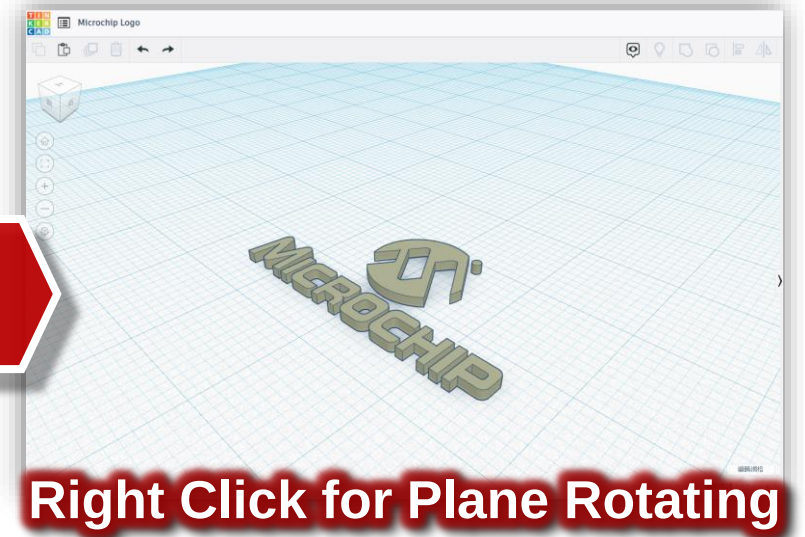
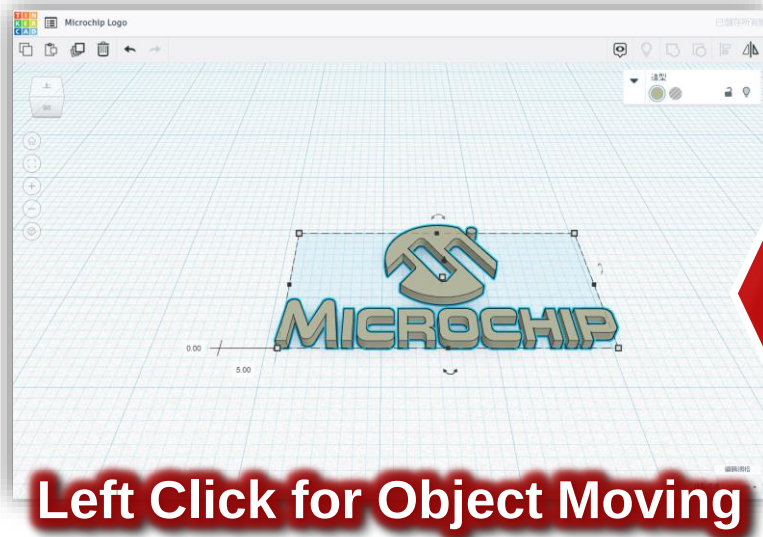
TINKERCAD Introduction *cont.1*



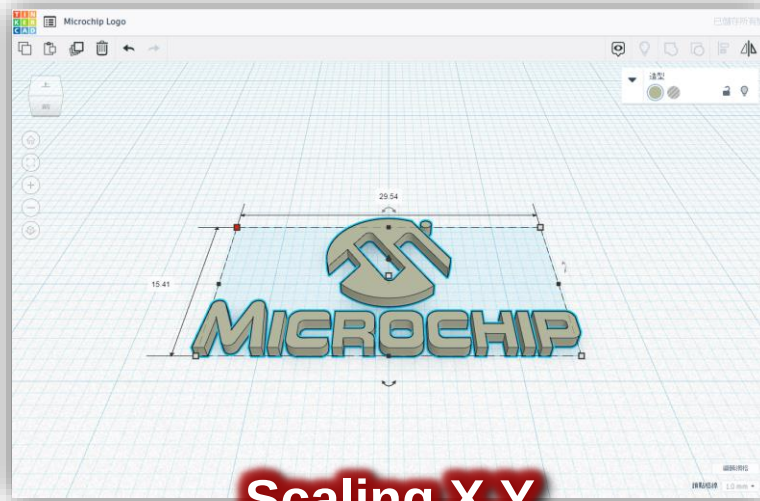
TINKERCAD Introduction *cont.2*



TINKERCAD Introduction *cont.3*



TINKERCAD Introduction *cont.4*



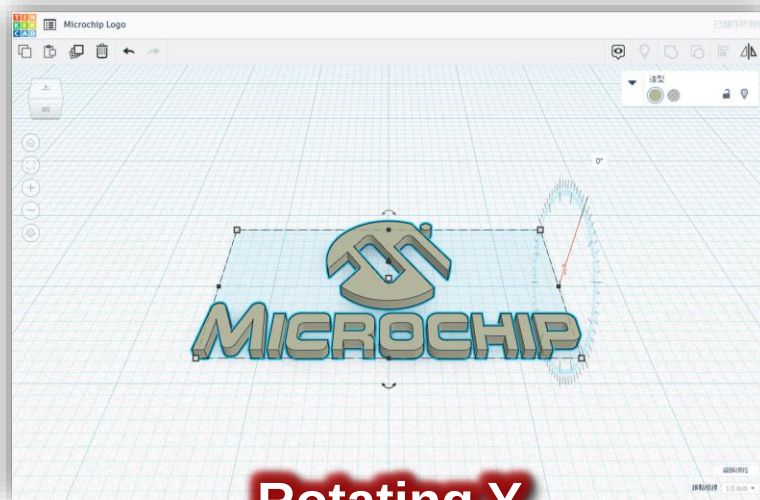
Scaling X,Y



Scaling Z



Standing (Z Offset)



Rotating Y



Rotating X

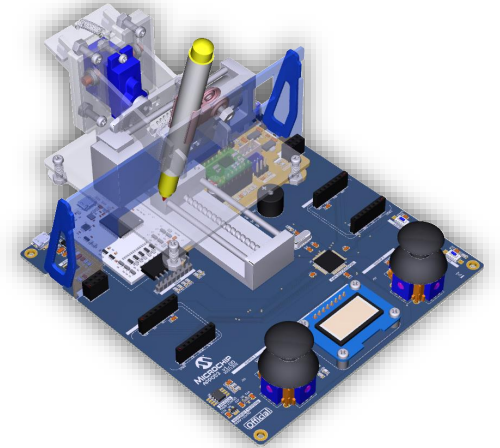


Rotating Z

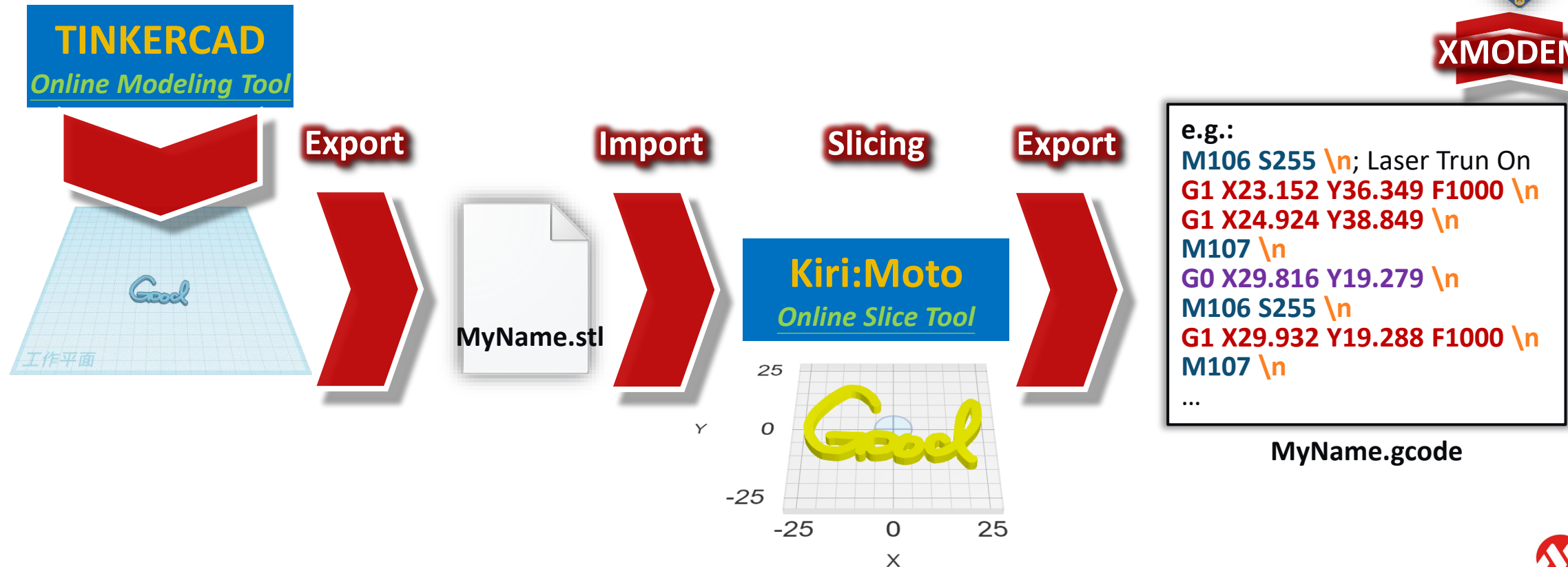
Exercise C

Create a 3D object and
Generate the G-code file at slice tool.

Exercise C Create a 3D object and Generate the G-code file at slice tool.



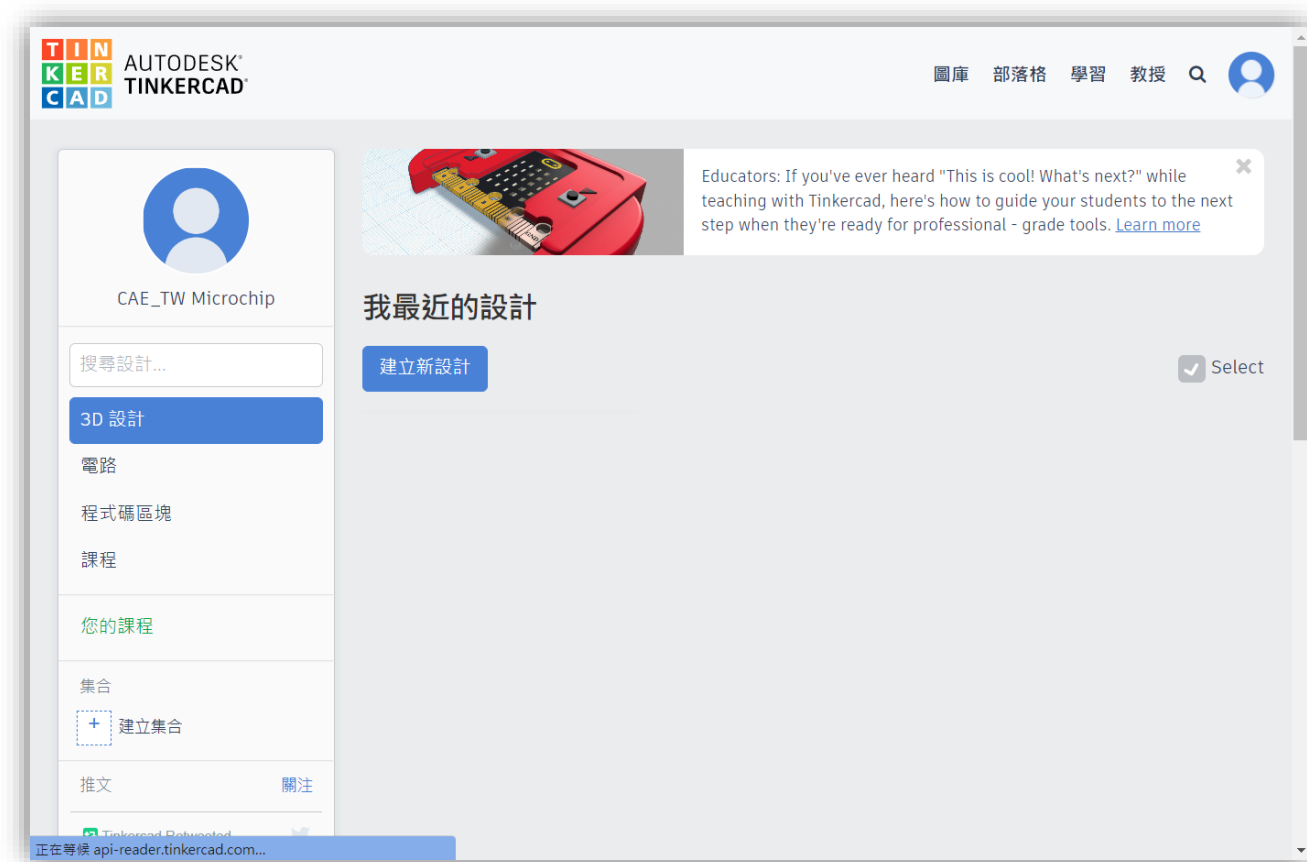
XMODEM



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Getting Started

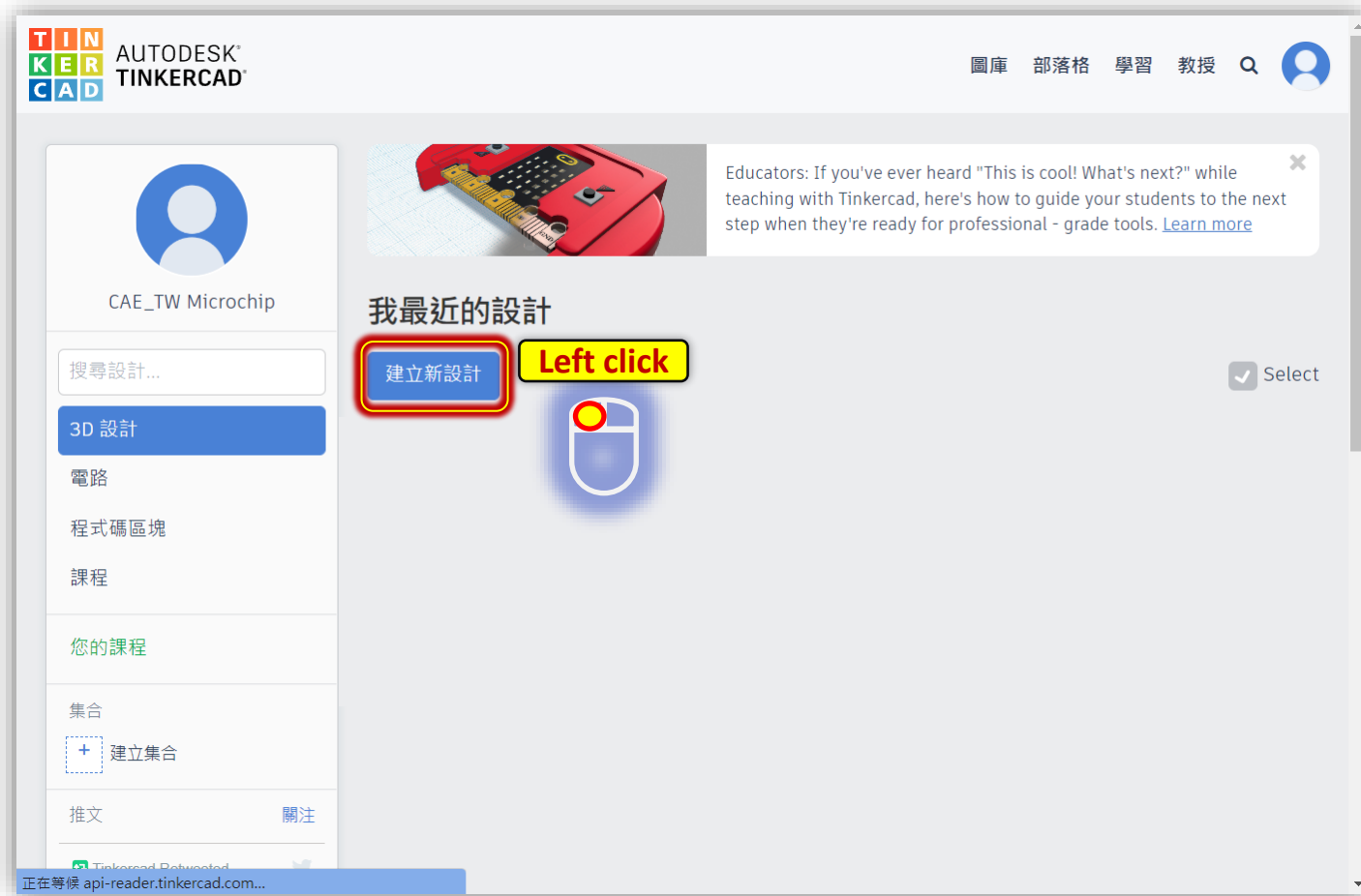
- Go to TINKERCAD website (www.tinkercad.com).
if you haven't account now. Please refer Appendix "[THINKCAD Sign Up](#)"



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Getting Started

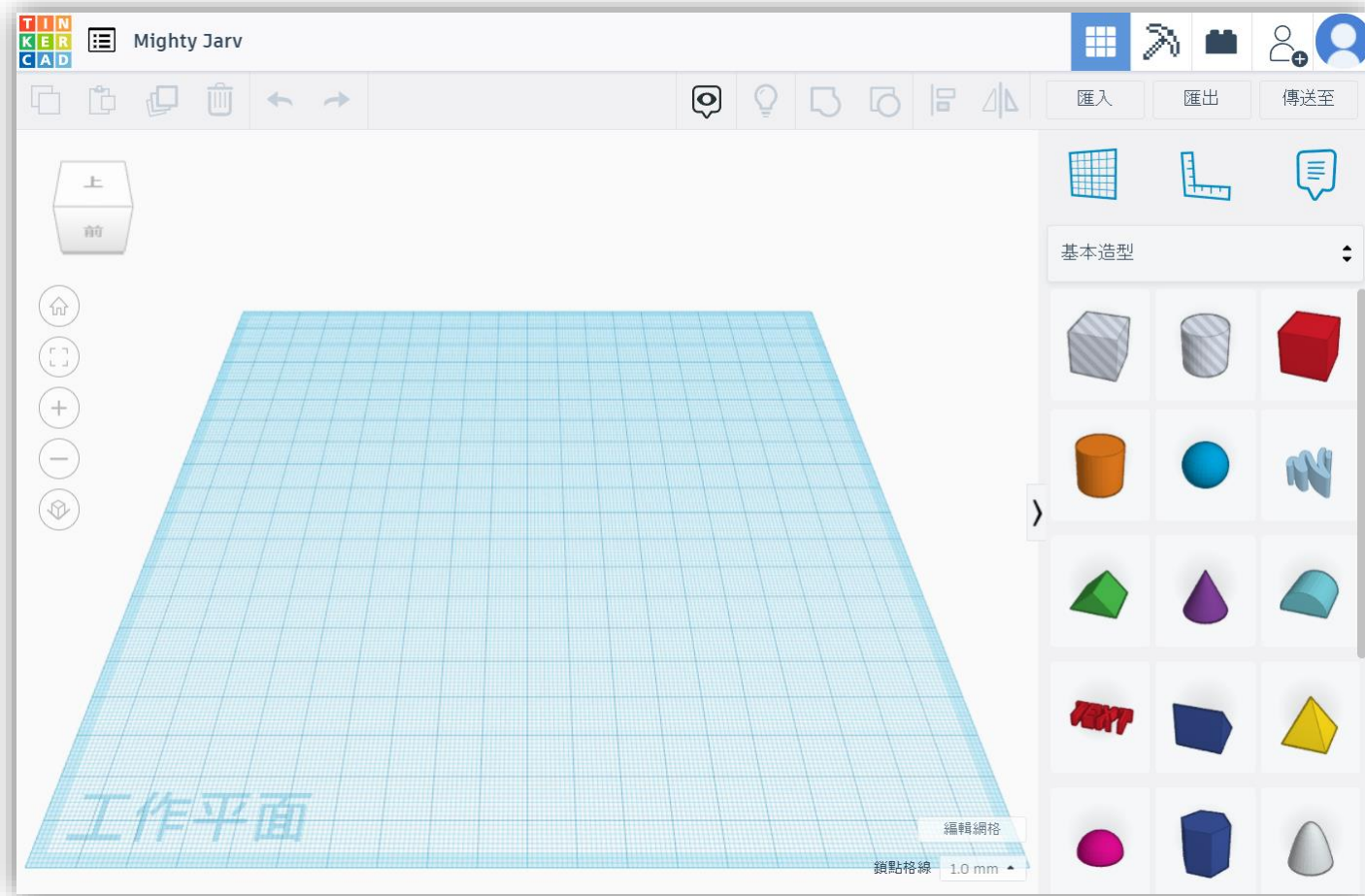
- Click “**建立新設計**” for new design.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Getting Started

