

# Microcontroller & MCC PIC1002 v1.00



---

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



SMART | CONNECTED | SECURE

**CAE Taiwan**  
**Microchip Technology Inc.**

July 1, 2022

# Major Course (1/3)

- IDE, Compilers, MCC and Development Tools Introduction
- APP1618-SD EVB Introduction
- Getting Started with First Project
  -  Lab0 First Project (MCC)
- GPIO Architecture
  -  Lab1.2 GPIO Output (MCC)
  -  Lab2.2 GPIO Input (MCC)
- Oscillator Architecture
  -  Lab3.2 Clock PRIPLL (MCC)
- Interrupt Architecture
- Timer Architecture
  -  Lab4.2 Timer2 Interrupt (MCC)

# Major Course (2/3)

- ADC Architecture

-  Lab5.2 ADC Channel Periodically (MCC)

- LCD Architecture

-  Lab7 LCD Display (MCC)

- UART Architecture

-  Lab8 UART Printf (MCC)

-  Lab9 UART Communication (MCC)

- I2C Architecture

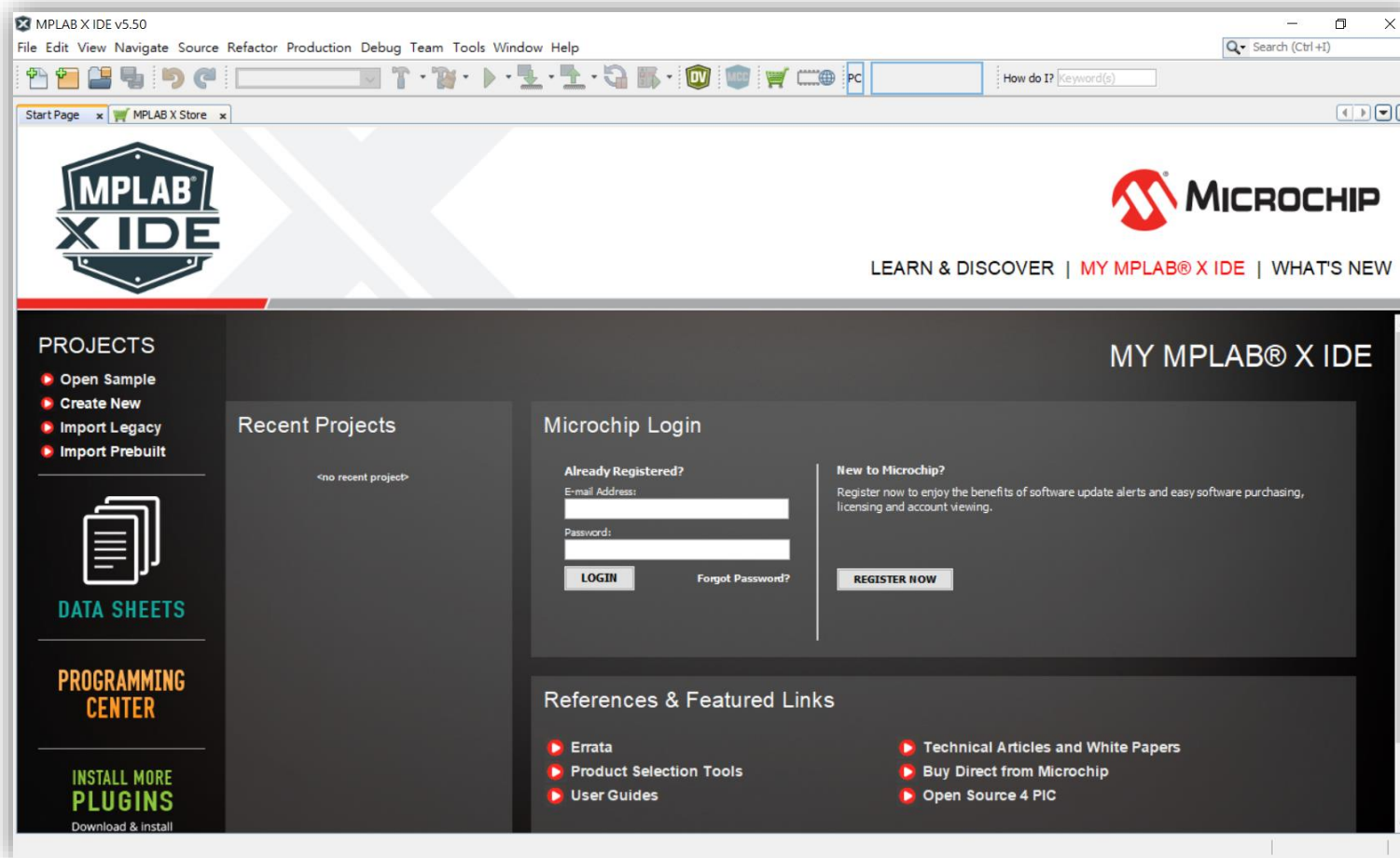
-  Lab10 Temperature Sensor (MCP9800) Operation (MCC)

# 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 8-bits PIC/dsPIC® MCU

- XC8 is new generation C compiler, it's supported all 8-bits PIC/AVR MCU (PIC10F/PIC12F /PIC16F /PIC18F, AVR / ATINY).
- 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.

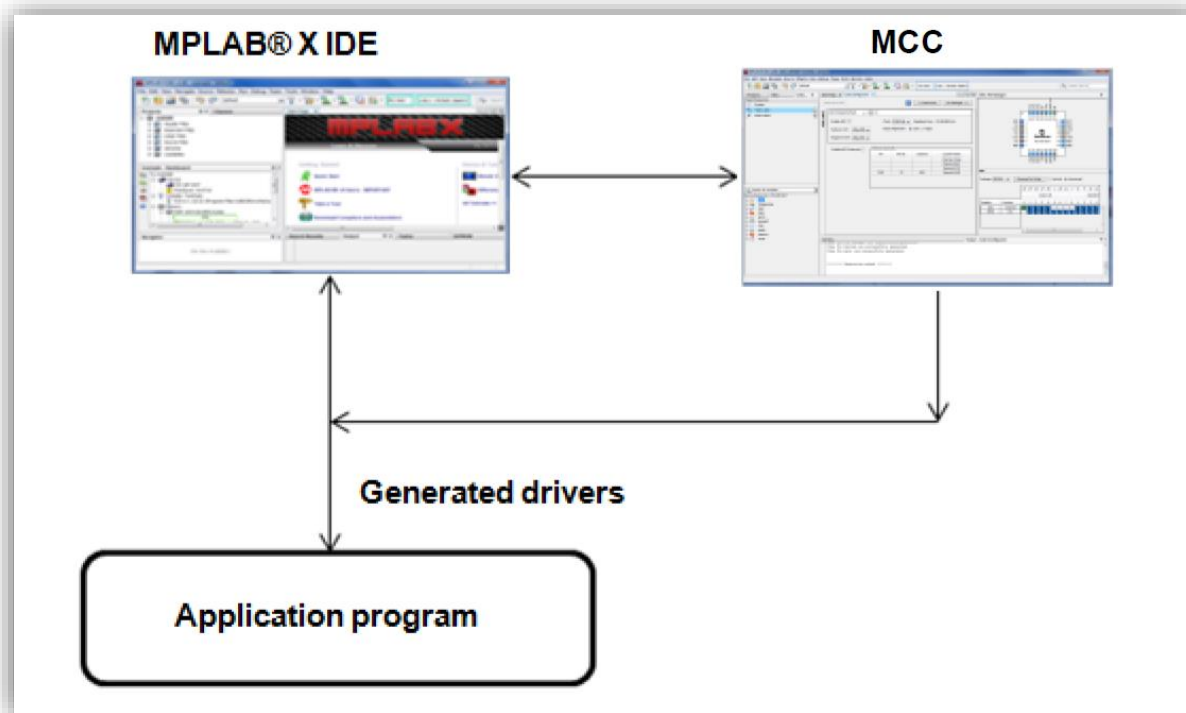
| License Type           | Installs On | # of Activations | # of Users     | Wait Time Between Users | HPA |
|------------------------|-------------|------------------|----------------|-------------------------|-----|
| Workstation License    | Workstation | 3                | 1              | None                    | 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