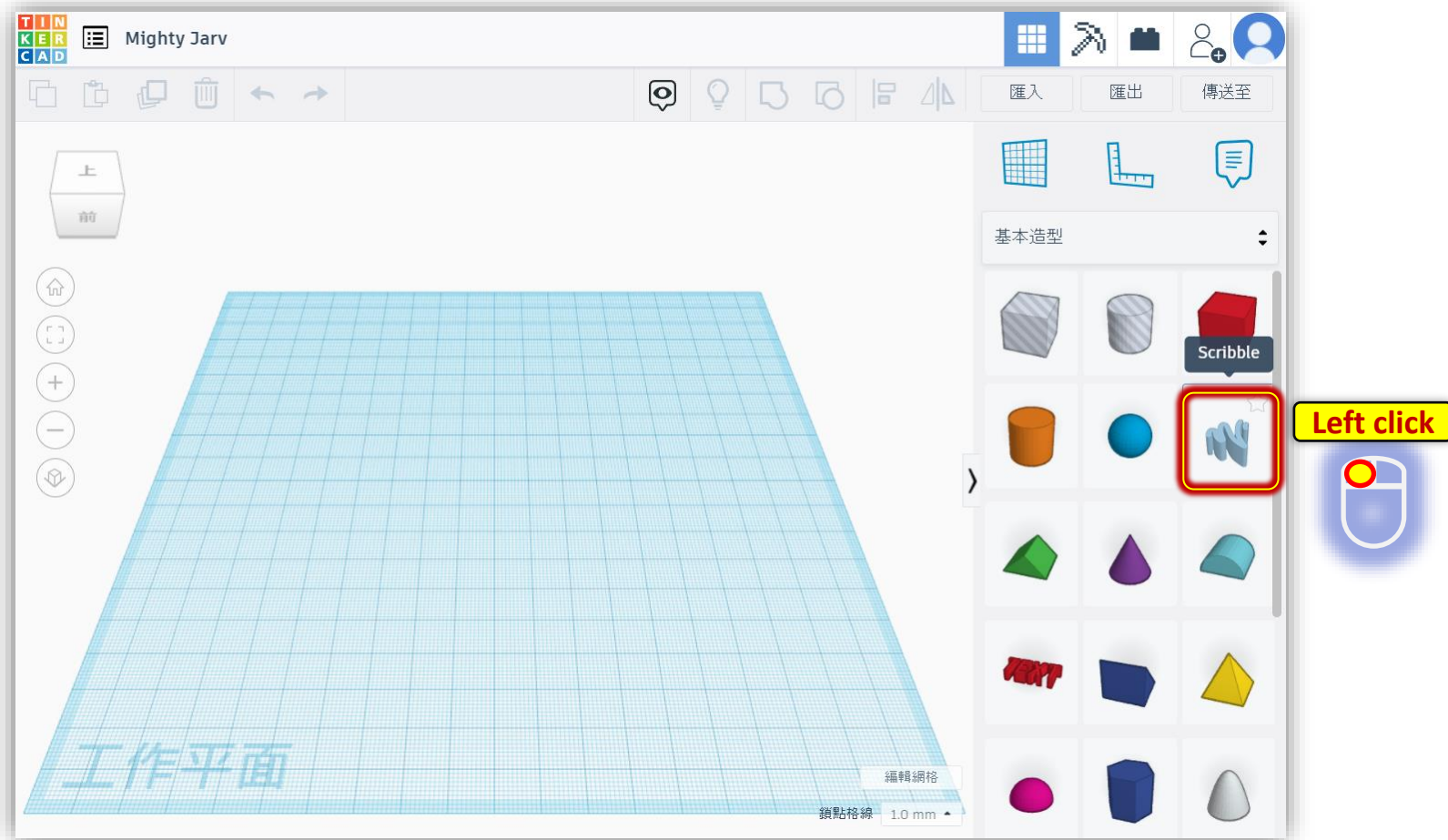
- Go to design panel of new workspace.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

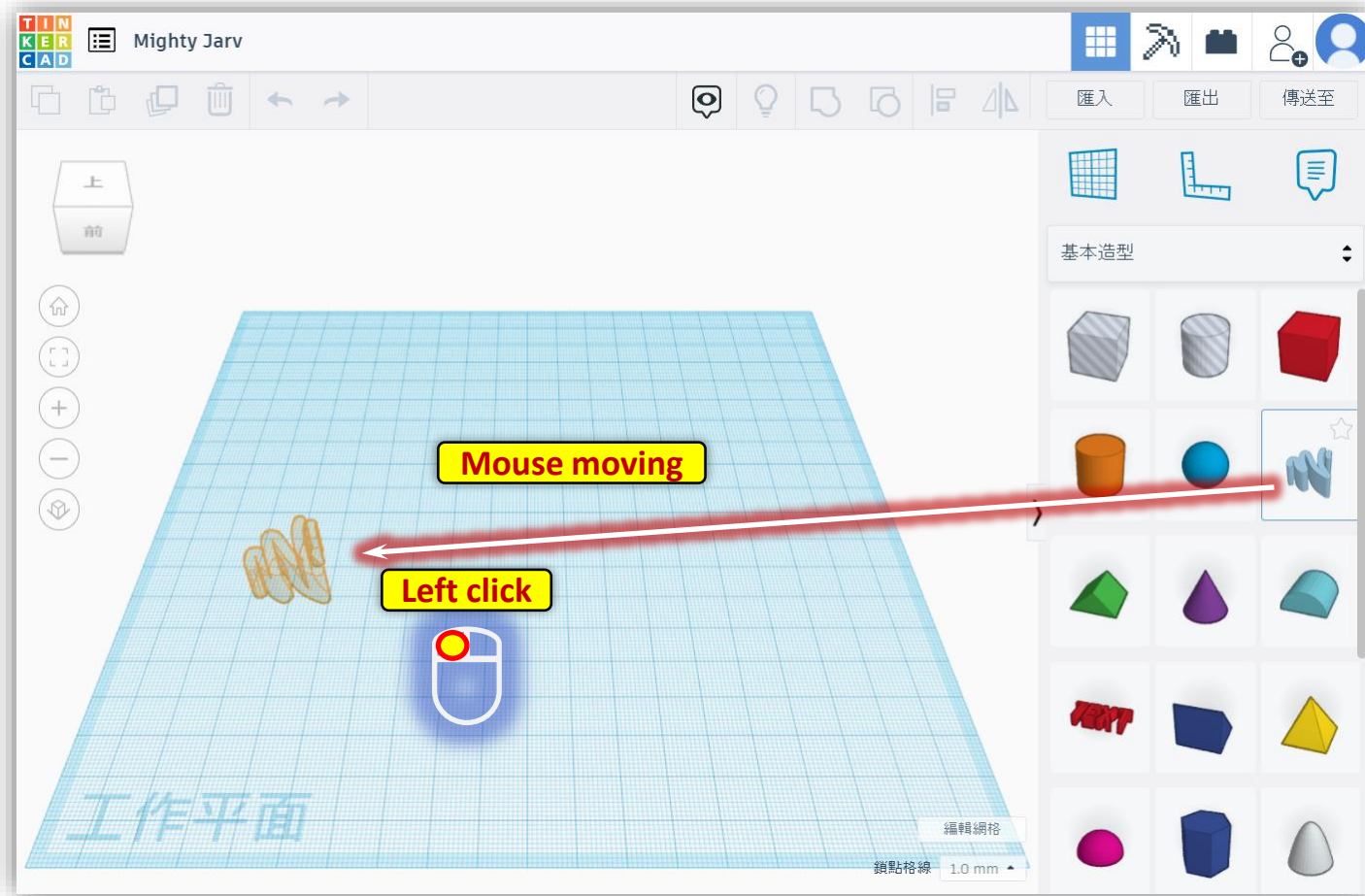
- Select “**scribble**” shape tool



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

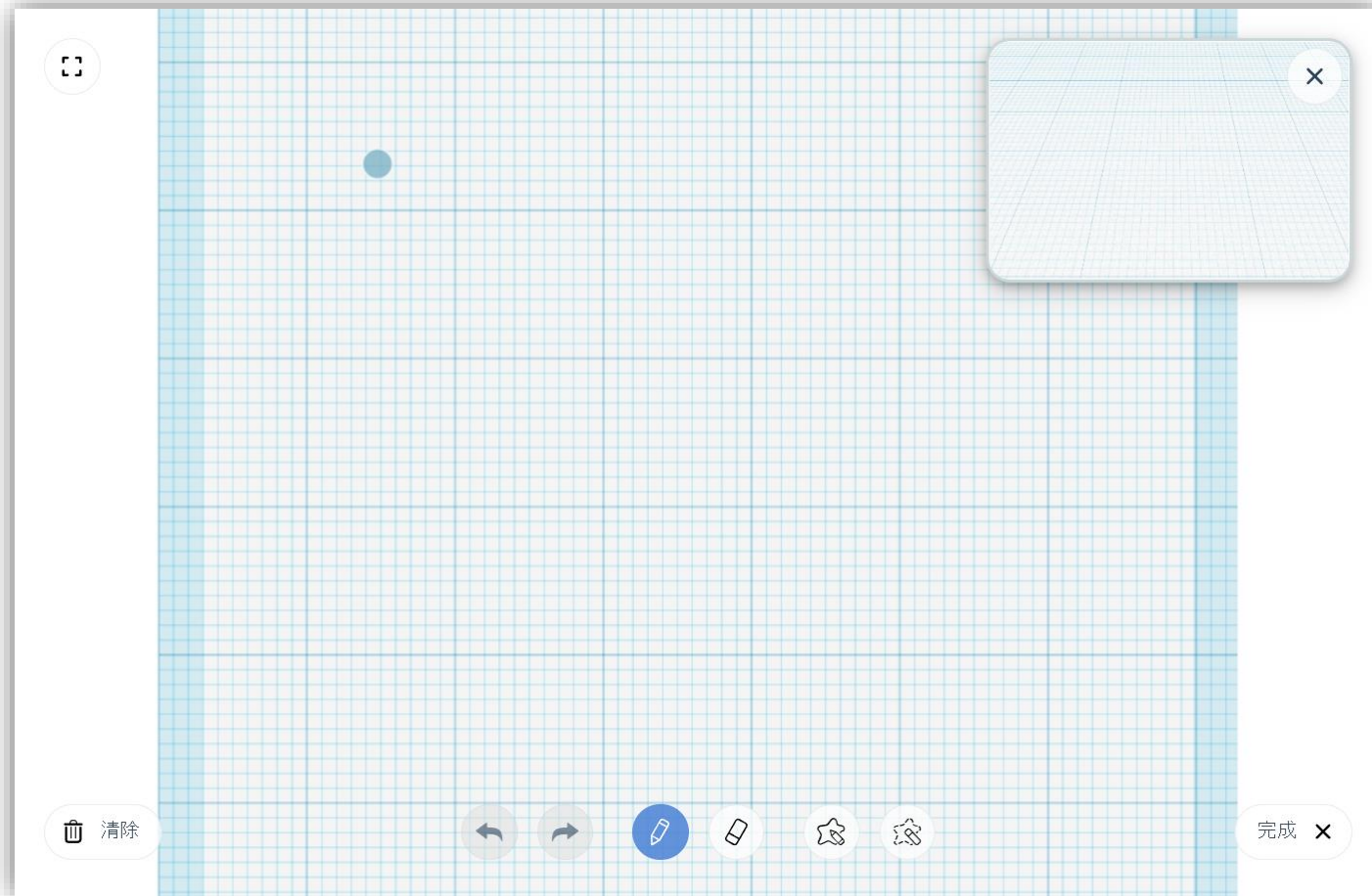
- Move to work panel and click left key of mouse



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

- Go to edit panel of scribble



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

- Draw your name by mouse



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

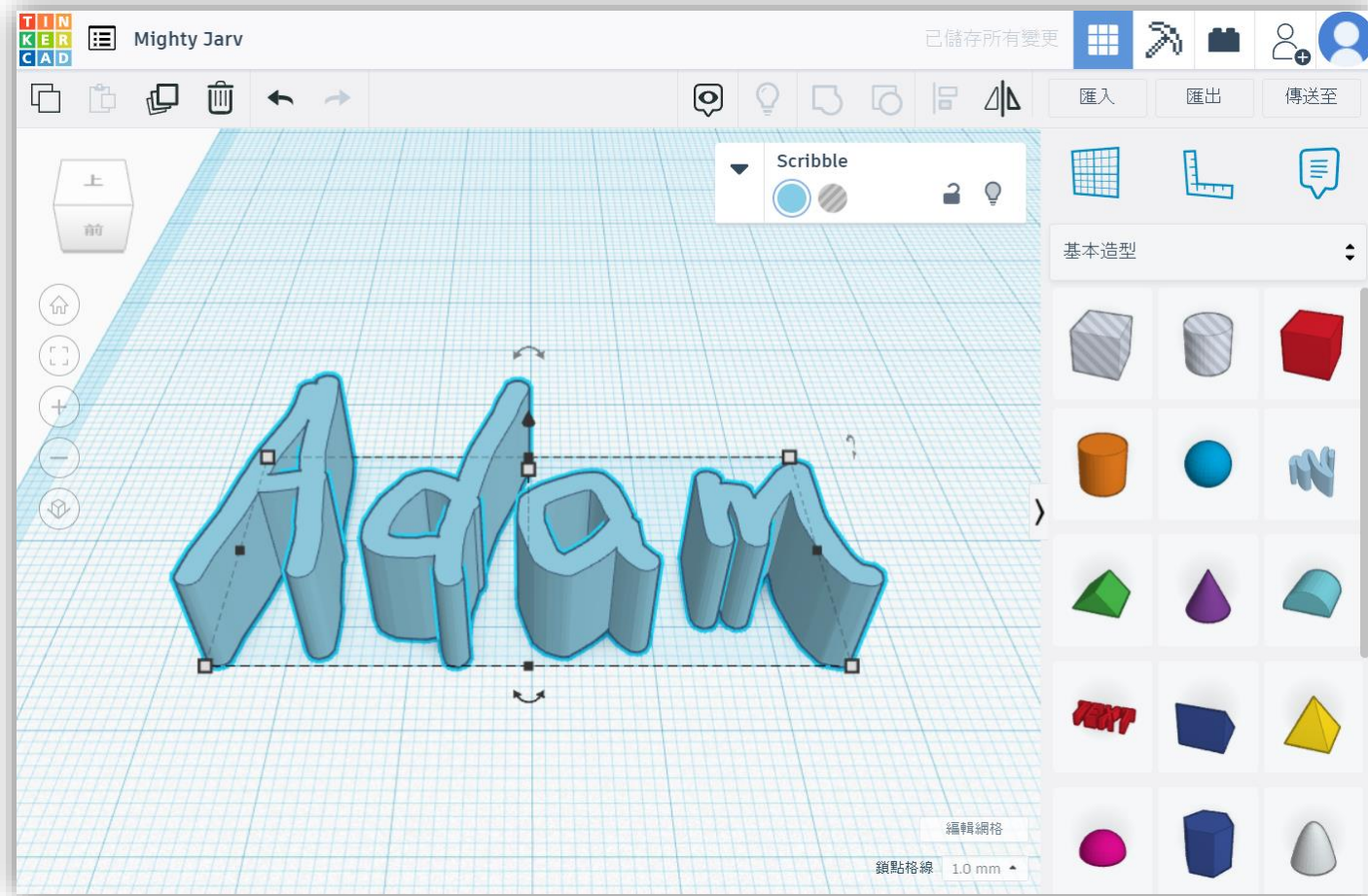
- Select “完成” after edited.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 1 Create your 3D object by scribble tool

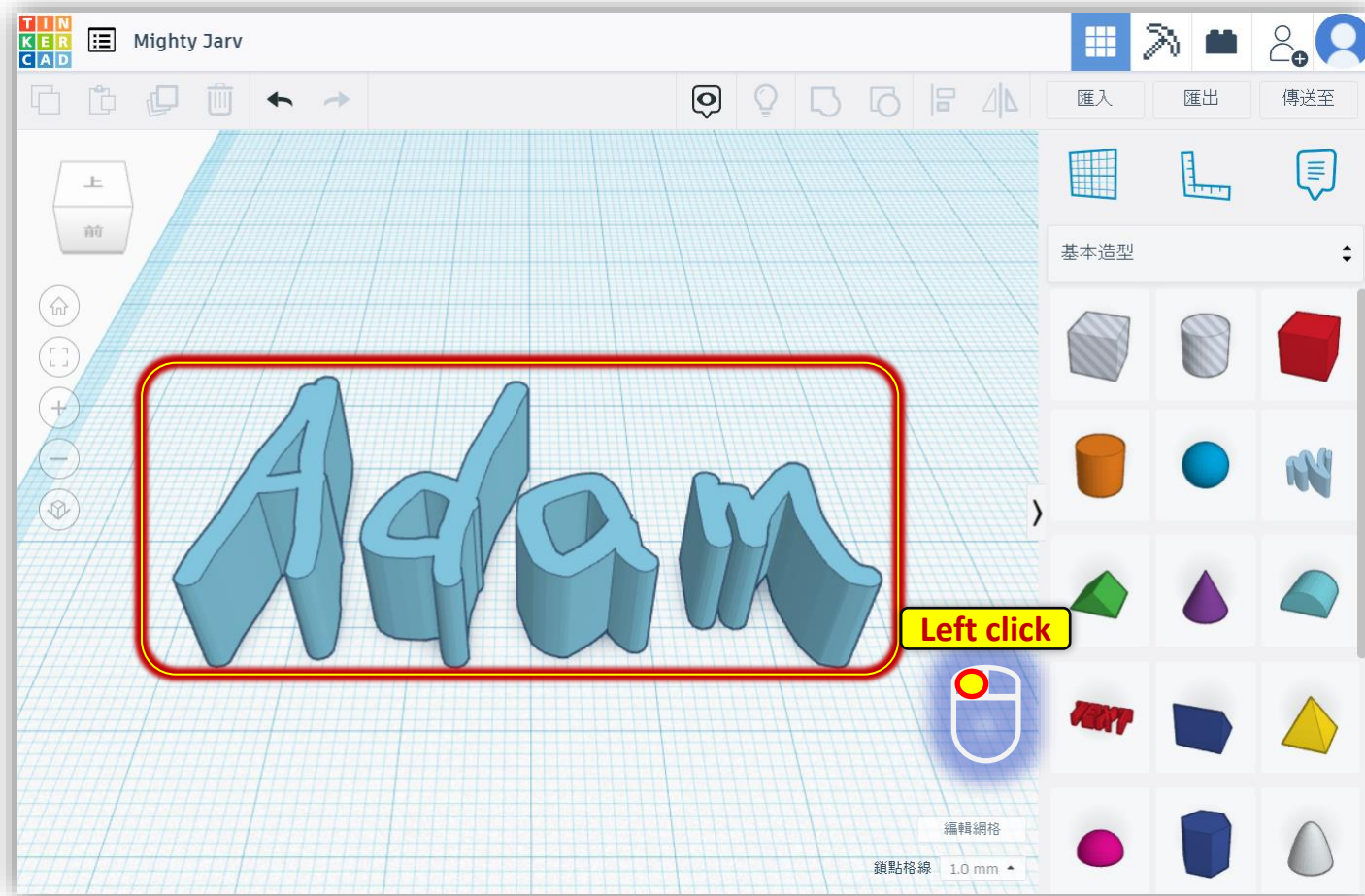
- The 3D object of your named was created.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 2 Scaling Length and Width of the 3D Object

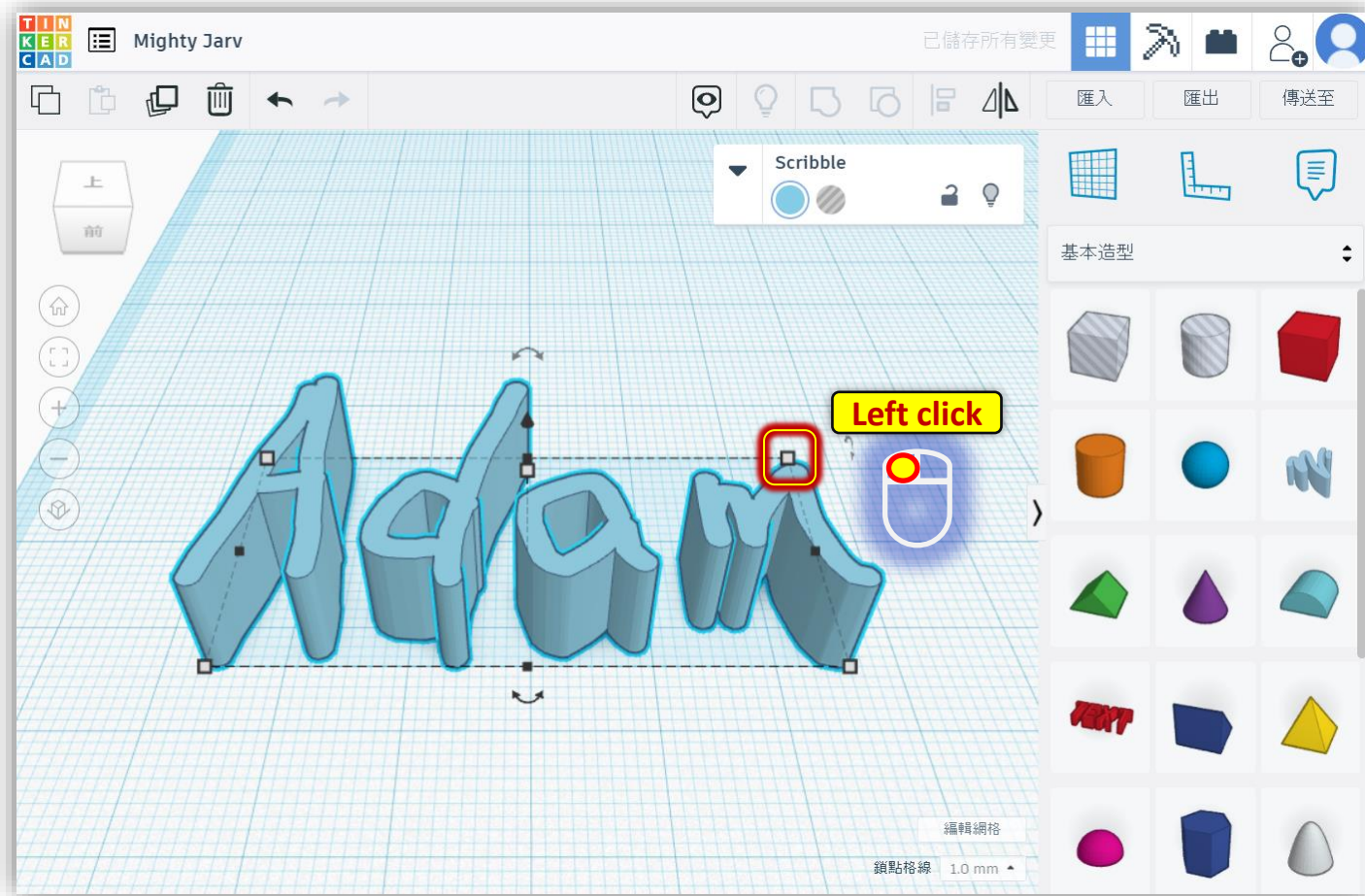
- Click your 3D object.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 2 Scaling Length and Width of the 3D Object

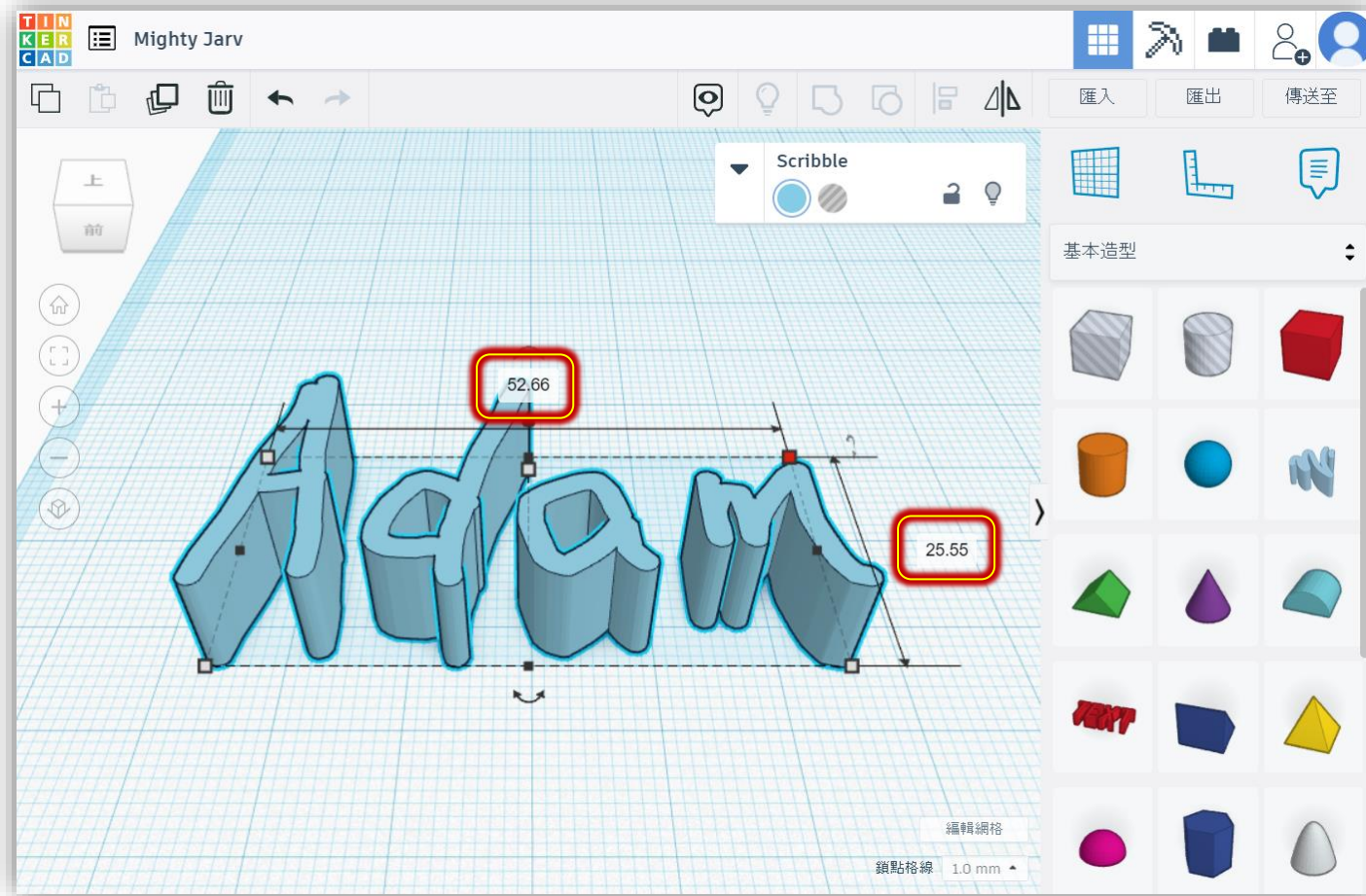
- Click the rectangle on the corner.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 2 Scaling Length and Width of the 3D Object

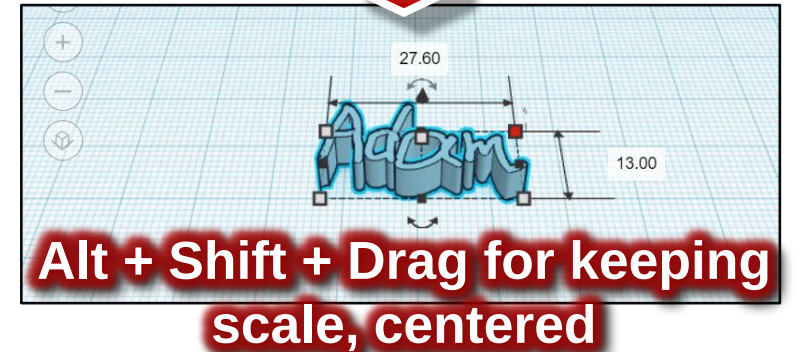
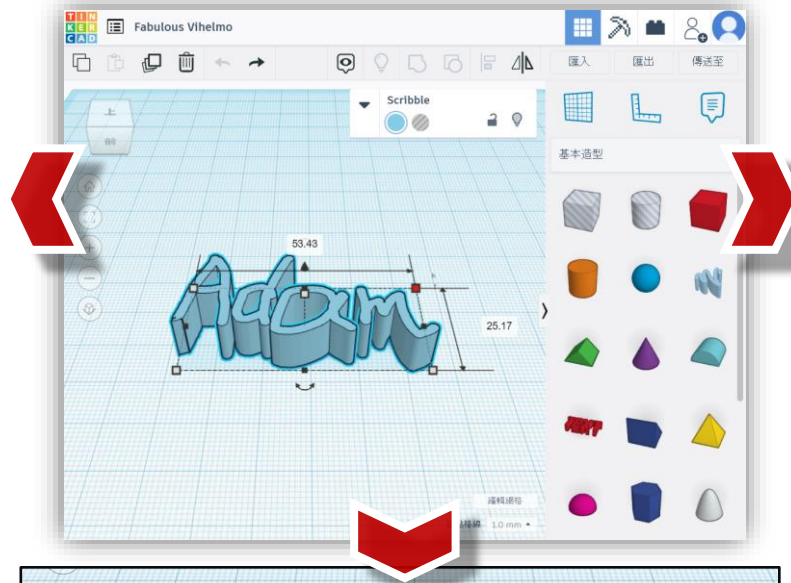
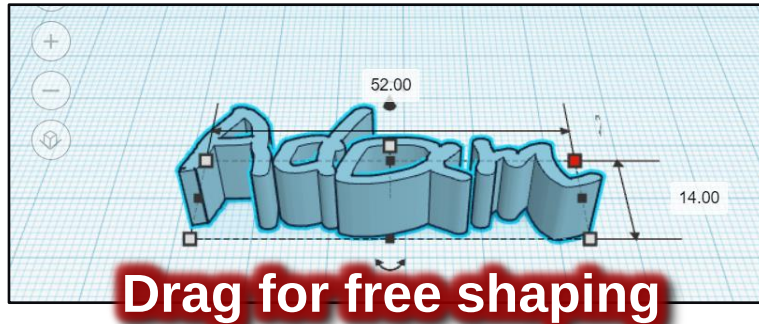
- You can find the dimension (X,Y) of object



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 2 Scaling Length and Width of the 3D Object

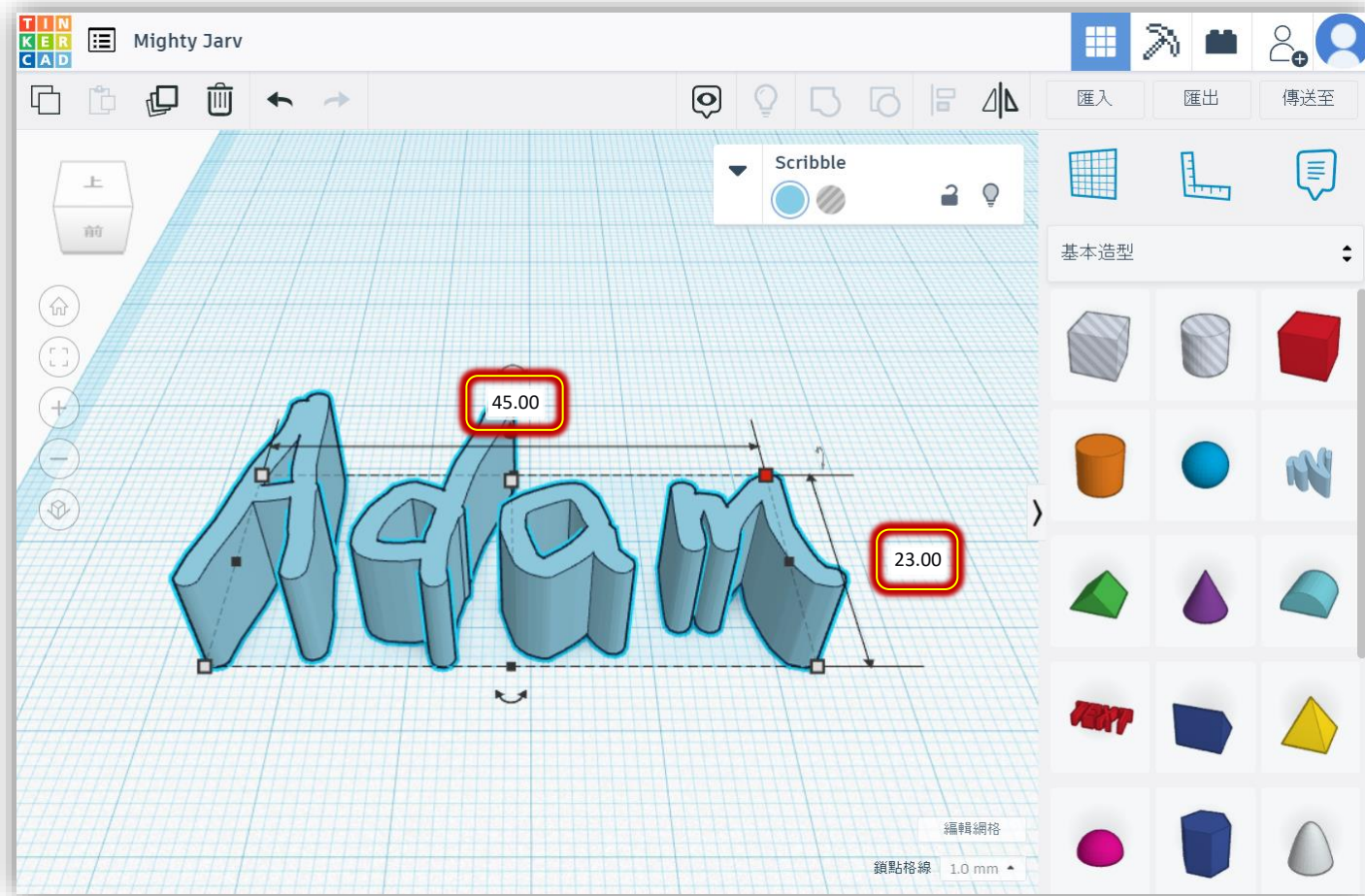
- Please scaling the object to fit Workable Maximum Dimension (**45.0mm x 45.0mm**)



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 2 Scaling Length and Width of the 3D Object

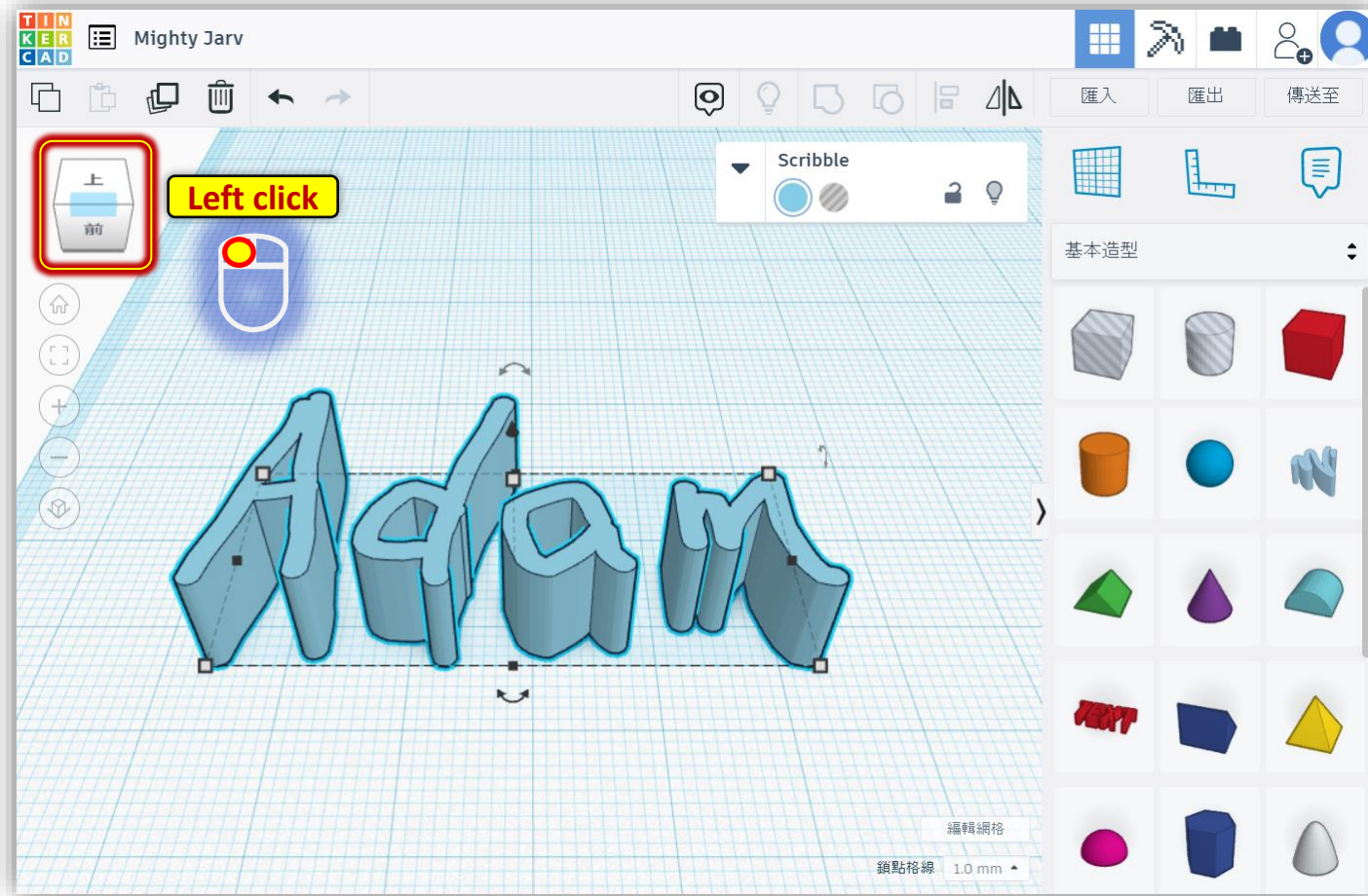
- Object was scaled. (e.g., 45.0 mm x 23.0 mm)



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 3 Adjust Height of the 3D Object