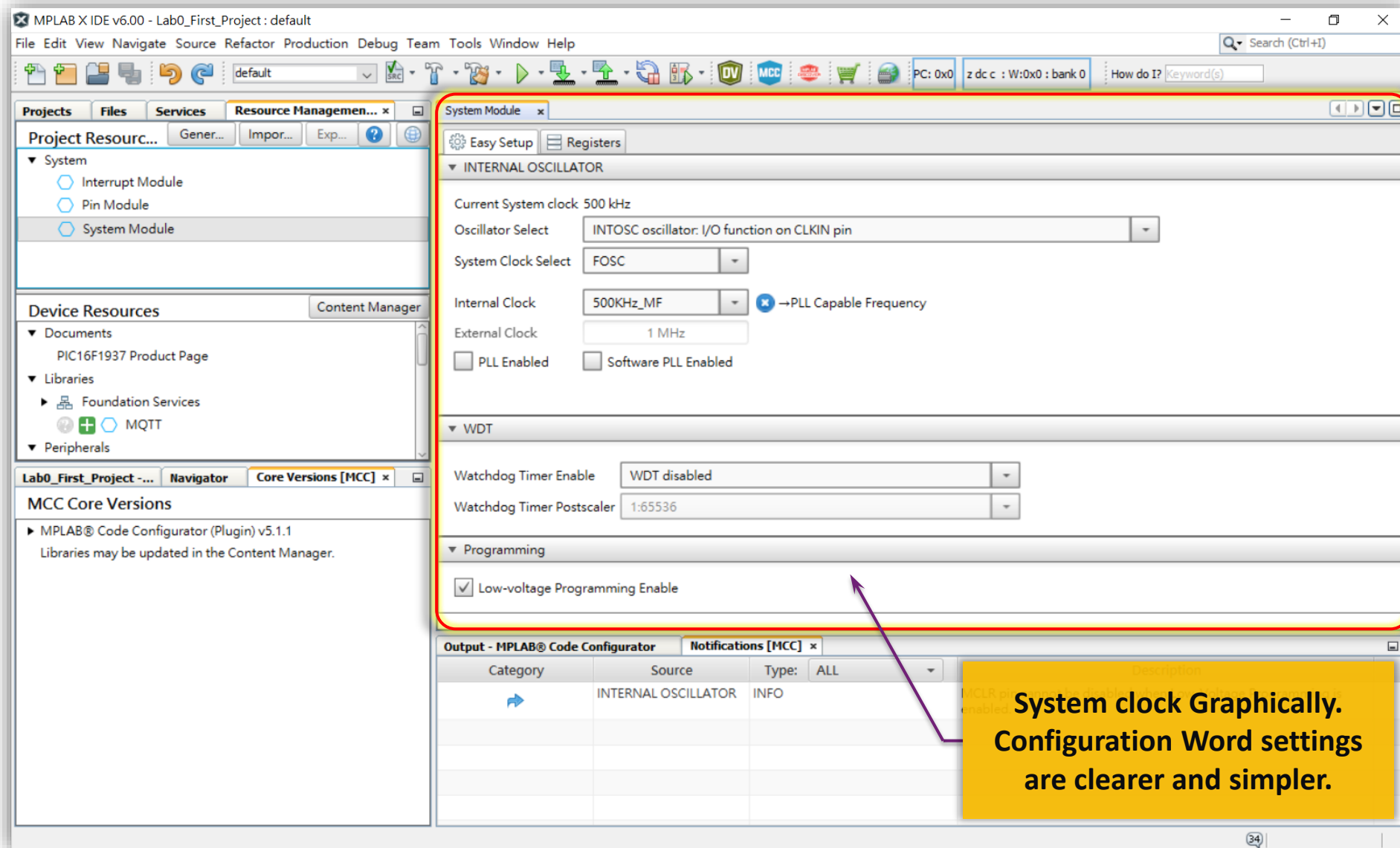
The screenshot displays the MPLAB X IDE v6.00 interface with the MCC (Microchip Code Configurator) GUI. The interface is divided into several panes:

- Project Resources:** Located on the left, it shows a tree view of project resources including System (Interrupt Module, Pin Module, System Module), Device Resources (PIC16F1937 Product Page), Libraries (Foundation Services, MQTT), and Peripherals.
- System Module:** The central pane shows configuration options for the INTERNAL OSCILLATOR, including Current System clock (500 kHz), Oscillator Select (INTOSC oscillator: I/O function on CLKIN pin), System Clock Select (FOSC), Internal Clock (500KHz\_MF), External Clock (1 MHz), and Watchdog Timer settings (WDT disabled, Postscaler 1:65536).
- Pin Manager: Package View:** Located on the right, it displays a pin diagram for the PIC16F1937 microcontroller, showing pin numbers and names (RC0-RC7, RA0-RA5, RB0-RB7, VSS, VDD).
- Bottom Pane:** Contains three sub-panes: Output - MPLAB® Code Configurator, Pin Manager: Grid View, and Notifications [MCC].

Three yellow callout boxes with arrows point to specific areas:

- Resources:** Points to the Project Resources pane.
- Composer:** Points to the System Module configuration pane.
- Pin Manager:** Points to the Pin Manager (Package View) pane.

# MCC's GUI Quick View



# MCC's GUI Quick View

**Project Resource Manager**

- System
  - Interrupt Module
  - Pin Module
  - System Module

**Device Resources**

- Documents
  - PIC16F1937 Product Page
- Libraries
  - Foundation Services
  - MQTT
- Peripherals

**Lab0\_First\_Project - ...**

- Navigator
  - Core Versions [MCC]

**MCC Core Versions**

- MPLAB® Code Configurator (Plugin) v5.1.1
  - Libraries may be updated in the Content Manager.

**Pin Manager: Grid View**

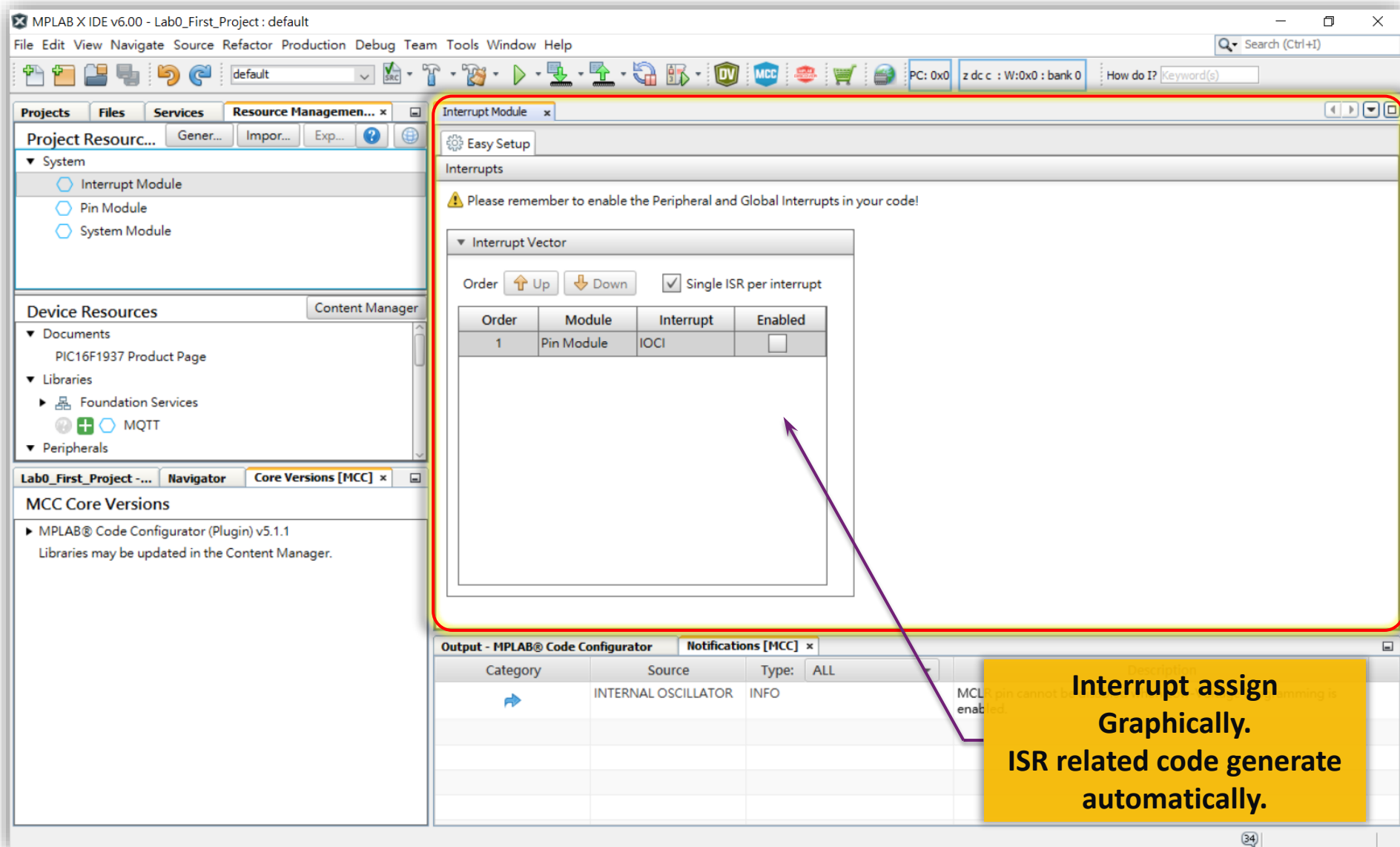
| Package:   | UQFN40 | Pin No:  | 17 | 18 | 19 | 20 | 21 | 22 | 29 | 28 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 30 | 31 | 32 | 33 | 38 | 39 | 40 | 1 | 34 | 35 | 36 |
|------------|--------|----------|----|----|----|----|----|----|----|----|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|---|----|----|----|
| Module     |        | Function |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |
| OSC        |        | CLKOUT   |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |
| Pin Module |        | GPIO     |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |
|            |        | GPIO     |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |
| RESET      |        | MCLR     |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |
|            |        | VCAP     |    |    |    |    |    |    |    |    |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |   |    |    |    |

**Output - MPLAB® Code Configurator**

| Category | Source              | Type | ALL |
|----------|---------------------|------|-----|
|          | INTERNAL OSCILLATOR | INFO |     |

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

# MCC's GUI Quick View



# More ?

[www.microchip.com/mcc](http://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).