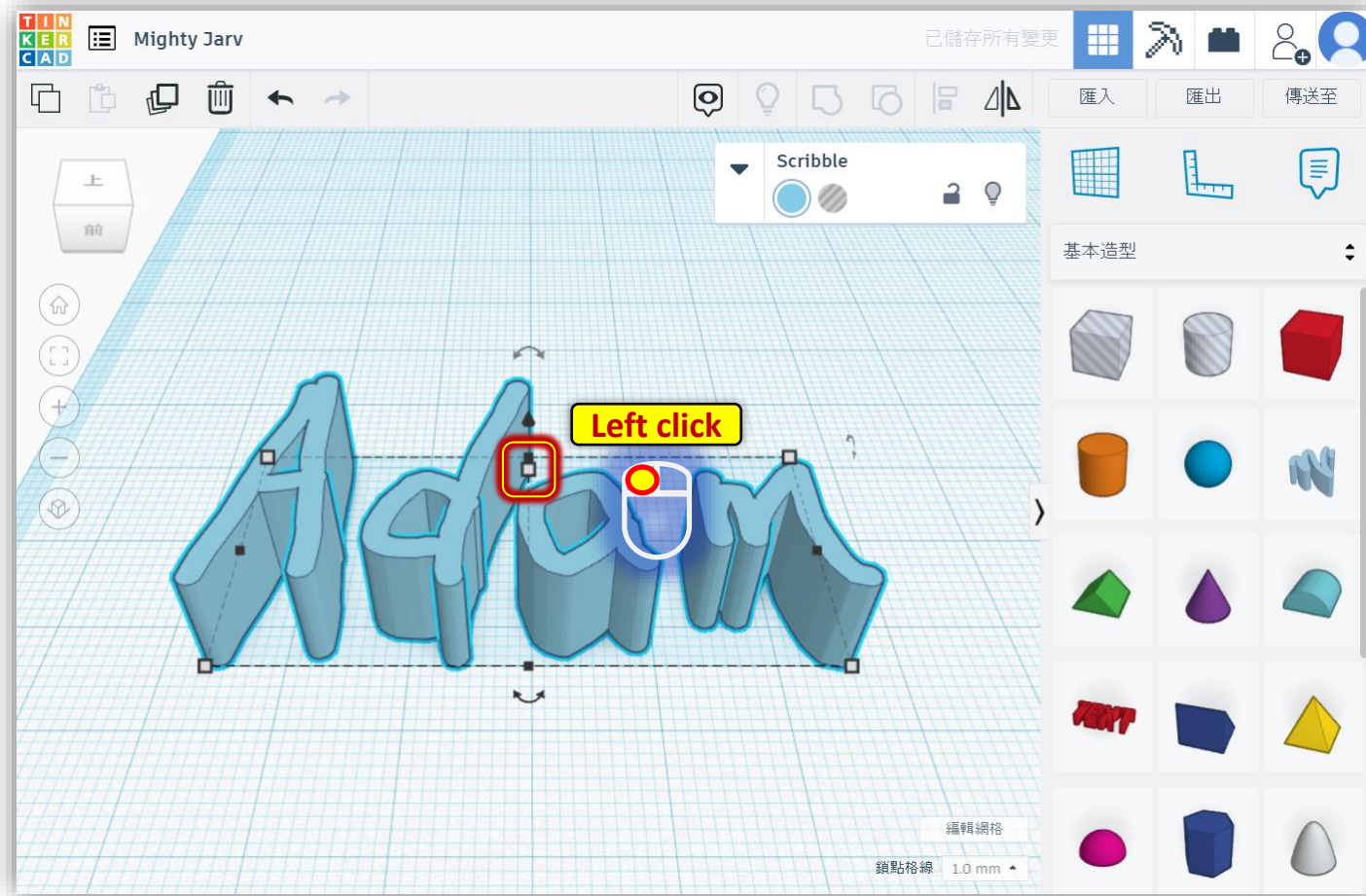
- Click view cube to change View angle firstly.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 3 Adjust Height of the 3D Object

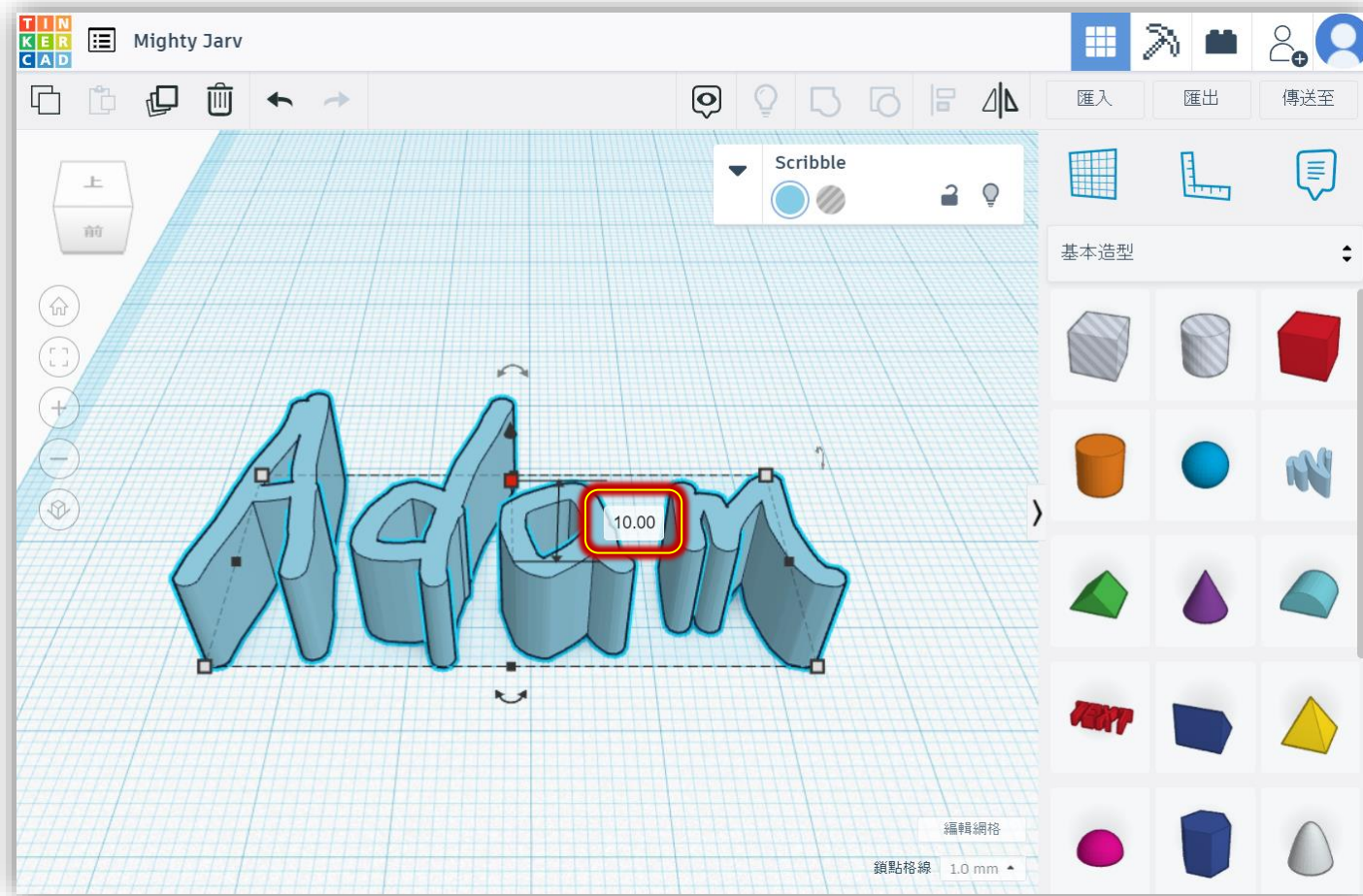
- Click the rectangle on the top.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 3 Adjust Height of the 3D Object

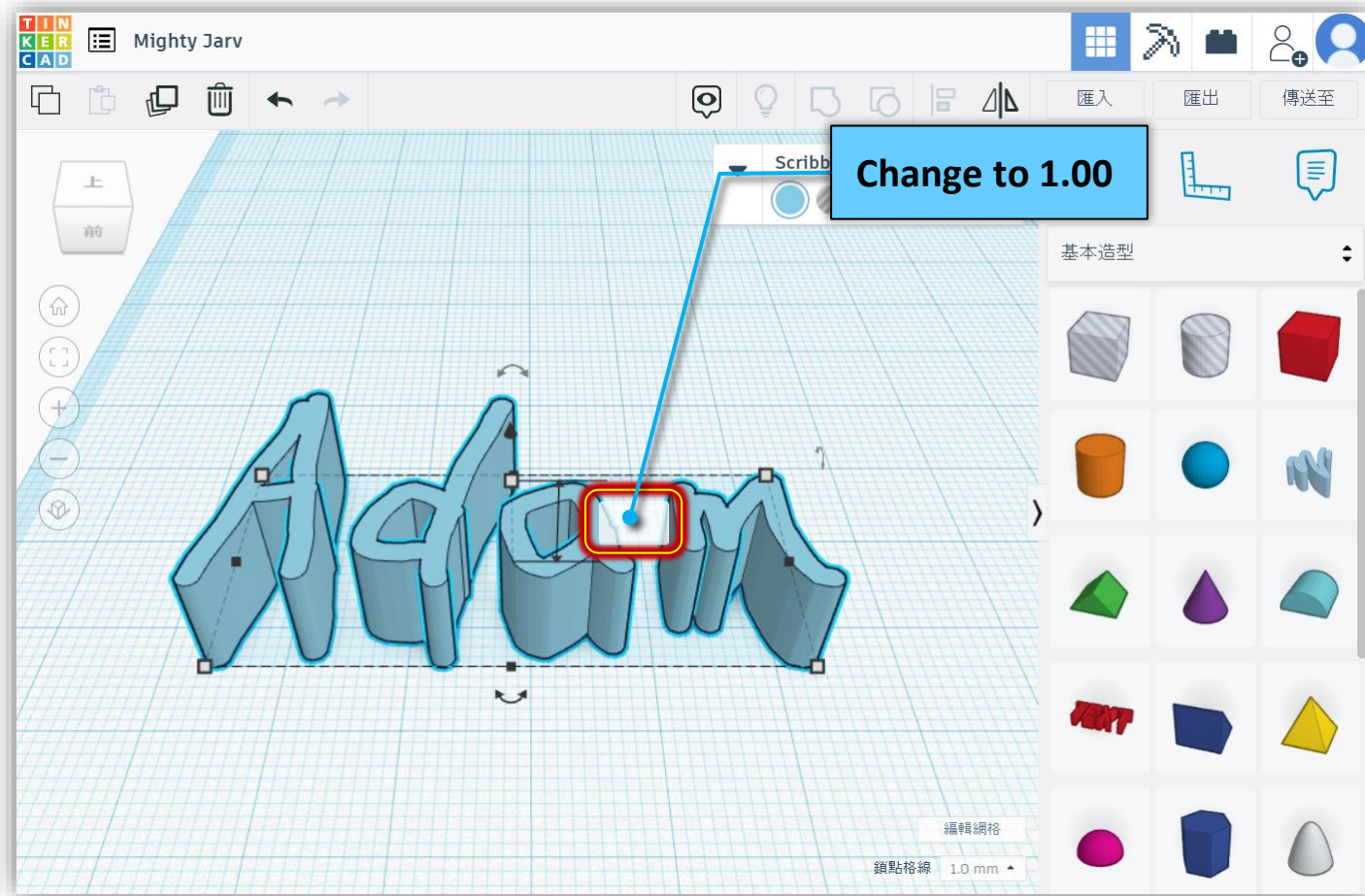
- You can find the dimension (Z) of object



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 3 Adjust Height of the 3D Object

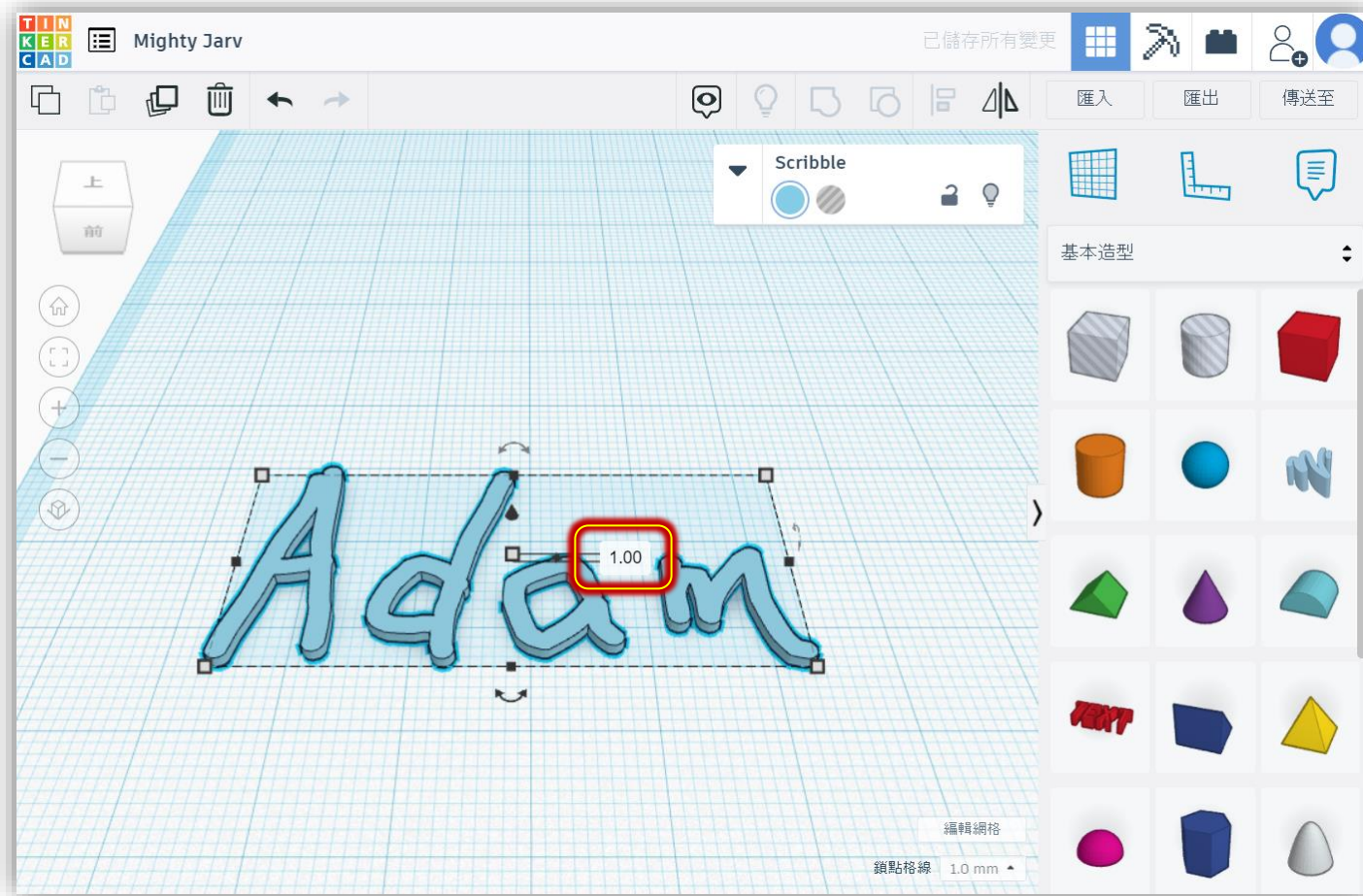
- Change the height to 1.00 (mm)



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 3 Adjust Height of the 3D Object

- Height of object was changed to 1.00 (mm)



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 4 Change Project Name

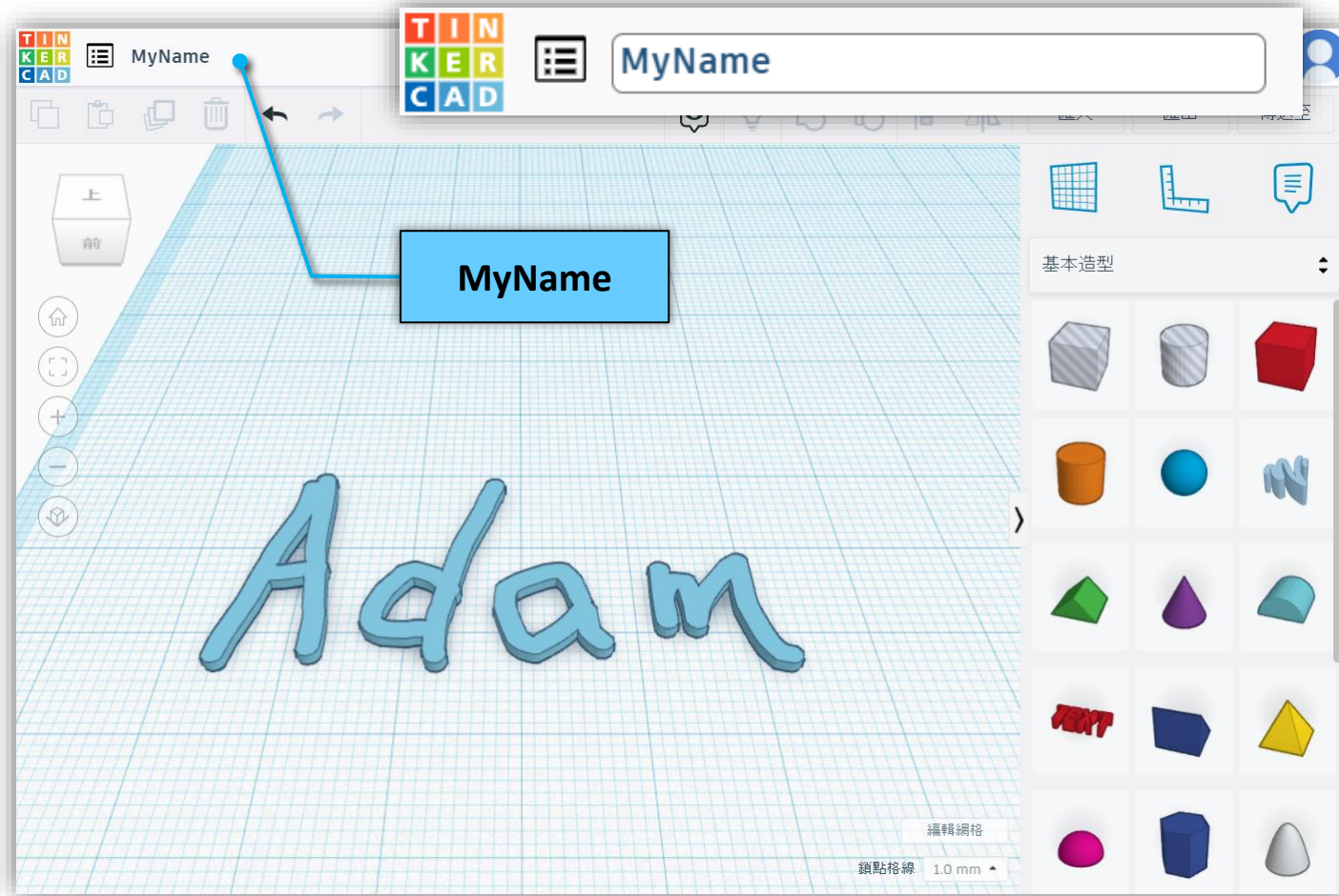
- Click Top of application to change project name to “MyName”.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 4 Change Project Name

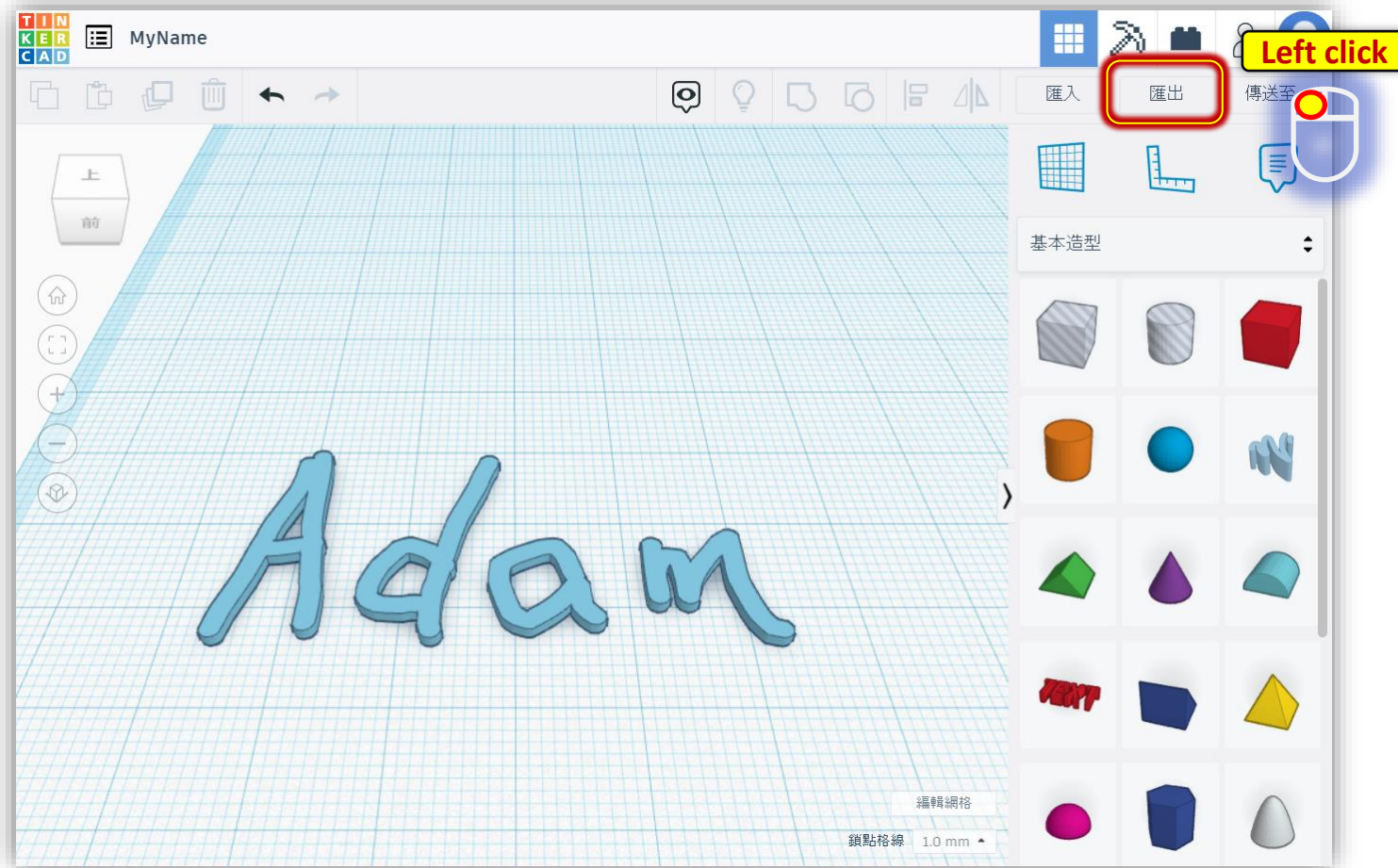
- The project name was changed to “MyName”.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