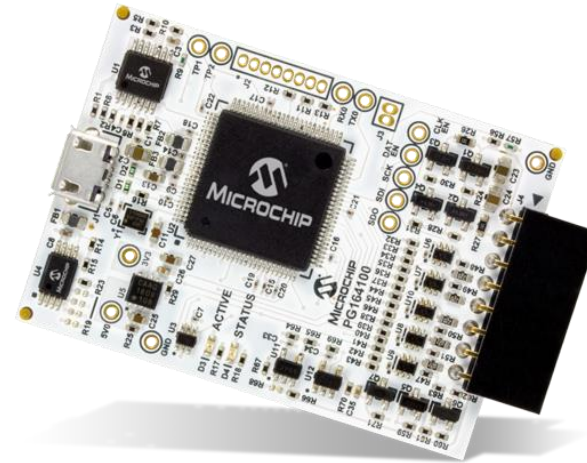
\$76.99



# 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



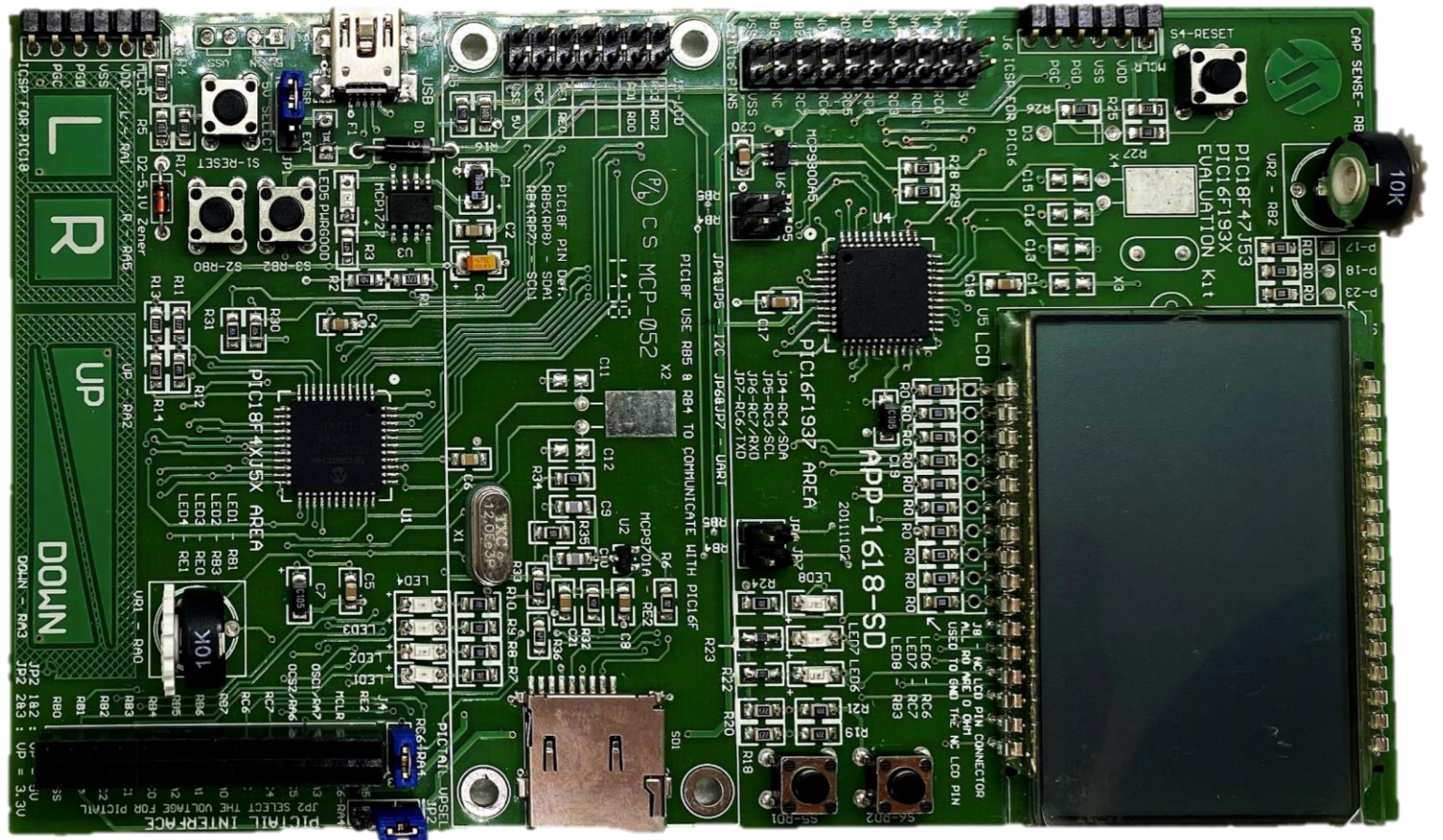
\$34.09

# APP1618-SD EVB Introduction

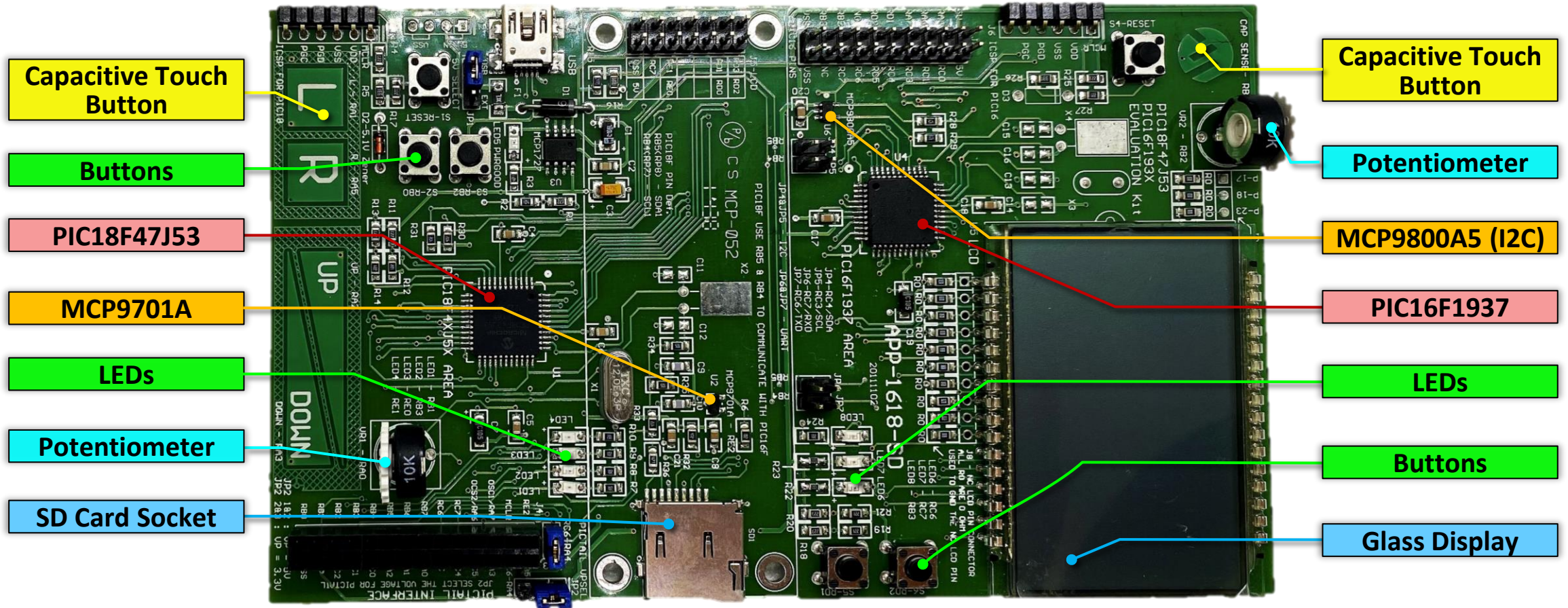
---

# APP1618-SD EVB

- APP1618-SD EVB combine two Microchip MCUs and Built-on rich components.
- **PIC16F1937 block**
  - 3 x LEDs, 2 x Buttons.
  - 1 x Potentiometer.
  - 1 x I2C Temperature Sensor.
  - 1 x Capacitive Touch Buttons.
  - 1 x Glass Display.
- **PIC18F47J53 block**
  - 4 x LEDs.
  - 3 x Buttons.
  - 1 x Potentiometer.
  - 1 x Analog Temperature Sensor .
  - 4 x Capacitive Touch Buttons.
  - 1 x MicroSD Socket



# APP1618-SD Overview



# First MCC Project

---

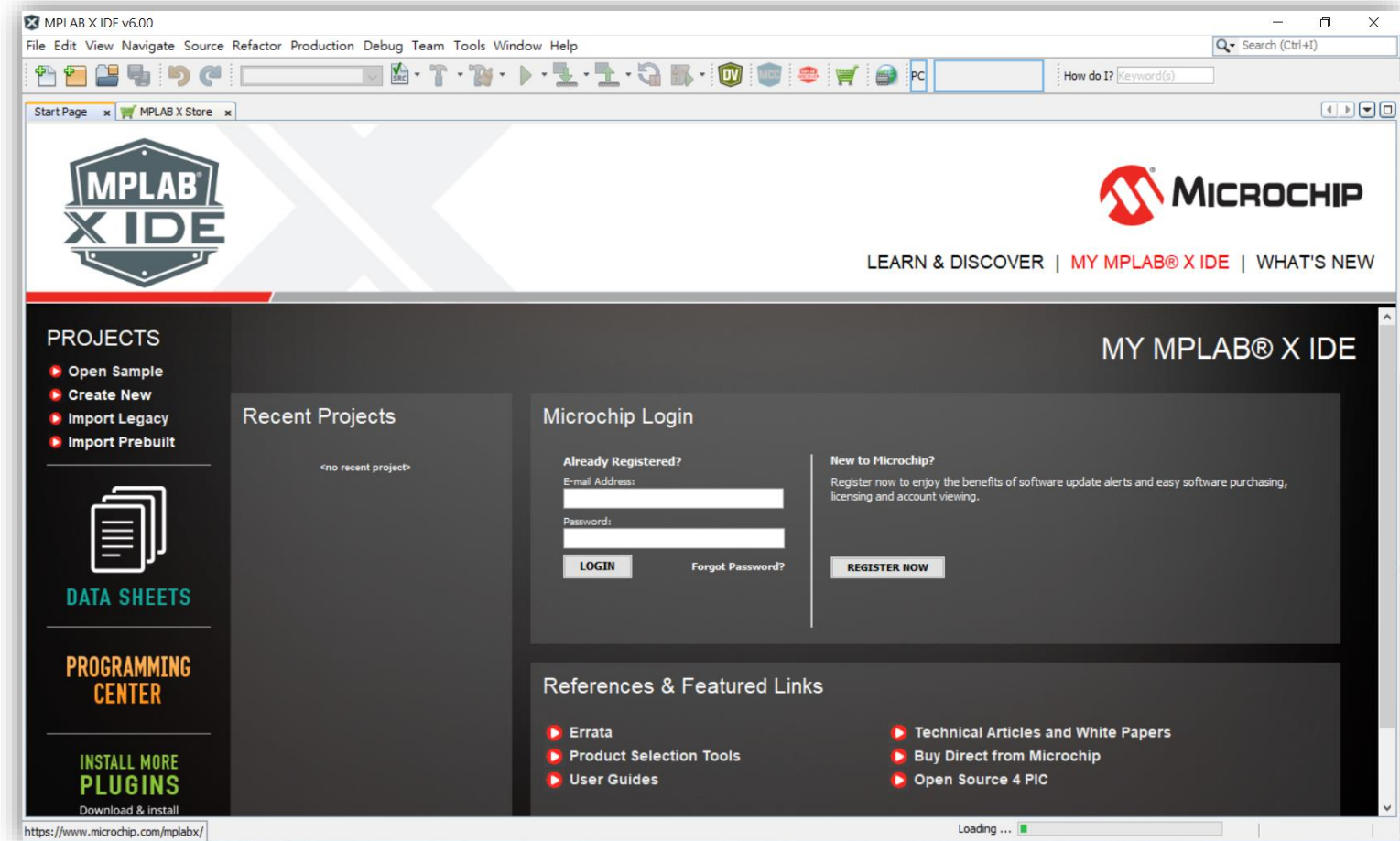
# Lab Preparation

## Step 1

- Open **MPLAB X IDE** firstly.



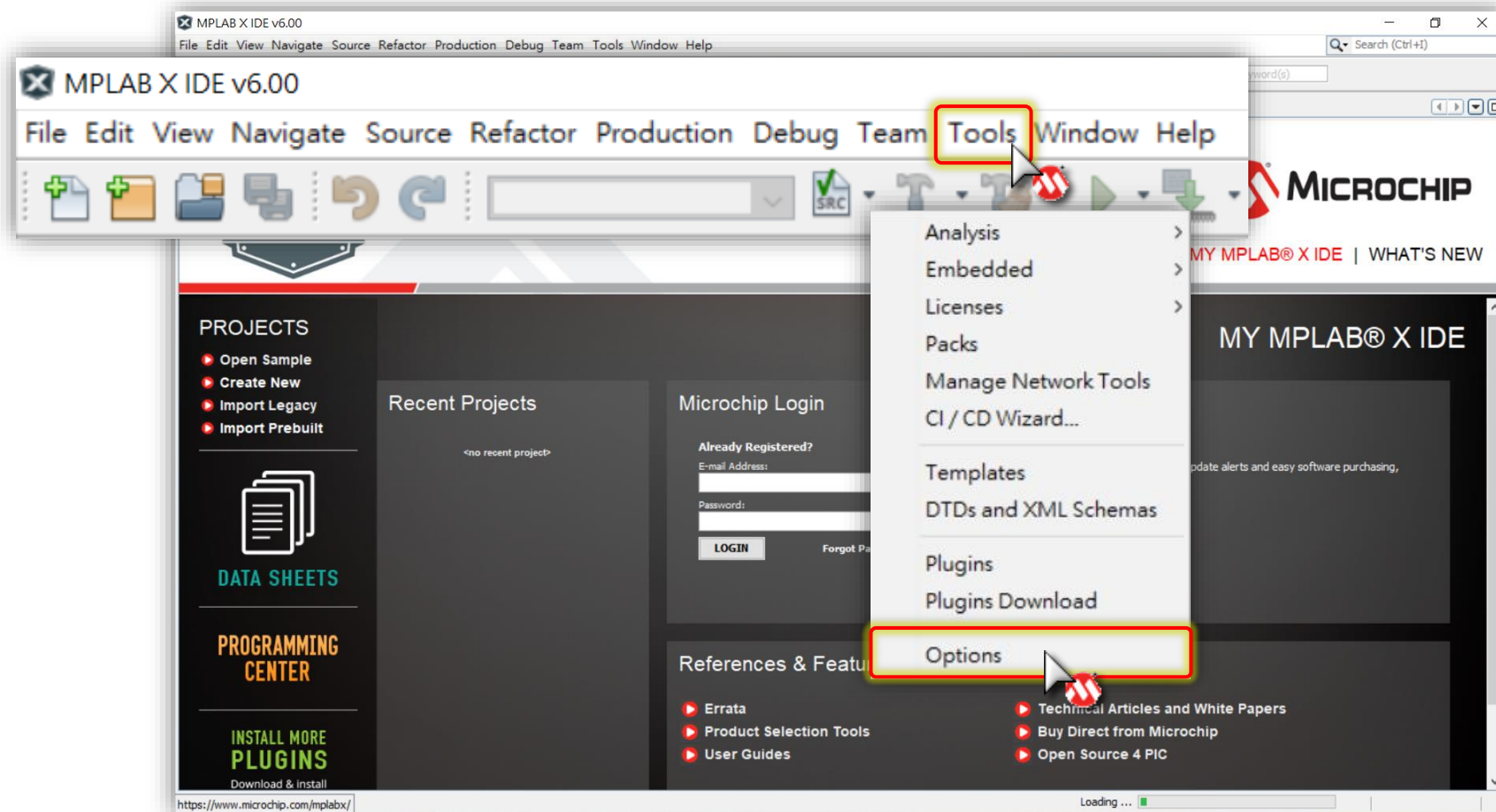
Double Click!