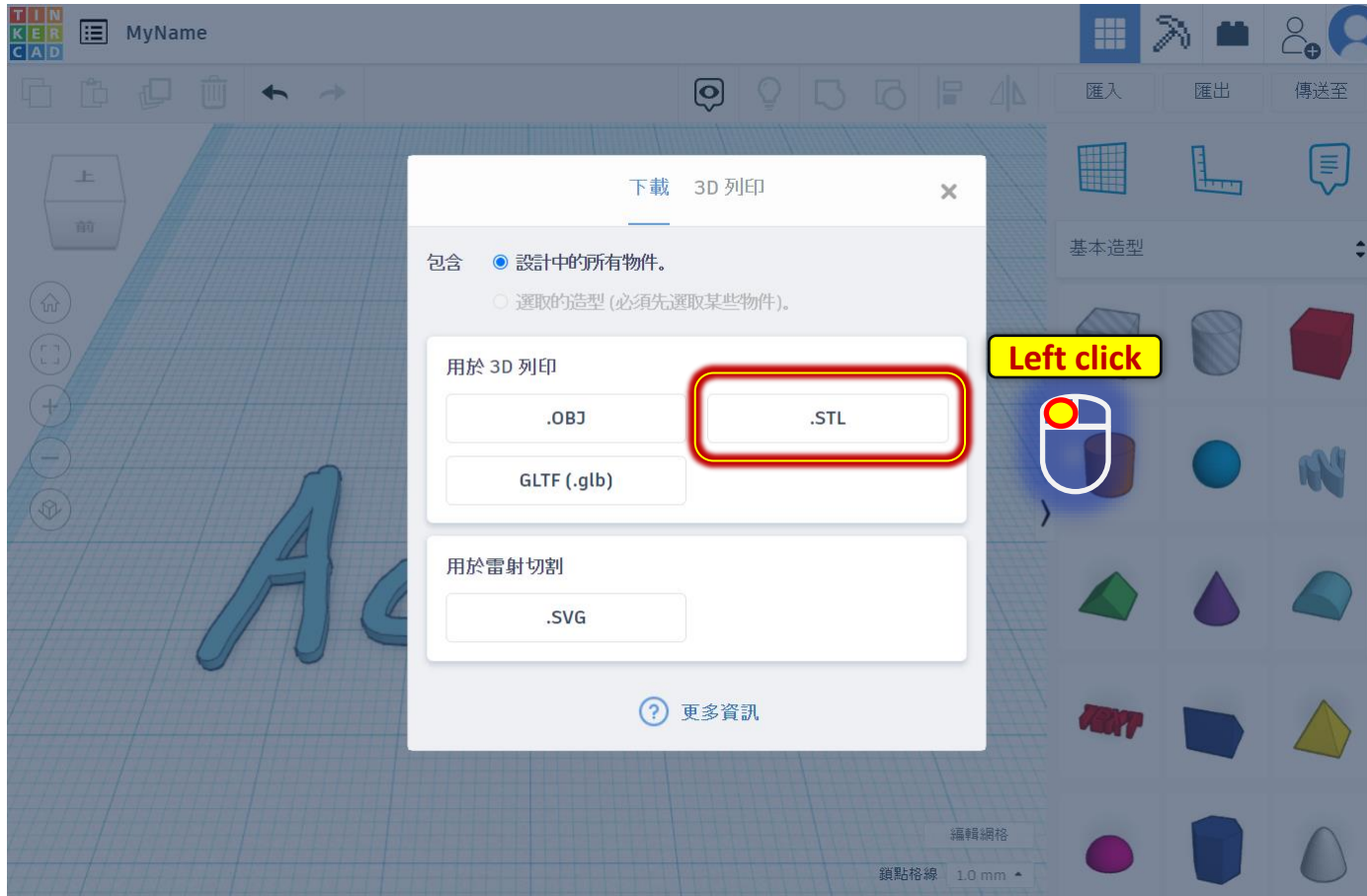
- Click “匯出” Button.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

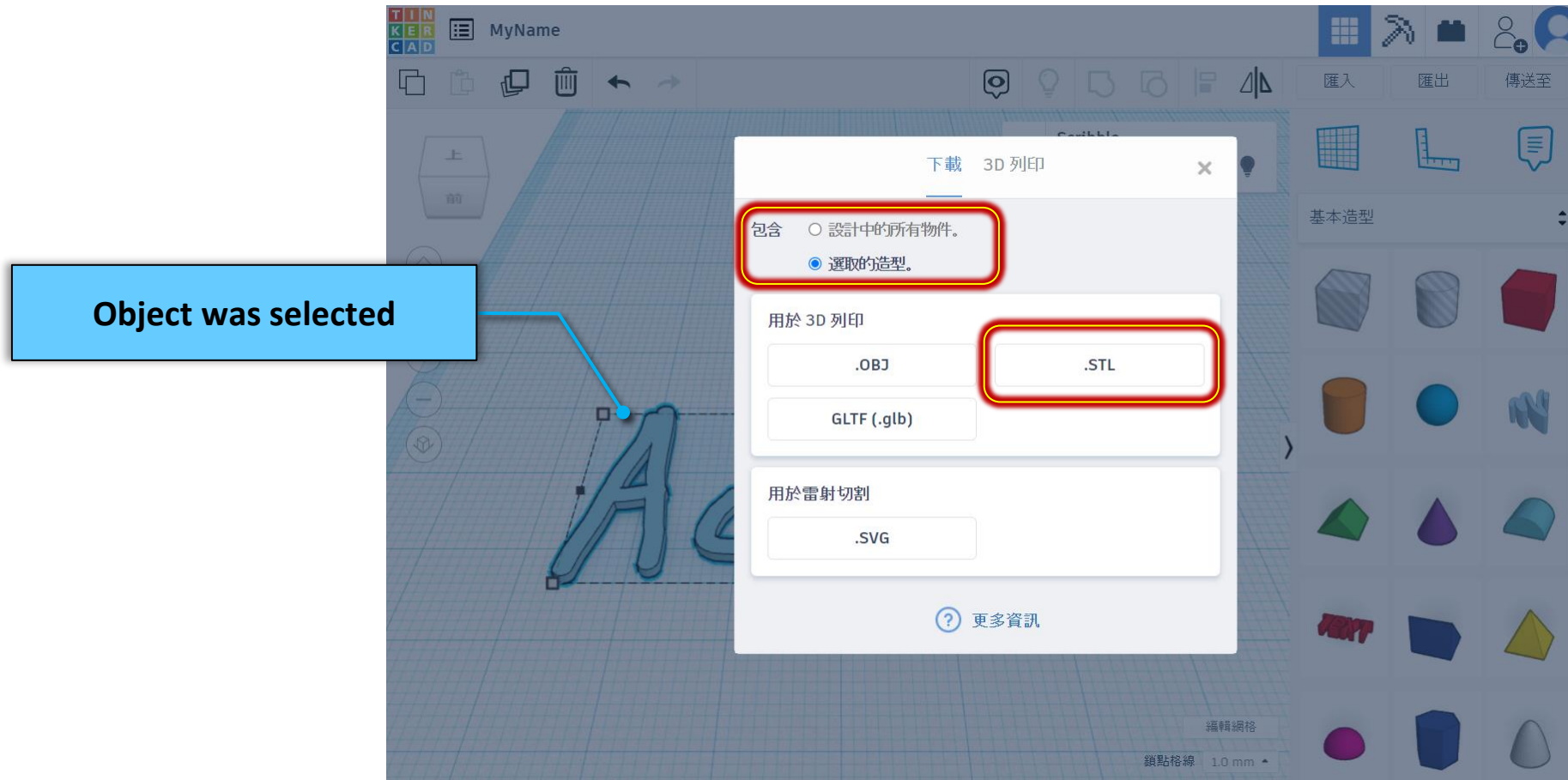
- Click “.STL” Button.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

- You will saw “☒選取的造型”, if the object was selected. Don't worry, it's same.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

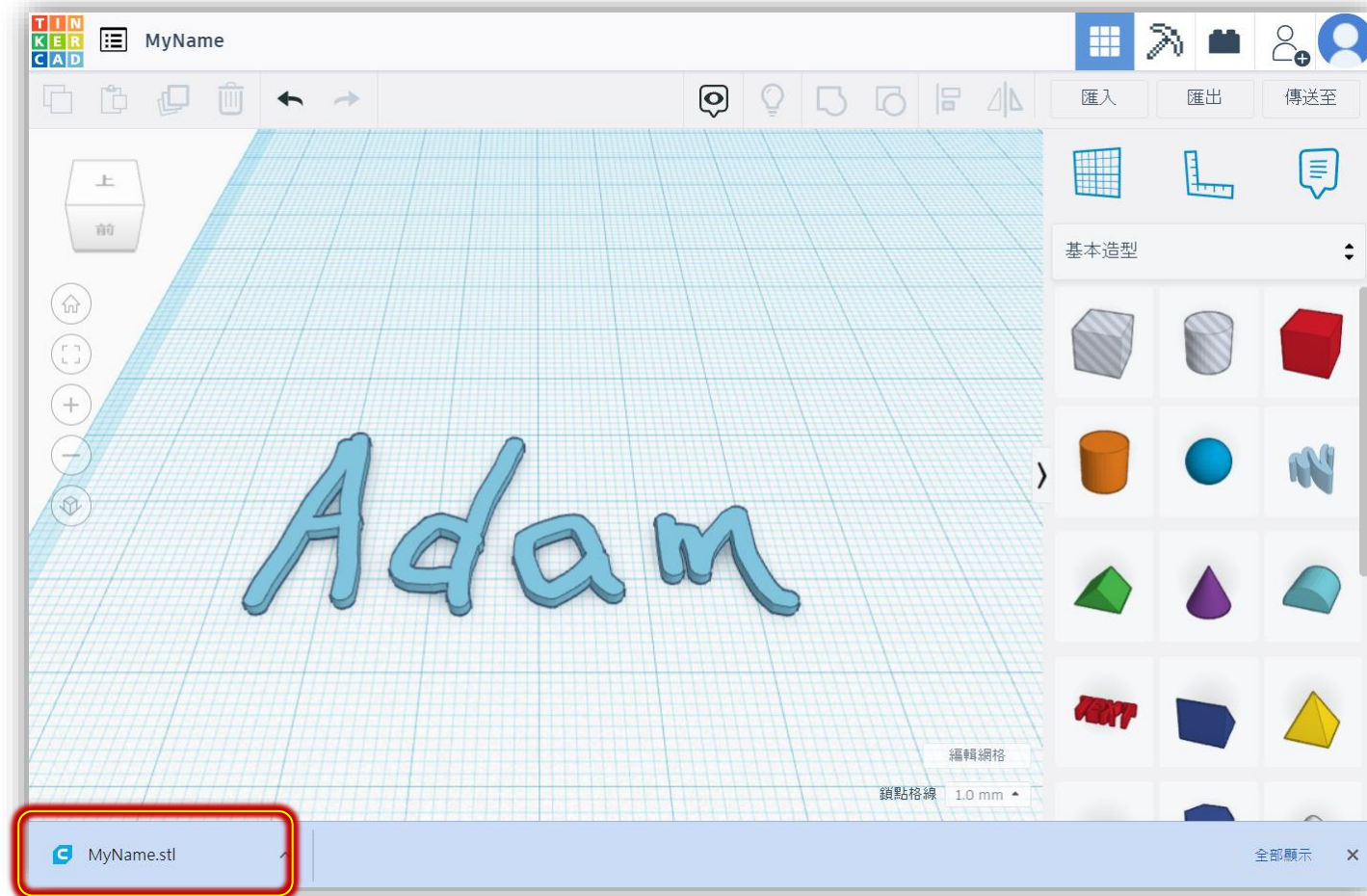
- Select local folder to save file(.stl) and select “確定”.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

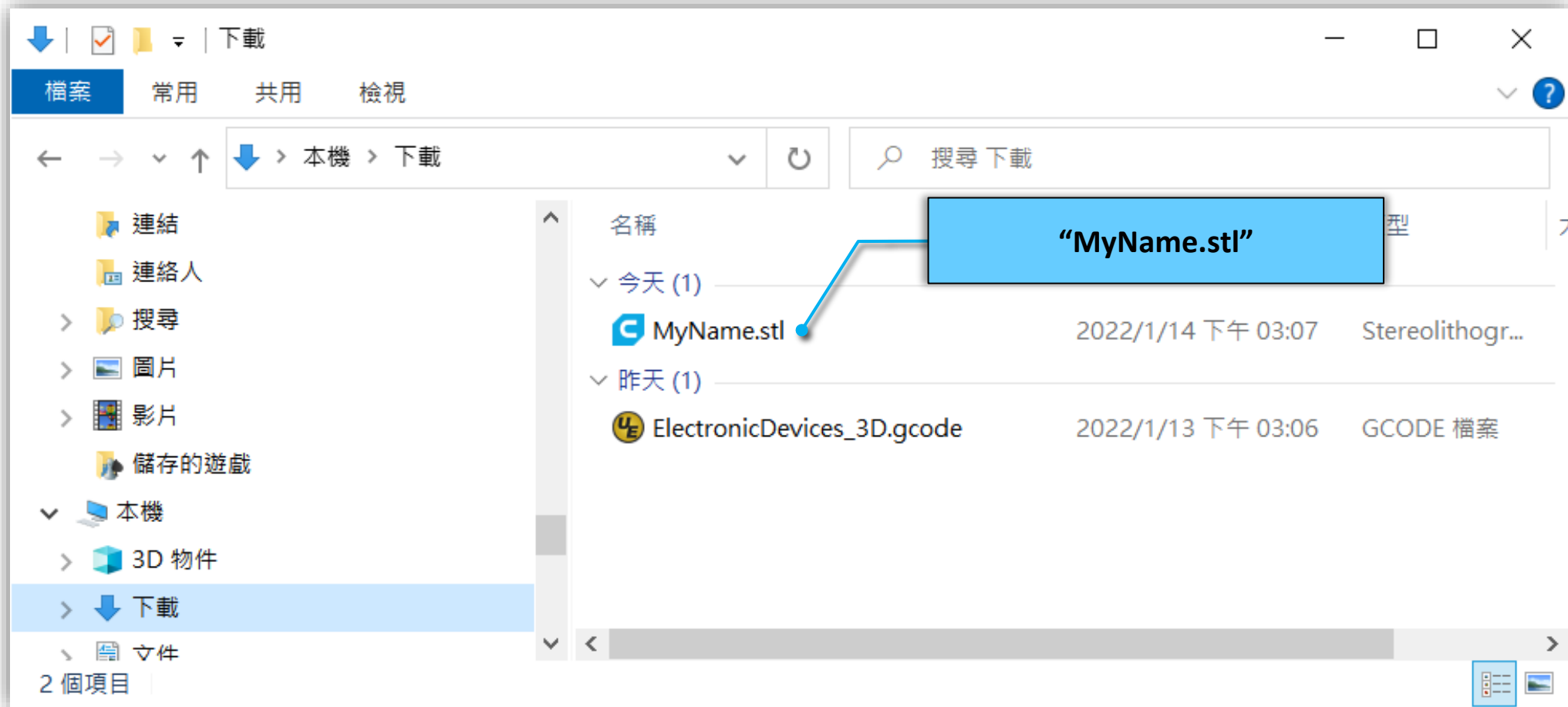
- The stl file was downloaded.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 5 Export the 3D Object to 3D file (*.stl)

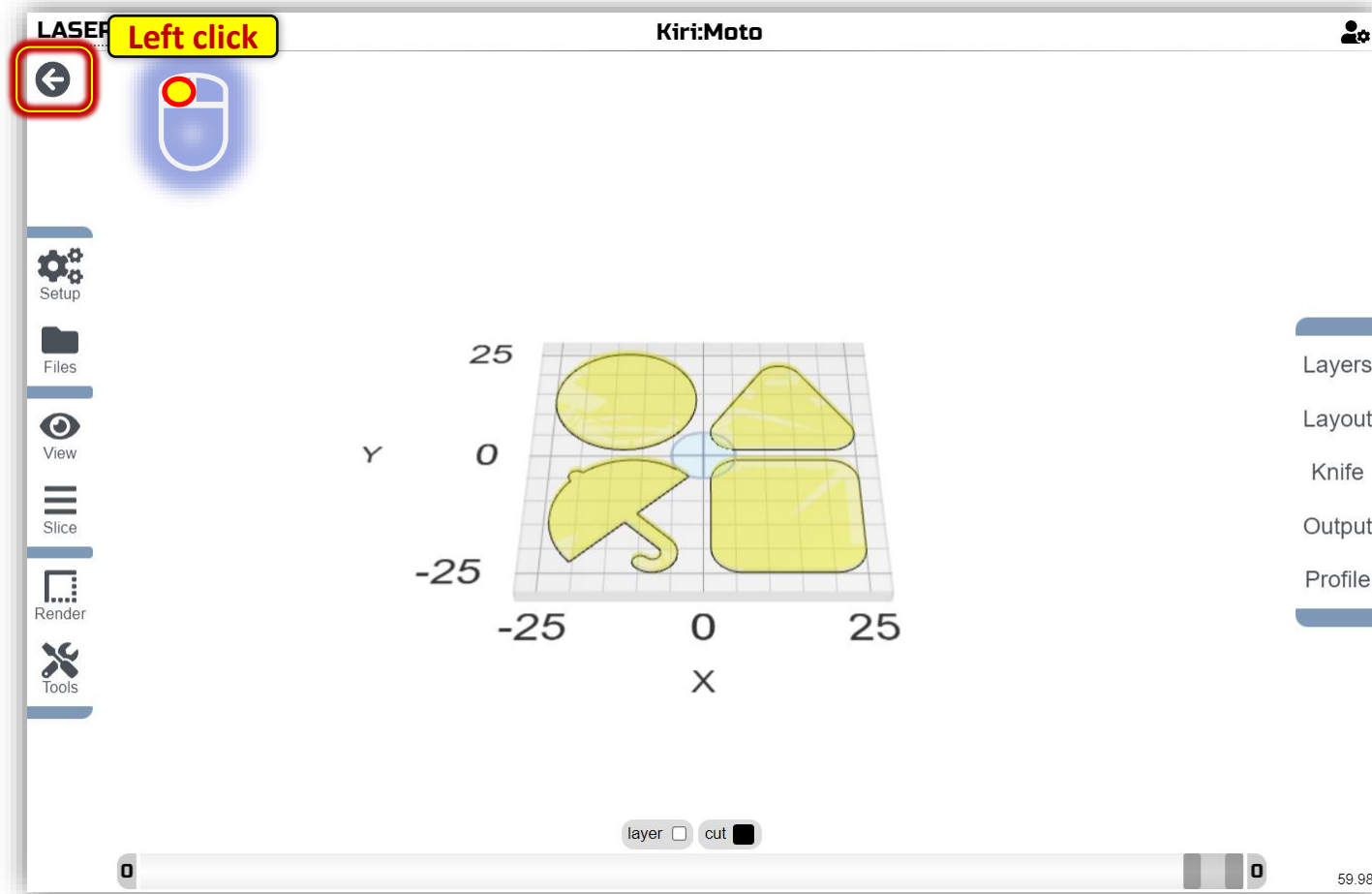
- The file named “MyName.stl” was saved.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 6 Exit Slice Mode