# Lab Preparation

## Step 2

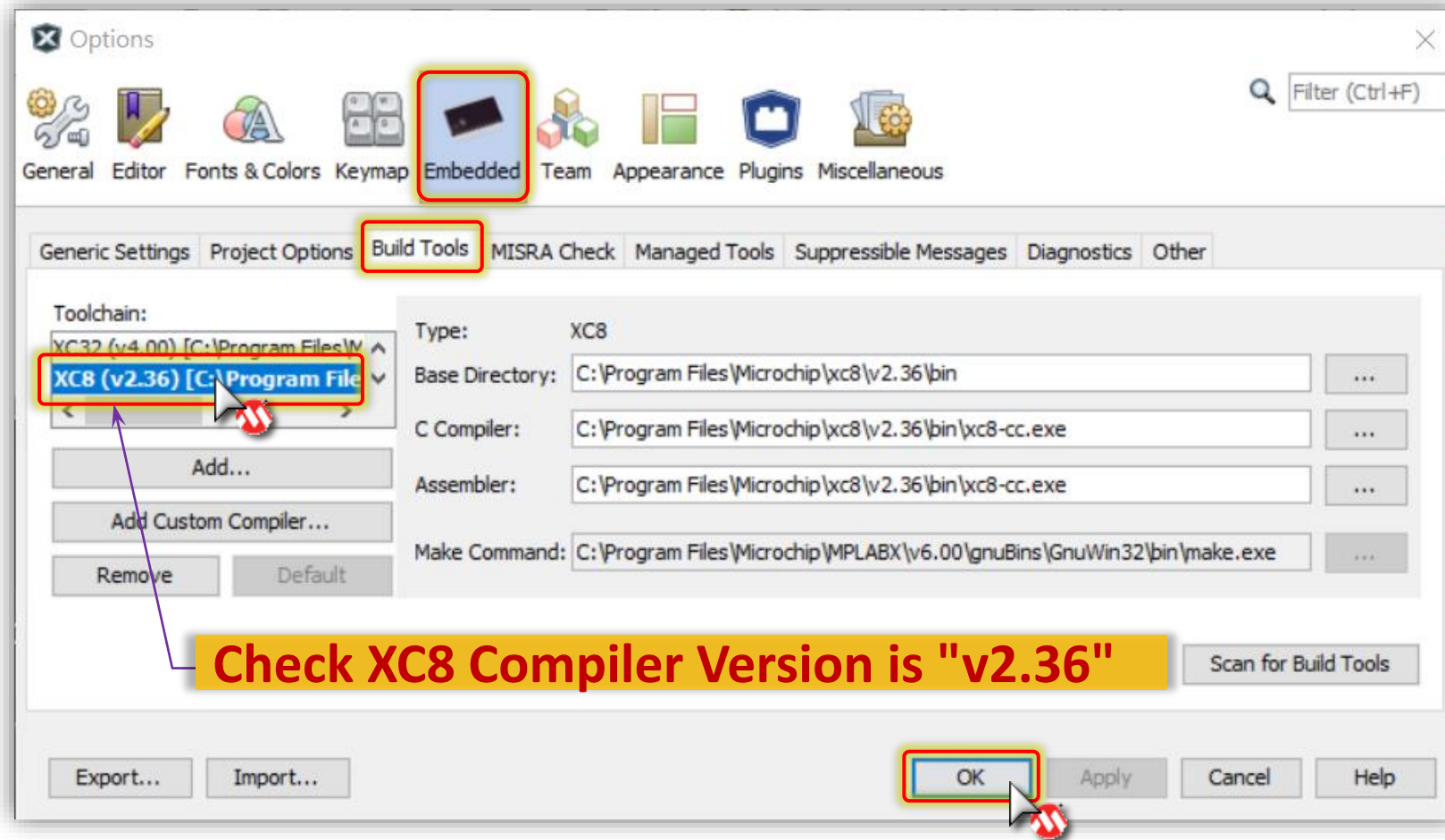
- Select : **Tools ▶ Options**



# Lab Preparation

## Step 3

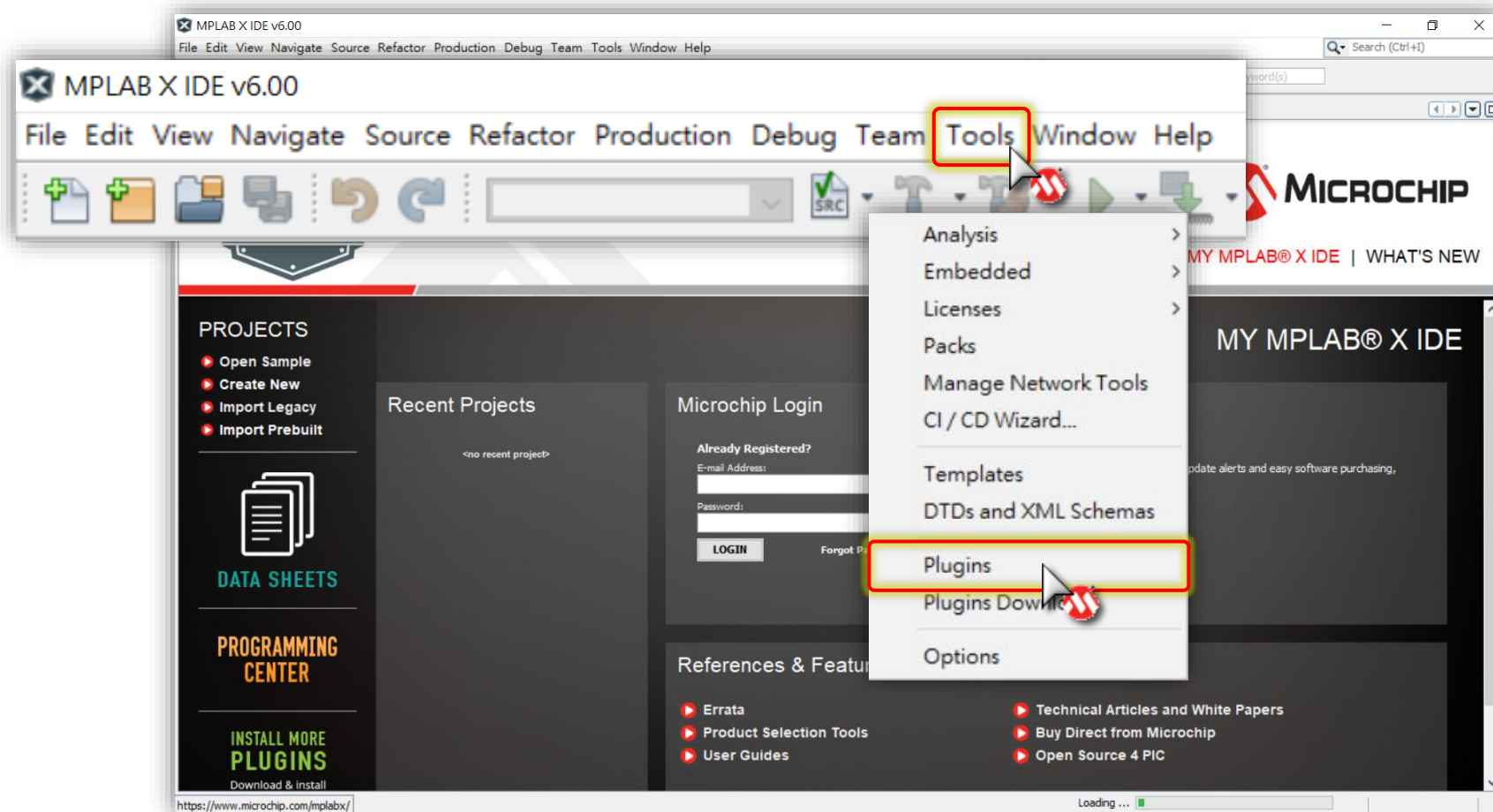
- Select : **Embedded** ► **Build Tools**



# Lab Preparation

## Step 4

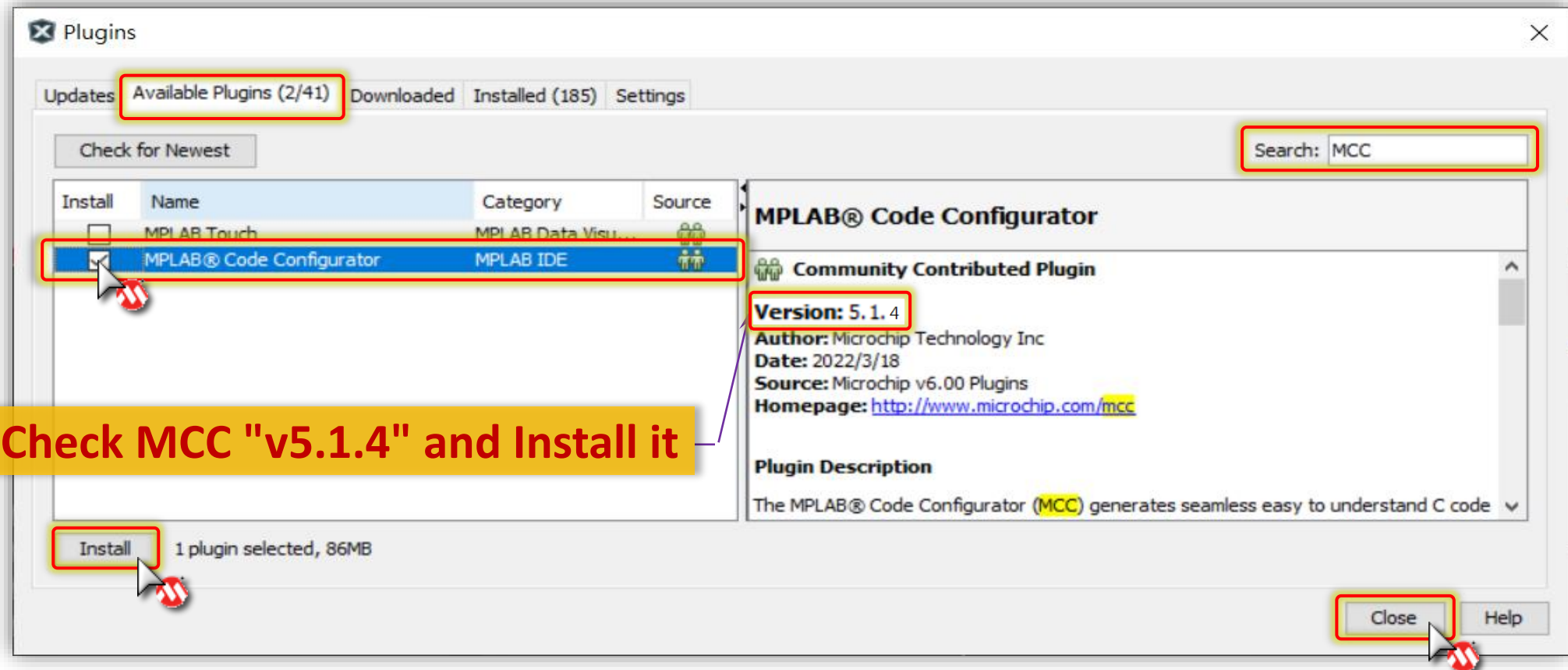
- Select : **Tools ▶ Plugins**



# Lab Preparation

## Step 5 (MCC not installed yet)

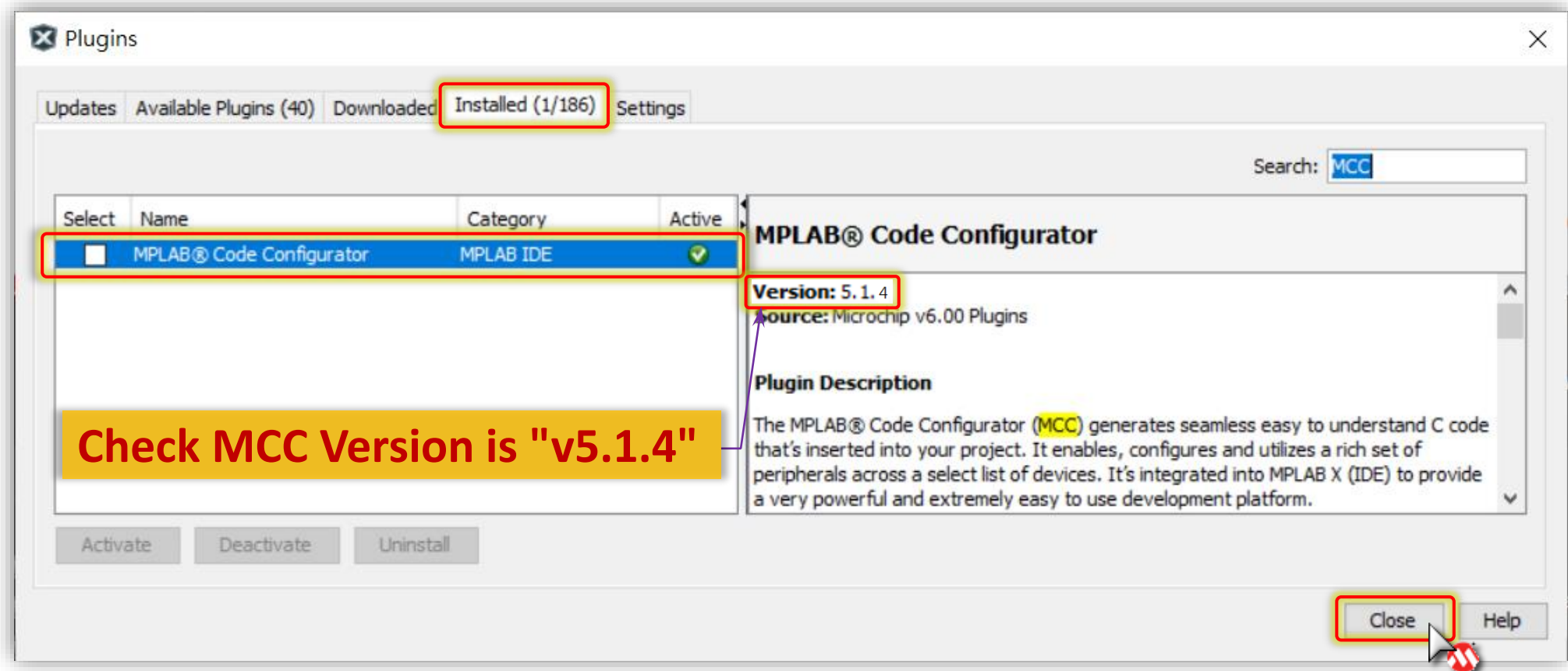
- Select : **Available Plugins**
- Search : **MCC**



# Lab Preparation

## Step 5 (MCC has installed)

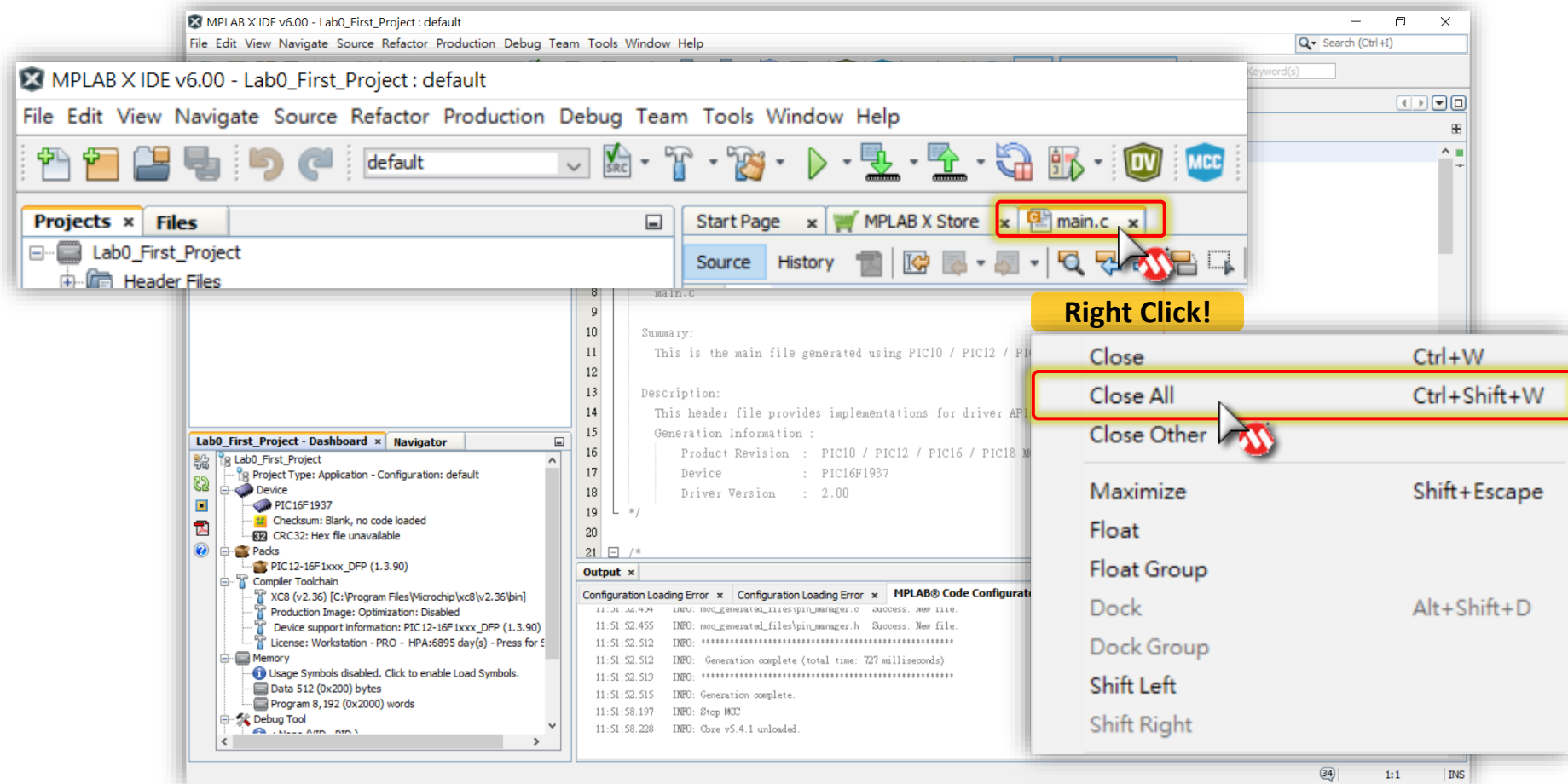
- Select : **Installed**
- Search : **MCC**