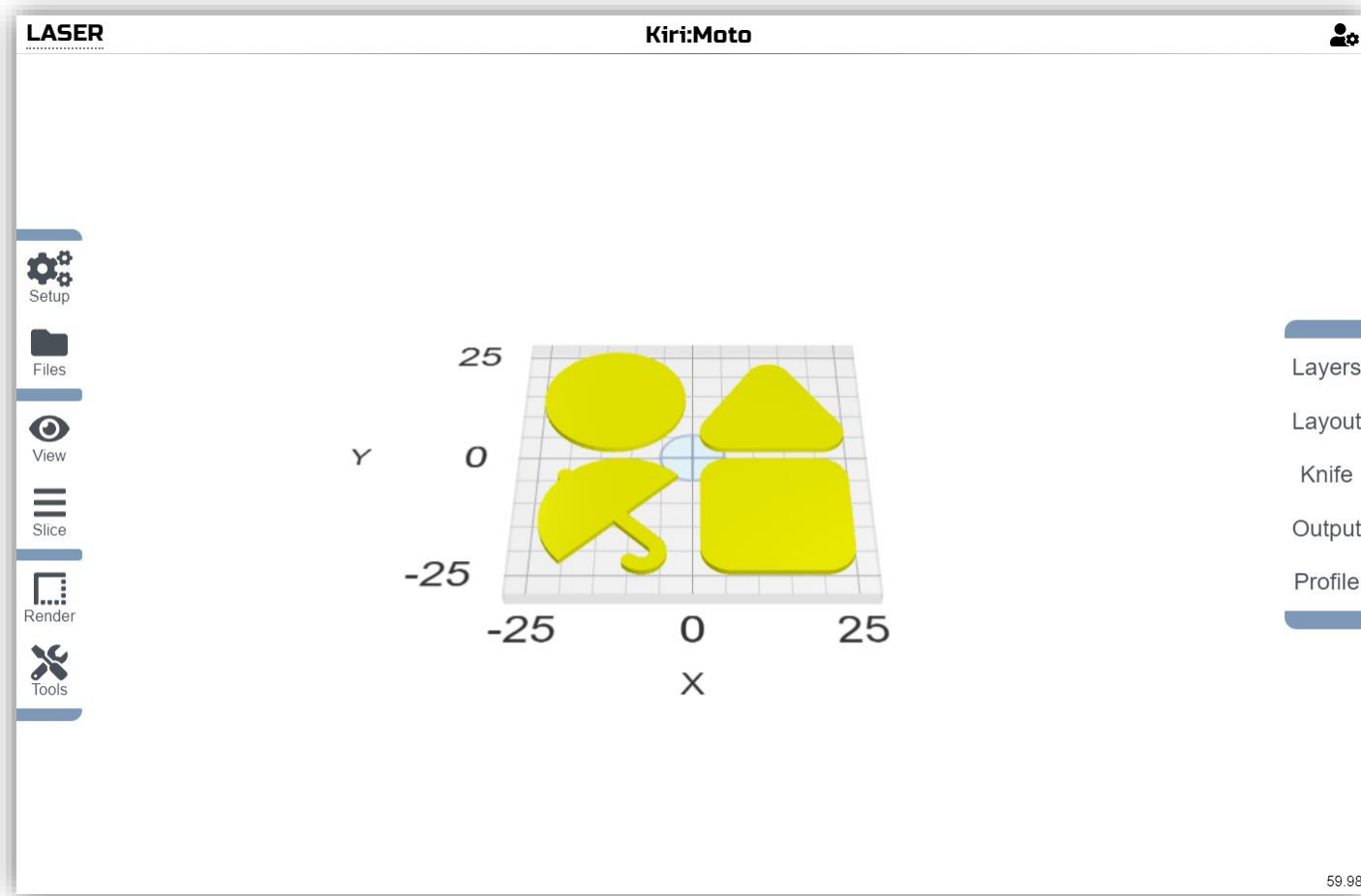
- Click “<-” to exit slice Mode



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 6 Exit Slice Mode

- Go backup to working mode

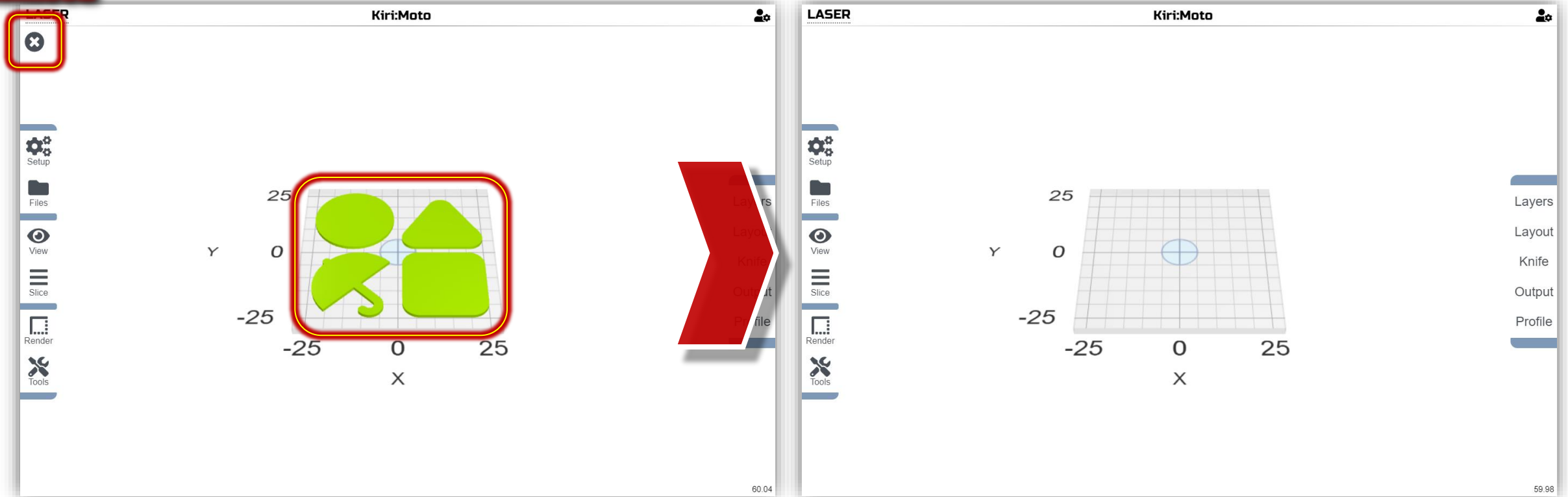


Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 7 Remove Exist Object

- Refer Exercise B Step 1. Removed exist 3D object.

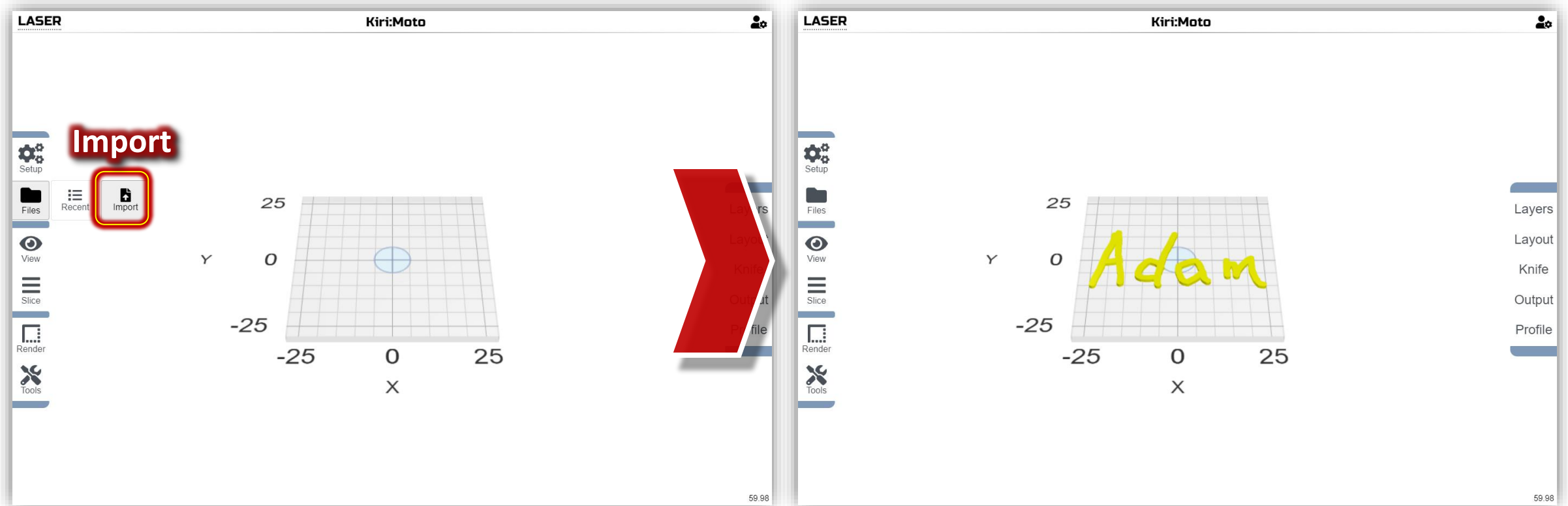
Remove



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 8 Import 3D Object

- Refer Exercise B Step 2. Import the 3D 3dobject named “MyName.stl”.



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Step 10 Export

- Refer Exercise A Step 3~5. Slice and export the “path” to gcode file named “MyName.gcode”.

The image illustrates the process of exporting a 3D model to a G-code file. It shows the LASER software interface with a 3D model of the word "Adam" on a grid. A red arrow points from the "Slice" button to the "Export" dialog box. The "Export" dialog shows "File Name" as "MyName-011" and "Segments" as "420". A red arrow points from the "Download" button to the "Save File" dialog box. The "Save File" dialog shows the file name "MyName.gcode" and the file type "GCODE 檔案 (*.gcode)".

Export

File Name: MyName-011
Segments: 420

Download

svg dxf gcode

Save File

名稱: MyName.gcode
儲存類型: GCODE 檔案 (*.gcode)

“MyName.gcode”

Exercise C Create a 3D object and Generate the G-code file at slice tool.

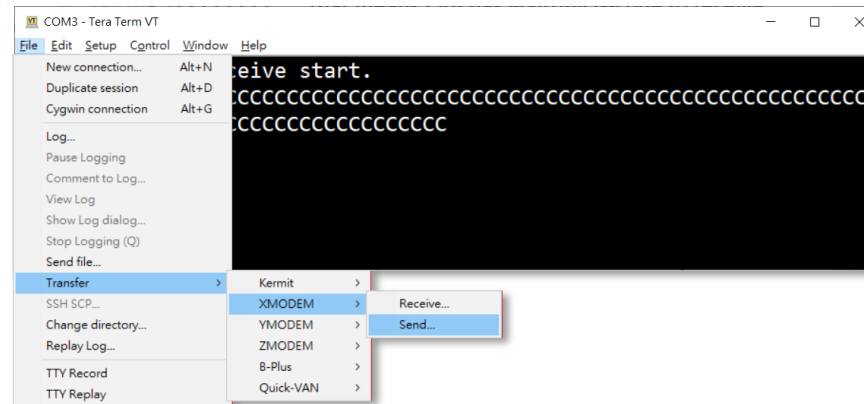
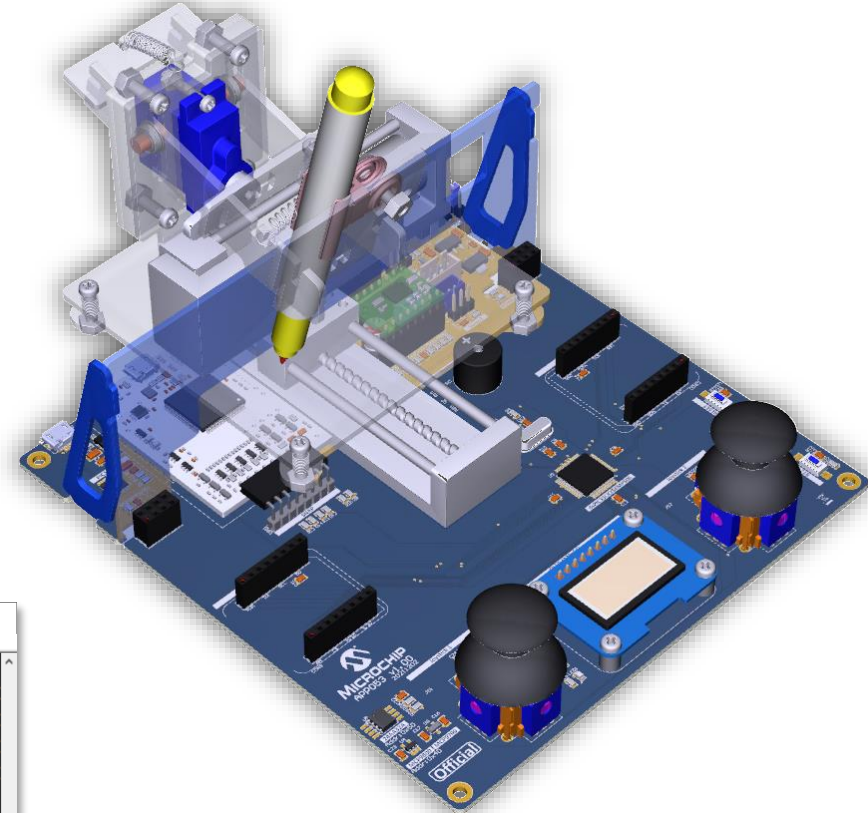
Finally, Draw the G-code

- Send the G-code to through XMODEM & enjoy.

e.g.:

```
M106 S255 \n; Laser Trun On
G1 X23.152 Y36.349 F1000 \n
G1 X24.924 Y38.849 \n
M107 \n
G0 X29.816 Y19.279 \n
M106 S255 \n
G1 X29.932 Y19.288 F1000 \n
M107 \n
...
```

MyName.gcode



Exercise C Create a 3D object and Generate the G-code file at slice tool.

Finally, Draw the G-code

- Result



Appendix

TINKERCAD Sign Up

TINKERCAD Sign Up

Step 1

- Go to TINKERCAD website (www.tinkercad.com)



TINKERCAD Sign Up

Step 2

- Click “立即加入”



The screenshot shows the Tinkercad website interface. At the top right, there is a navigation bar with links: 圖庫, 部落格, 學習, 教授, 登入, and 立即加入. The '立即加入' button is highlighted with a red box. A yellow callout box with the text 'Left click' and a mouse cursor icon points to the button. The main content area features the Tinkercad logo (a 3x3 grid of colored squares) and the text 'AUTODESK TINKERCAD'. Below the logo, there is a section titled '從構想到設計只需幾分鐘' (From idea to design in just a few minutes). The text below this title states: 'Tinkercad 是免費且易於使用的網頁應用程式，可為新一代設計師和工程師提供創新的基礎技能，可用於 3D 設計、電子器件和編碼！' (Tinkercad is a free and easy-to-use web application that provides innovative basic skills for the next generation of designers and engineers, used for 3D design, electronic components, and coding!). There are two buttons: '開始編輯' (Start editing) and '加入您的課程' (Join your course). To the right of these buttons is a 3D model of a car with a star on its side, labeled 'Workplane'. Below the model, there is a video player with the text '設計氣球動力汽車 試試看！' (Design a balloon-powered car, try it out!). At the bottom of the page, there are three sections: '3500 萬人的社群' (A community of 35 million people) with a '加入' (Join) button, '快速、免費、易於使用' (Fast, free, easy to use) with a '學習' (Learn) button, and '受全球教育工作者喜愛' (Loved by educators worldwide) with a '教授' (Teach) button.

TINKERCAD Sign Up

Step 3

- Click “建立個人帳戶”



TINKERCAD Sign Up

Step 4

- Click “使用 Google 登入”



TINKERCAD Sign Up

Step 5

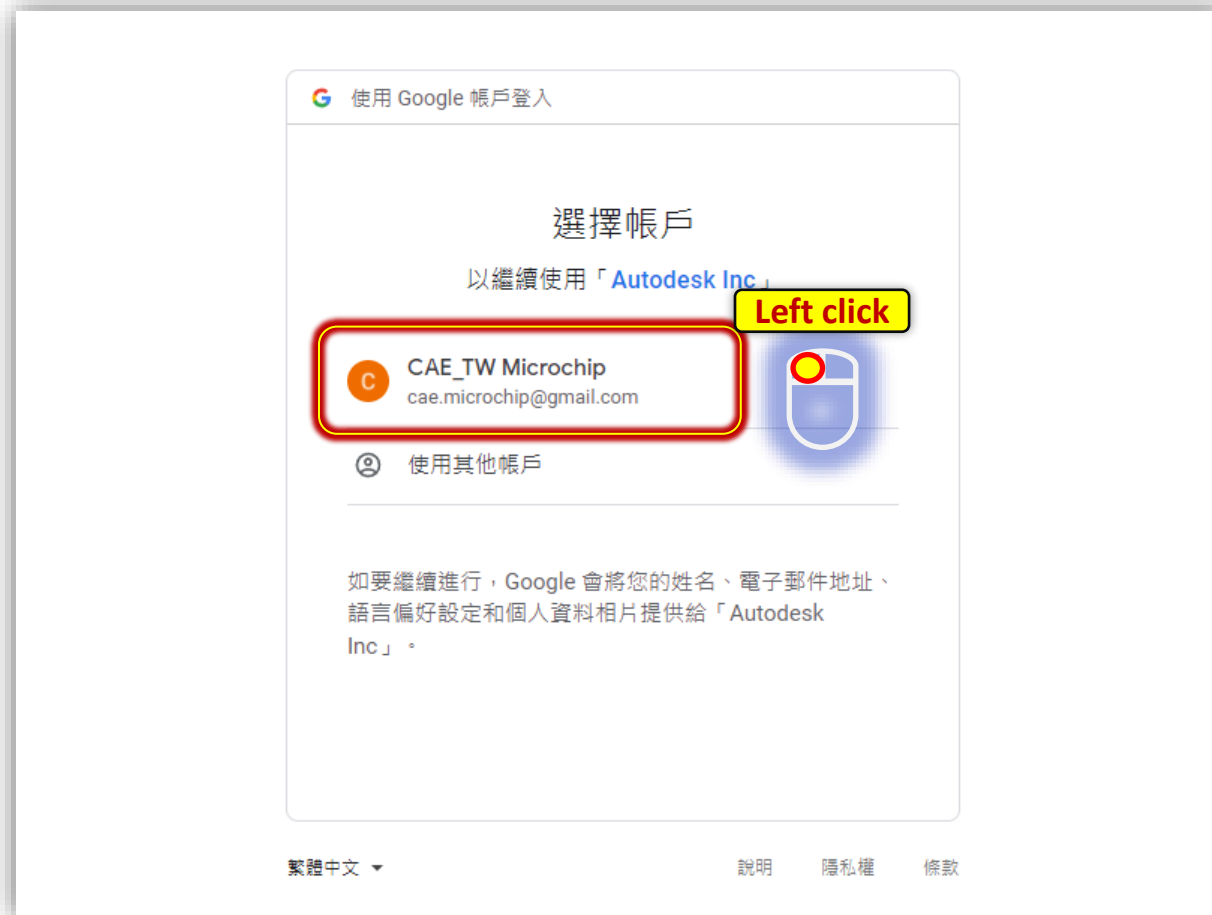
- Click “使用 Google 登入”



TINKERCAD Sign Up

Step 6

- Select your google account to sign up.