# Lab Preparation

## Step 6

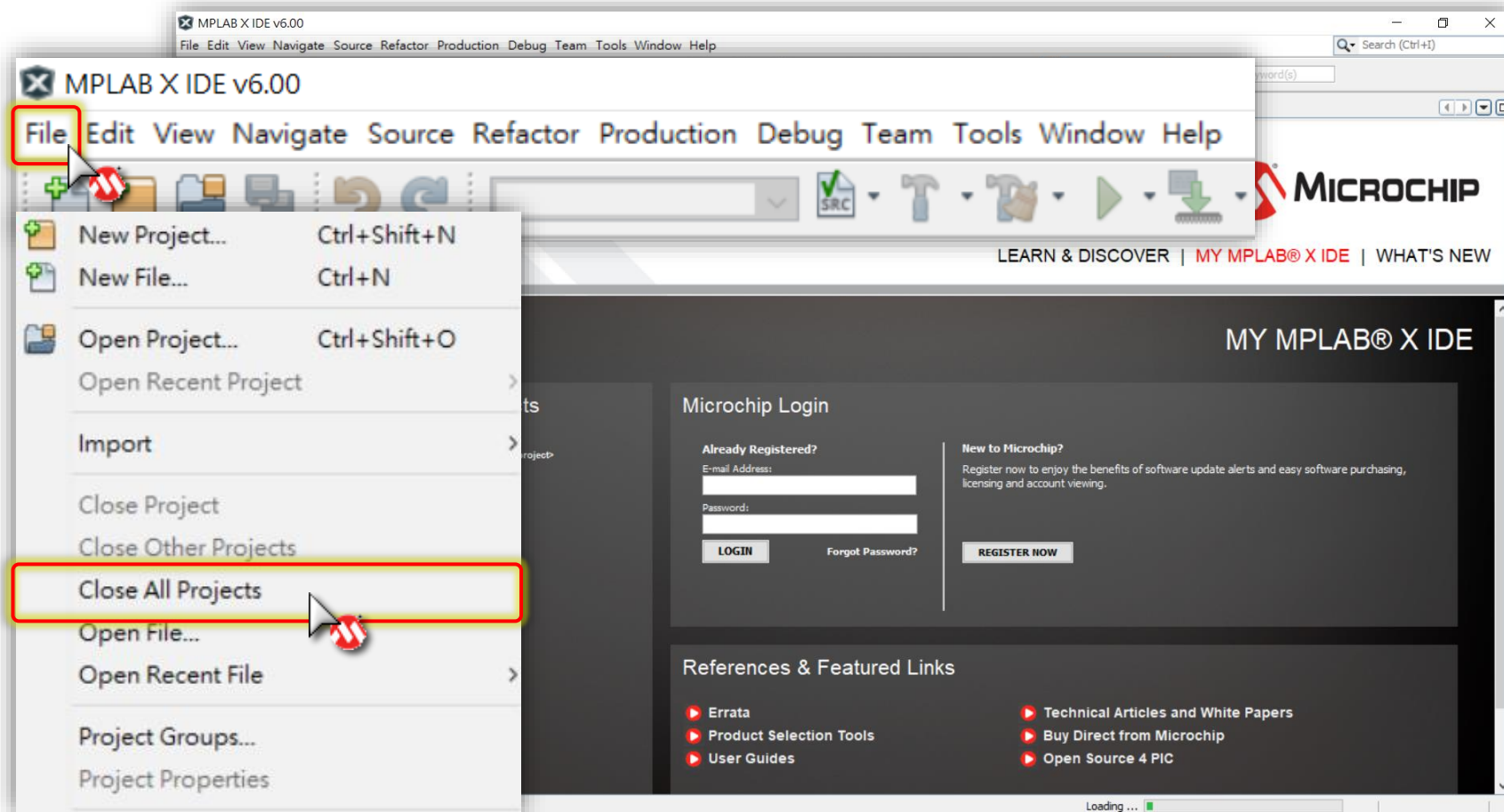
- To prevent opened files to confuse our implementation.
- Please **[Right Click]** on opened file and choose **Close All**.



# Lab Preparation

## Step 7

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



# Lab0 First Project

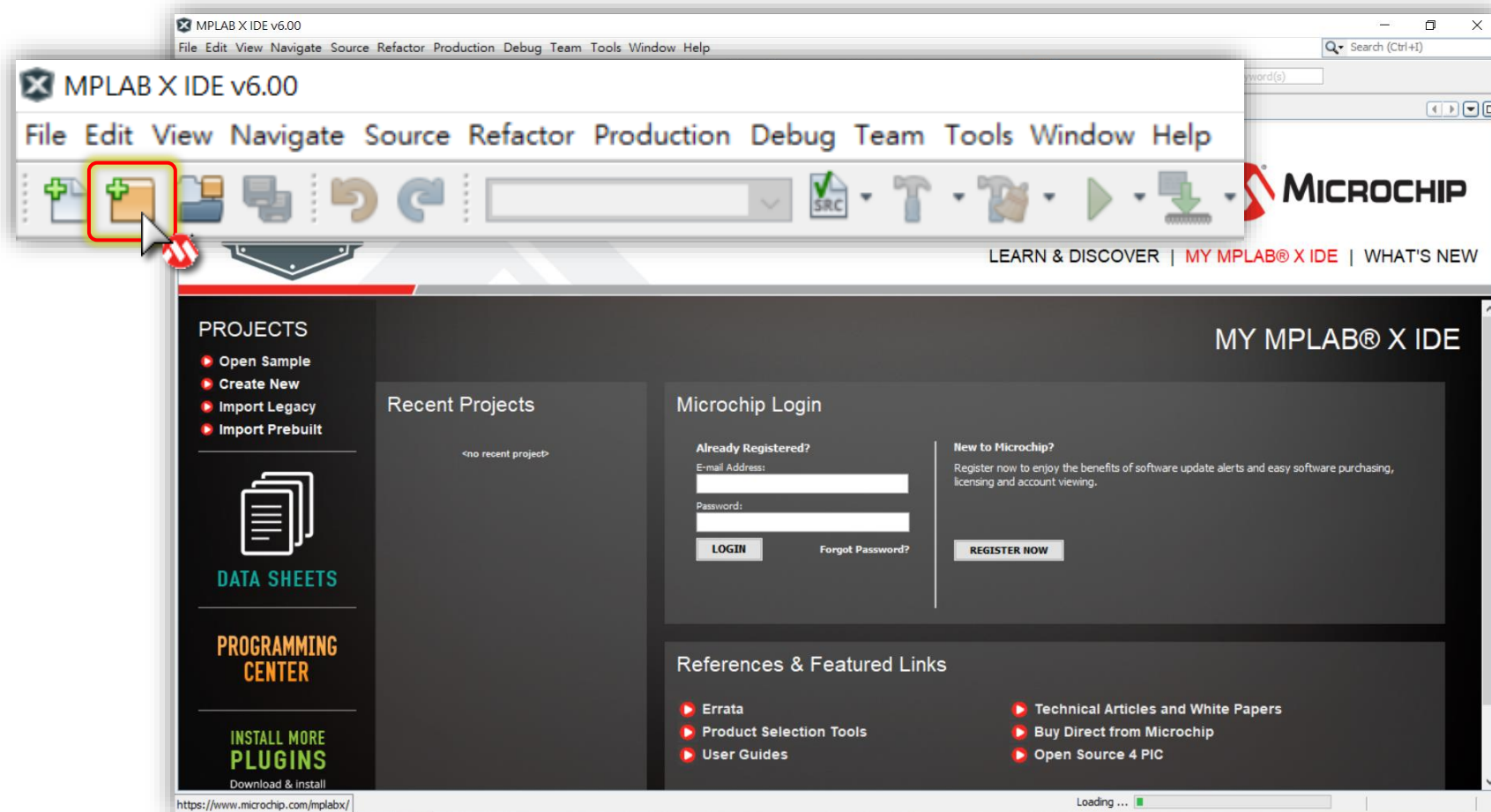
- Try to create your first MPLAB X IDE Project.
- Try to create your first MCC style Project.
- To understanding MPLAB X IDE basic operation.
- Learn how to use edit, build, project manager and program functions.