TINKERCAD Sign Up

Step 7

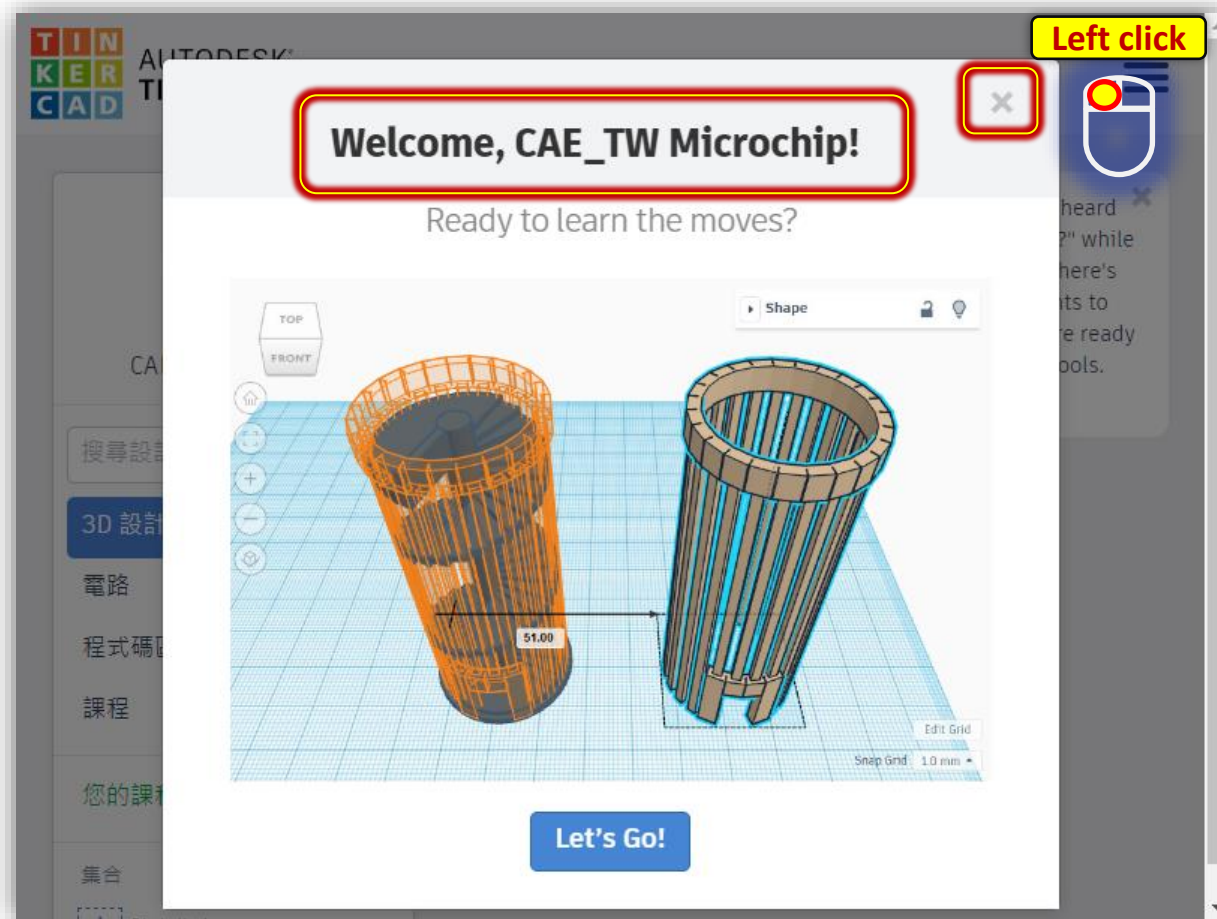
- Click “繼續” button.



TINKERCAD Sign Up

Step 8

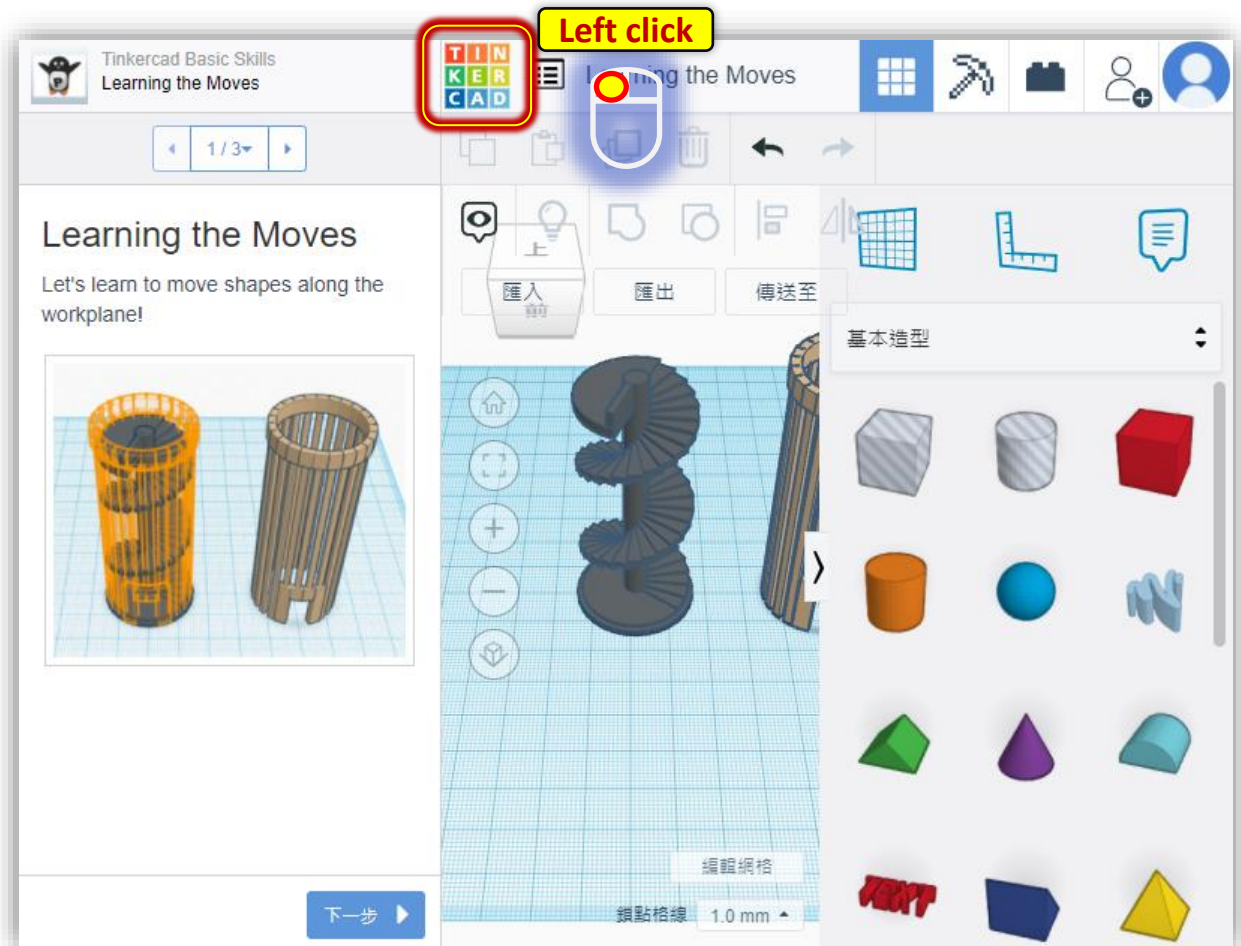
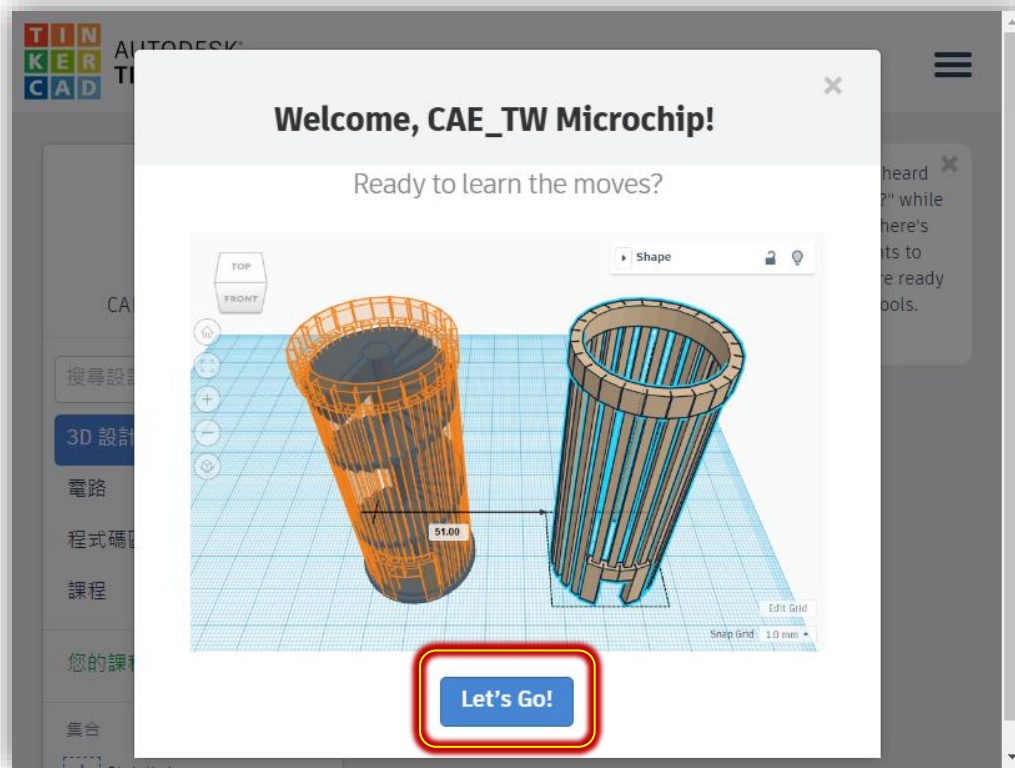
- Click “X” button.



TINKERCAD Sign Up

Step 8.a

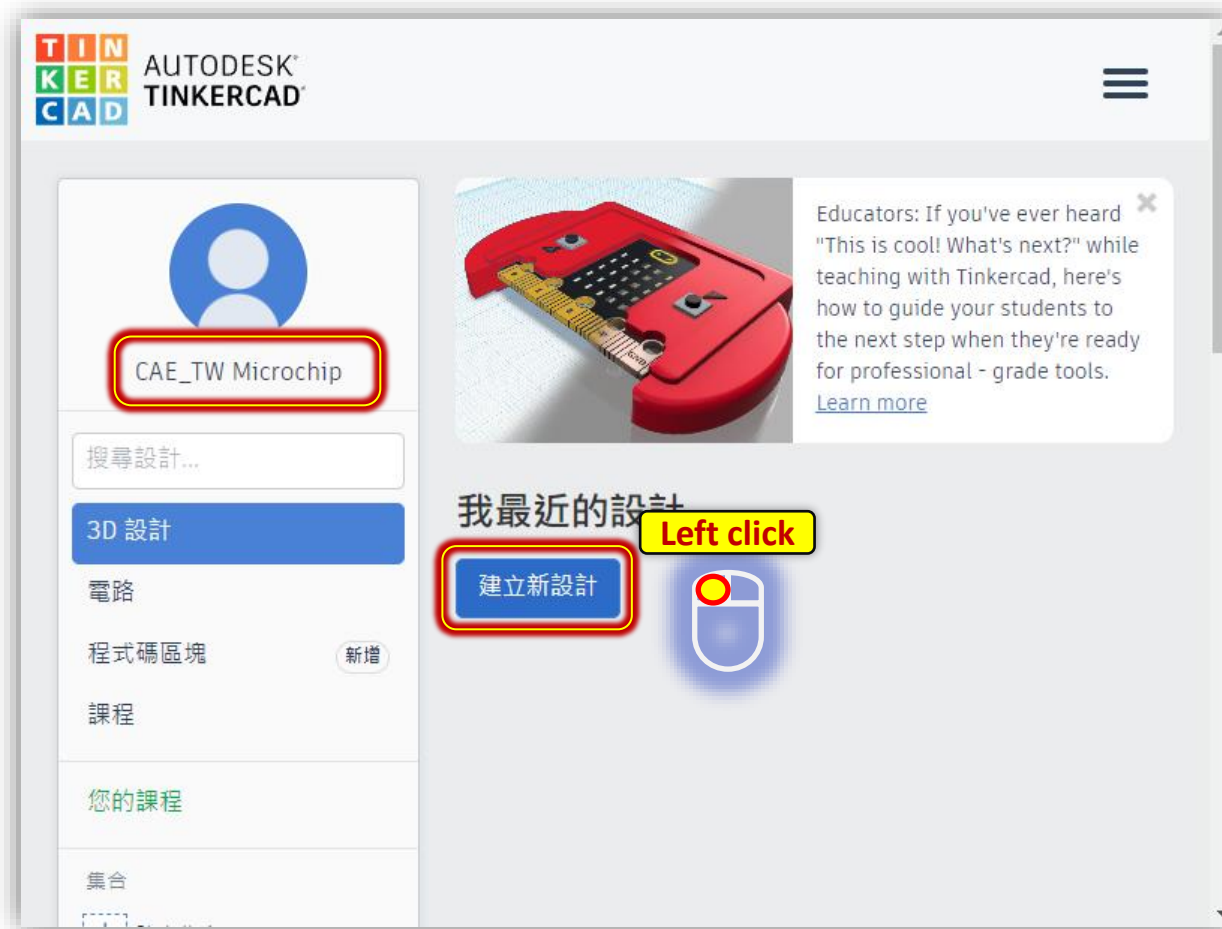
- Click “TINKERCAD” logo. If you click “Let’s go” at Step 8.



TINKERCAD Sign Up

Finally

- Select “建立新設計” and enjoy.

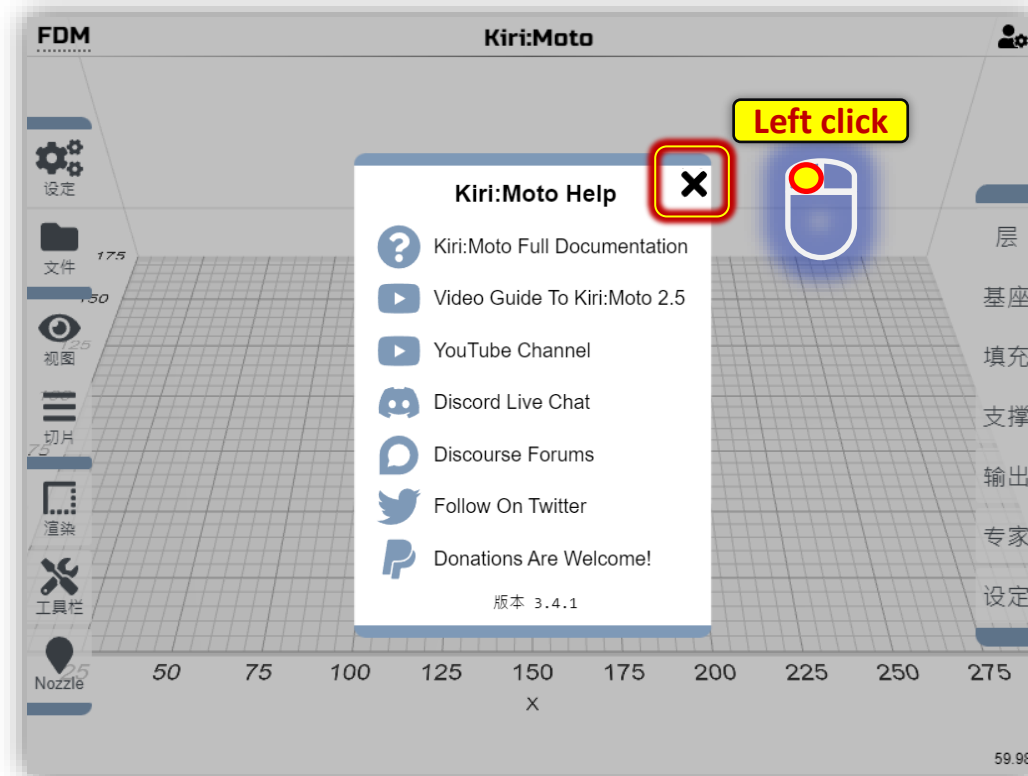


Appendix : Kiri:Moto Profiles Import

Kiri:Moto Profiles Import

Step 1

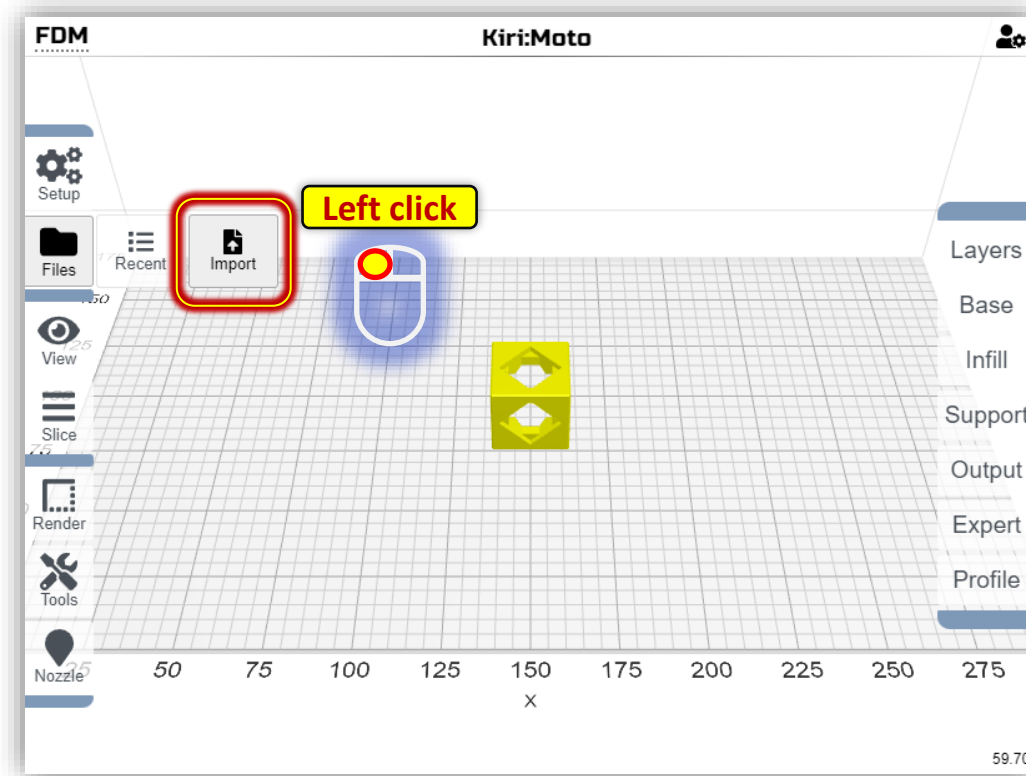
- Go to Kiri:Moto website (grid.space/kiri)



Kiri:Moto Profiles Import

Step 2

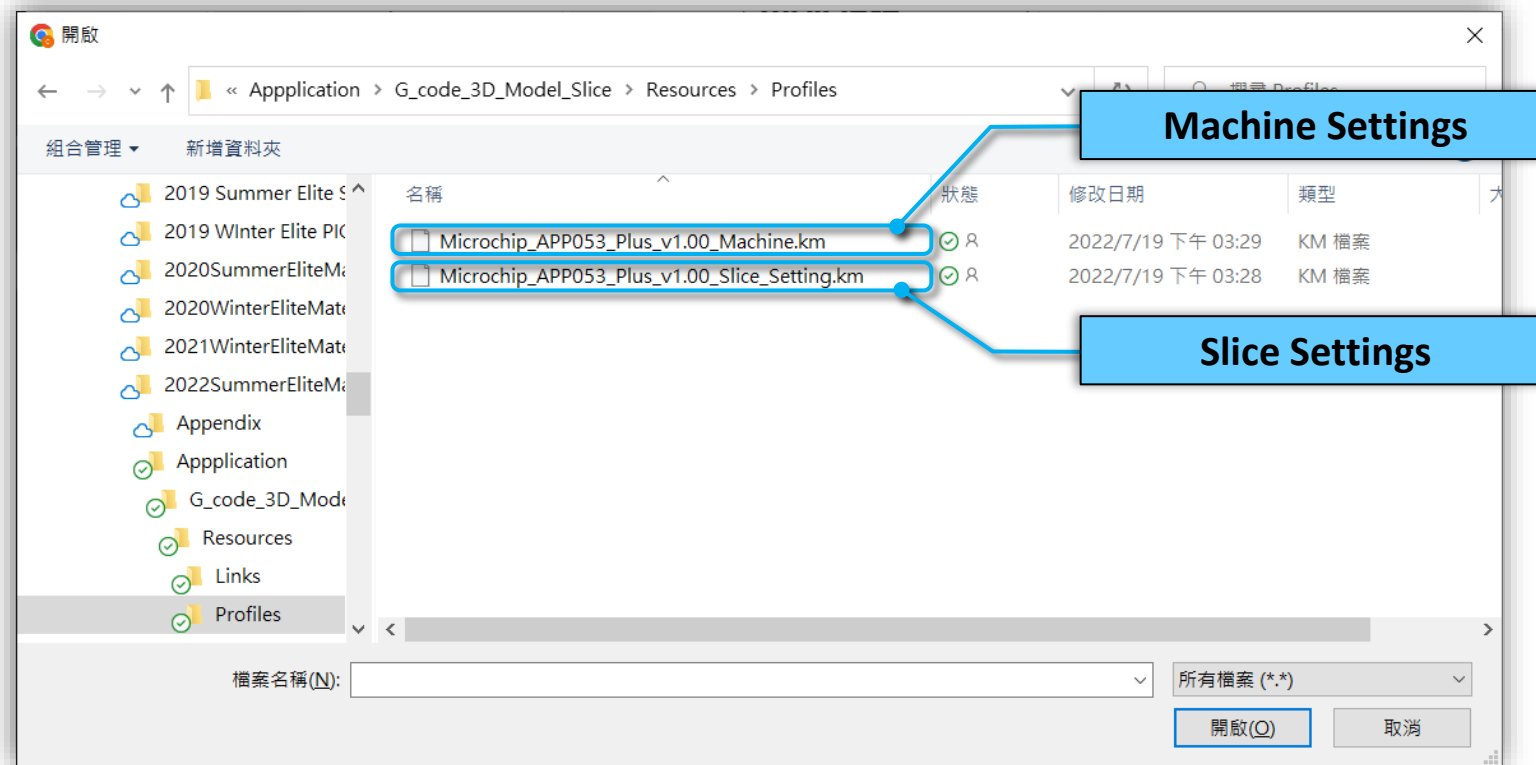
- Select Files\Import



Kiri:Moto Profiles Import

About setting files (.km)

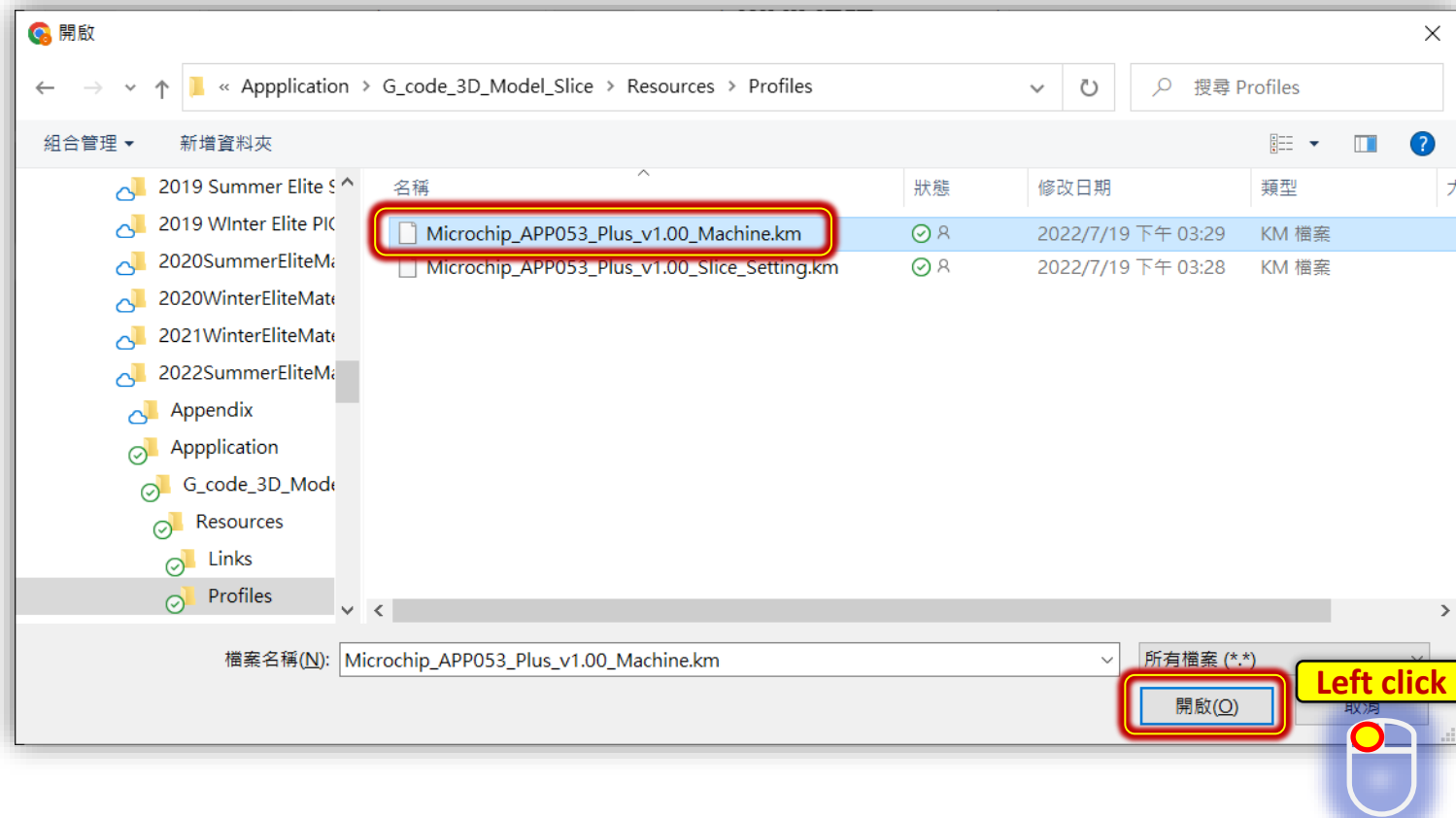
- There are two profiles one is machine setting another one is slice setting.



Kiri:Moto Profiles Import

Step 3

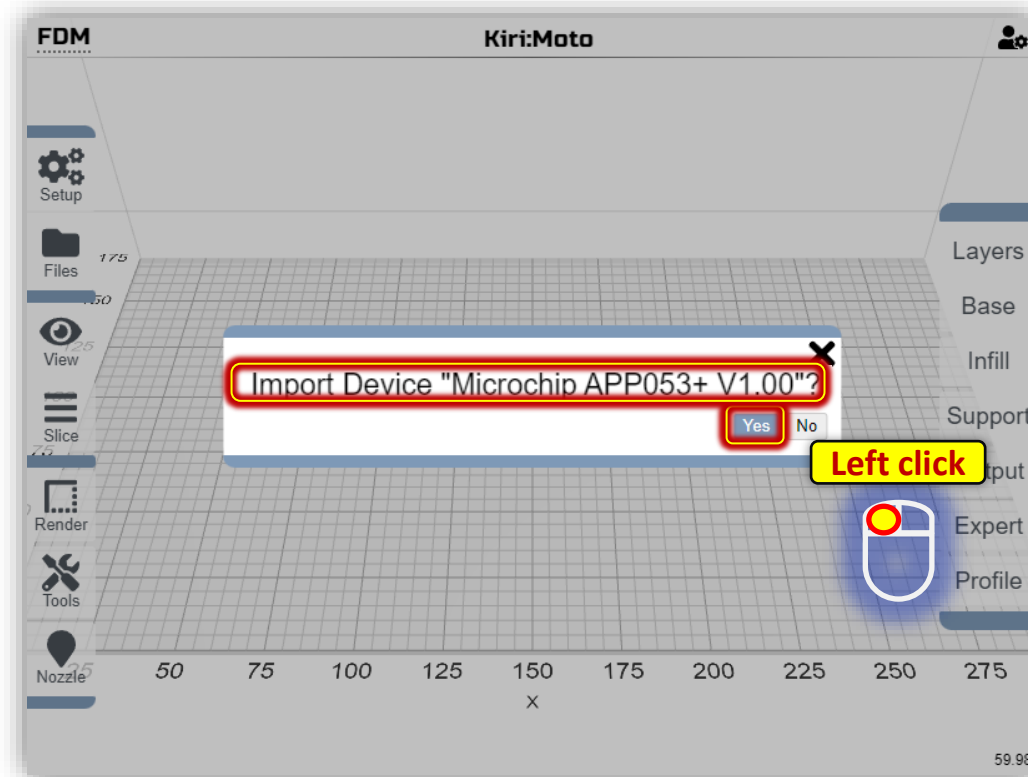
- Select “**Microchip_APP053_Plus_v1.00_Machine.km**” and click “確定”



Kiri:Moto Profiles Import

Step 4

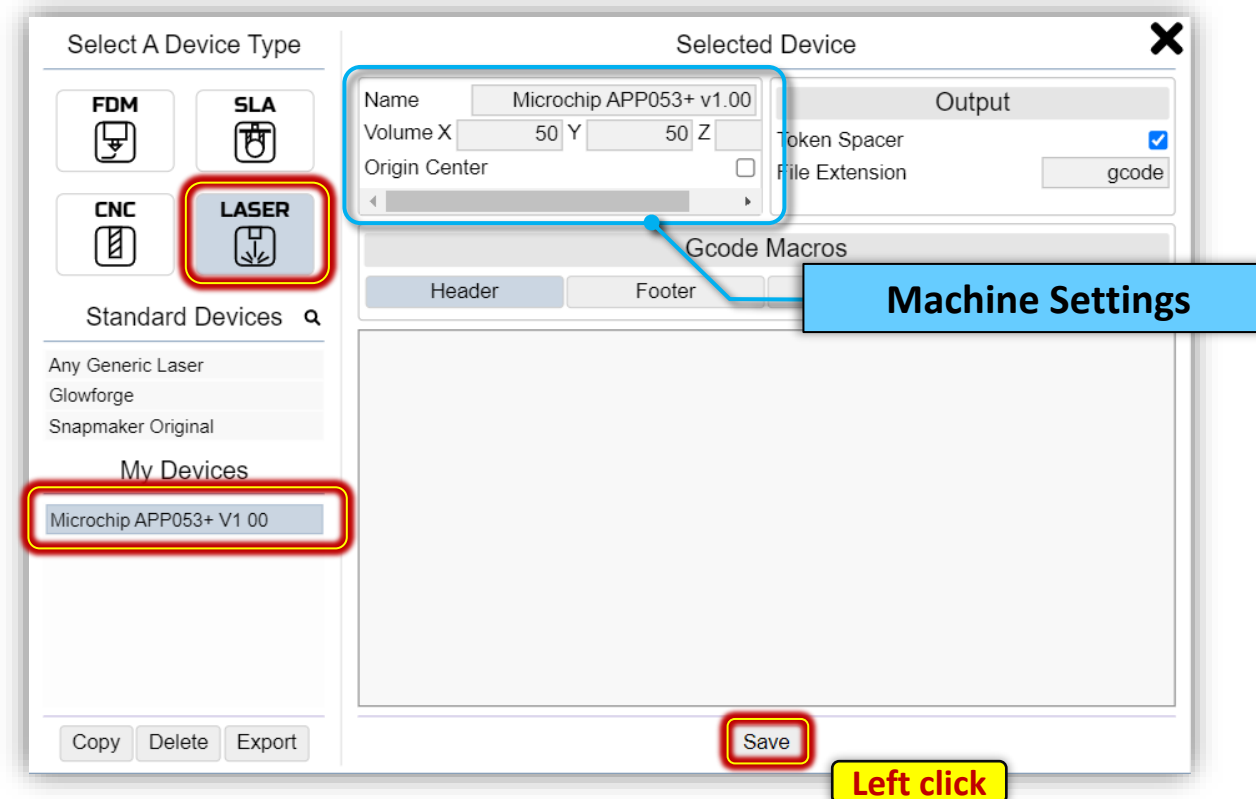
- Select “Yes” to apply setting



Kiri:Moto Profiles Import

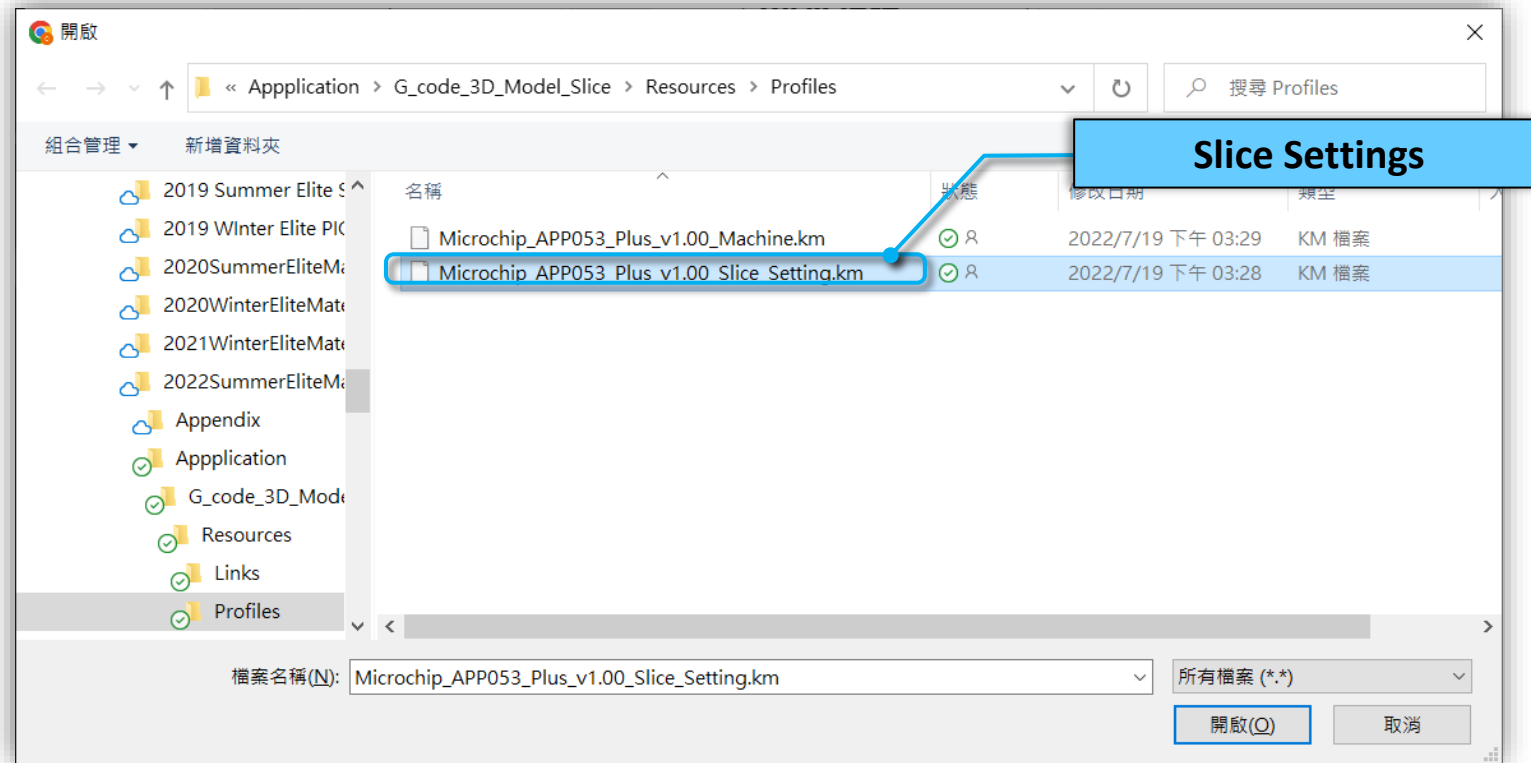
Step 5

- Select “LASER > **Microchip APP053+ V1.00**” and click “Save”.
- Setting was applied.



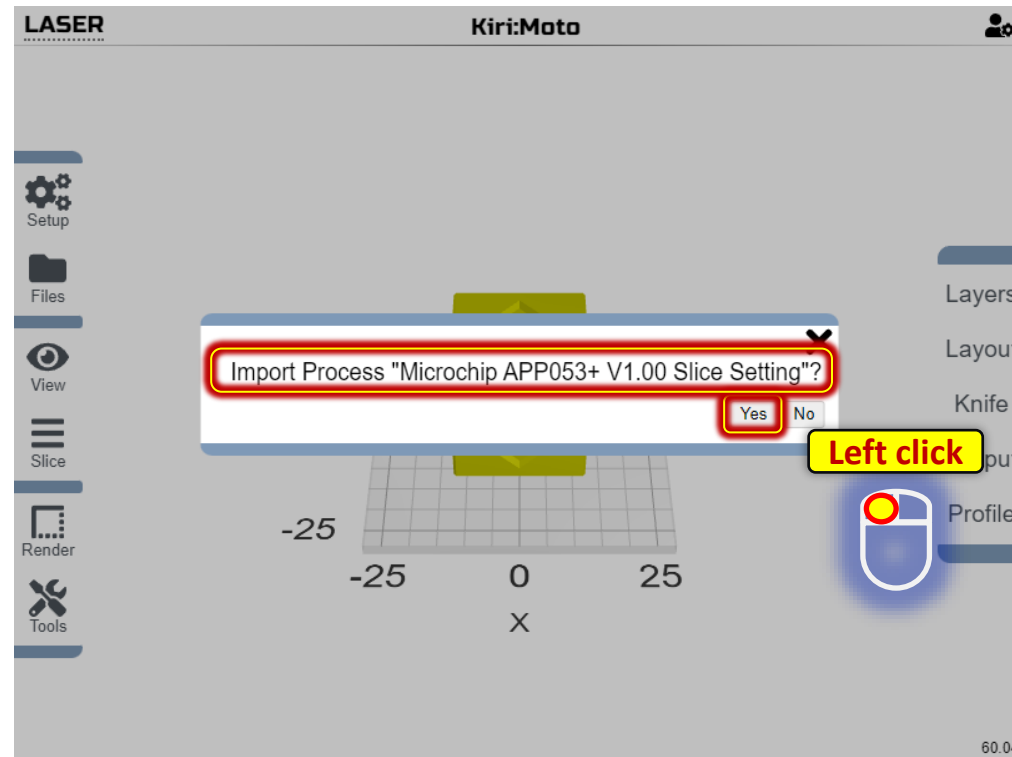
Kiri:Moto Profiles Import

- You can import slice setting, also.



Kiri:Moto Profiles Import

- Select “Yes” to apply slice setting



Kiri:Moto Profiles Import

LASER

Kiri:Moto

Setup

Files

View

Slice

Render

Tools

Offset0.01

Height0.01

Single☒

Layers

Layout

Knife

Output

Profile

25

0

-25

-25

0

25

MICROCHIP

Setup

Files

View

Slice

Render

Tools

LASER

Kiri:Moto

Setup

Files

View

Slice

Render

Tools

Spacing1

Merged☐

Grouped☒

Layers

Layout

Knife

Output

Profile

25

0

-25

-25

0

25

MICROCHIP

Setup

Files

View

Slice

Render

Tools

LASER

Kiri:Moto

Setup

Files

View

Slice

Render

Tools

Cut Depth0

Cut Passes1

Tip Offset2

Enable☐

Layers

Layout

Knife

Output

Profile

25

0

-25

-25

0

25

MICROCHIP

Setup

Files

View

Slice

Render

Tools

LASER

Kiri:Moto

Setup

Files

View

Slice

Render

Tools

Power100

Speed1000

Layer Color☐

Layer Order☐

Origin Bounds☐

Origin Center☒

Layers

Layout

Knife

Output

Profile

25

0

-25

-25

0

25

MICROCHIP

Setup

Files

View

Slice

Render

Tools

LASER

Kiri:Moto

Setup

Files

View

Slice

Render

Tools

Saved Settings

Microchip APP053+ v1.00 Slice Setting

Microchip APP053+ v1.00 Slice Setting

Save

25

0

-25

-25

0

25

MICROCHIP

Setup

Files

View

Slice

Render

Tools

Slice Setting

Microchip Trademarks

Overview

Microchip Technology Incorporated's products are marketed and sold under one or more of the following trademarks belonging to Microchip or one of Microchip's subsidiaries. Questions regarding use of these trademarks should be addressed to the Microchip's Legal Department via email: legal.department@microchip.com.

The following are registered trademarks of Microchip in the U.S.A. and other countries:

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA

The following are registered trademarks of Microchip in the U.S.A:

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, WinPath, and ZL

The following are registered trademarks of Microchip in other countries: BeaconThings, GestIC, Silicon Storage Technology

The following are trademarks of Microchip in the U.S.A. and other countries:

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICKit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA

The following is a service mark of Microchip in the U.S.A.: SQTP

The following are logos of Microchip in the U.S.A. and other countries: The Adaptec logo, Frequency on Demand, Silicon Storage Technology, Symmcom, and Trusted Time

The following is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries: GestIC



Thank You !!