## How to start ?

# Lab0 First Project

## Step 1

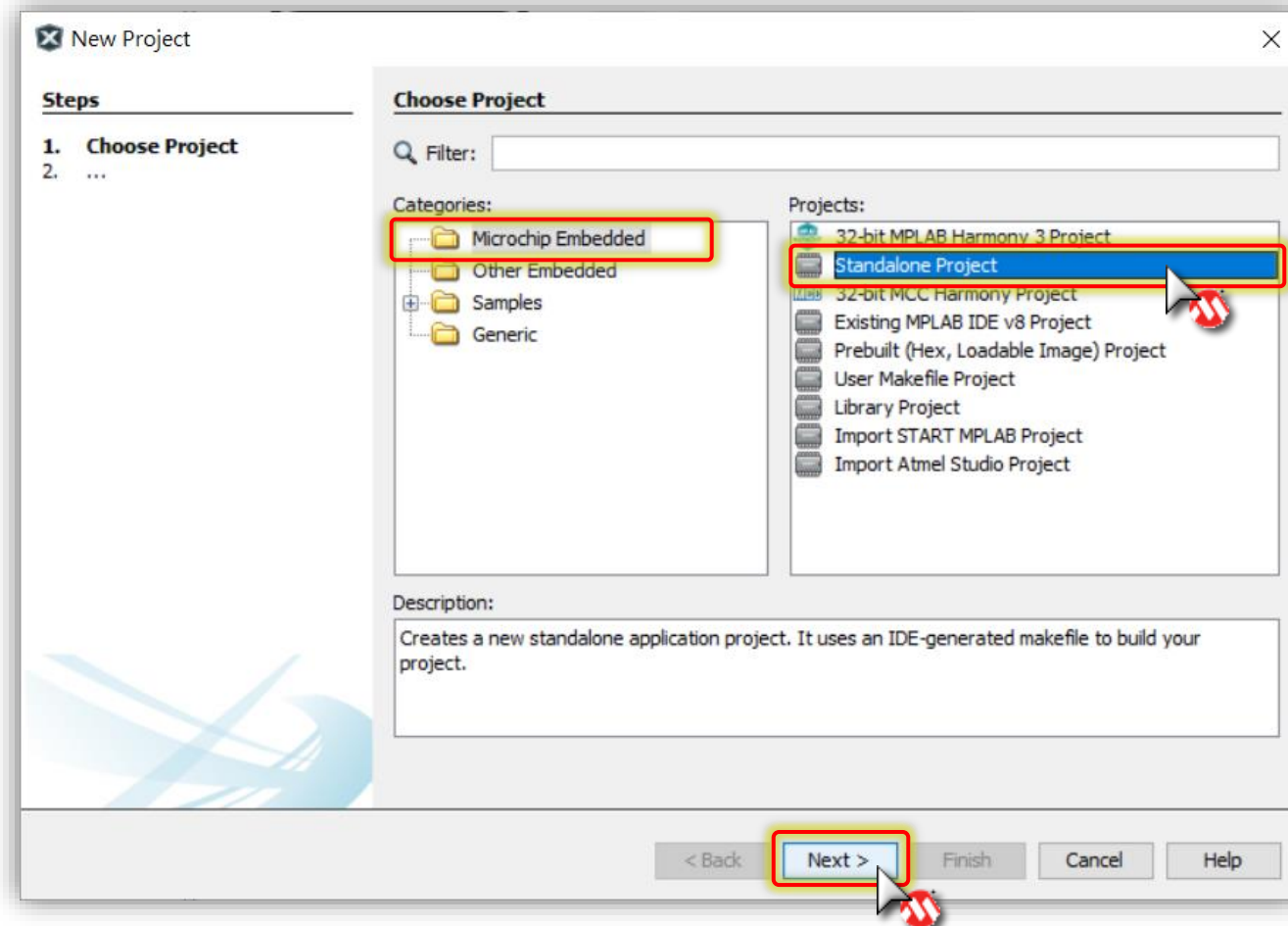
- Execute Project Wizard Select : **File ▶ New Project** or Click **icon** 



# Lab0 First Project

## Step 2

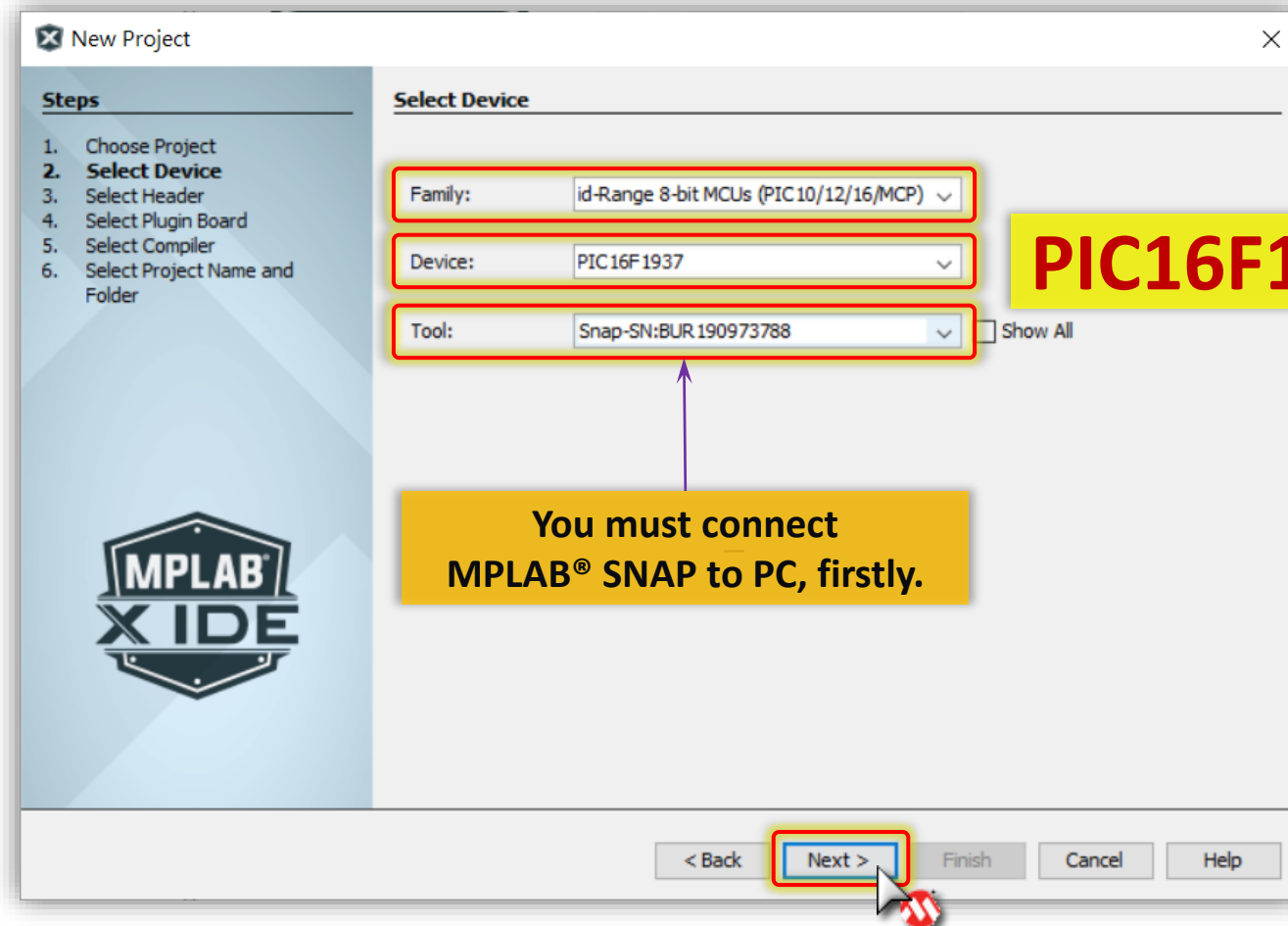
- Categories : **Microchip Embedded**
- Projects : **Standalone Project**



# Lab0 First Project

## Step 3

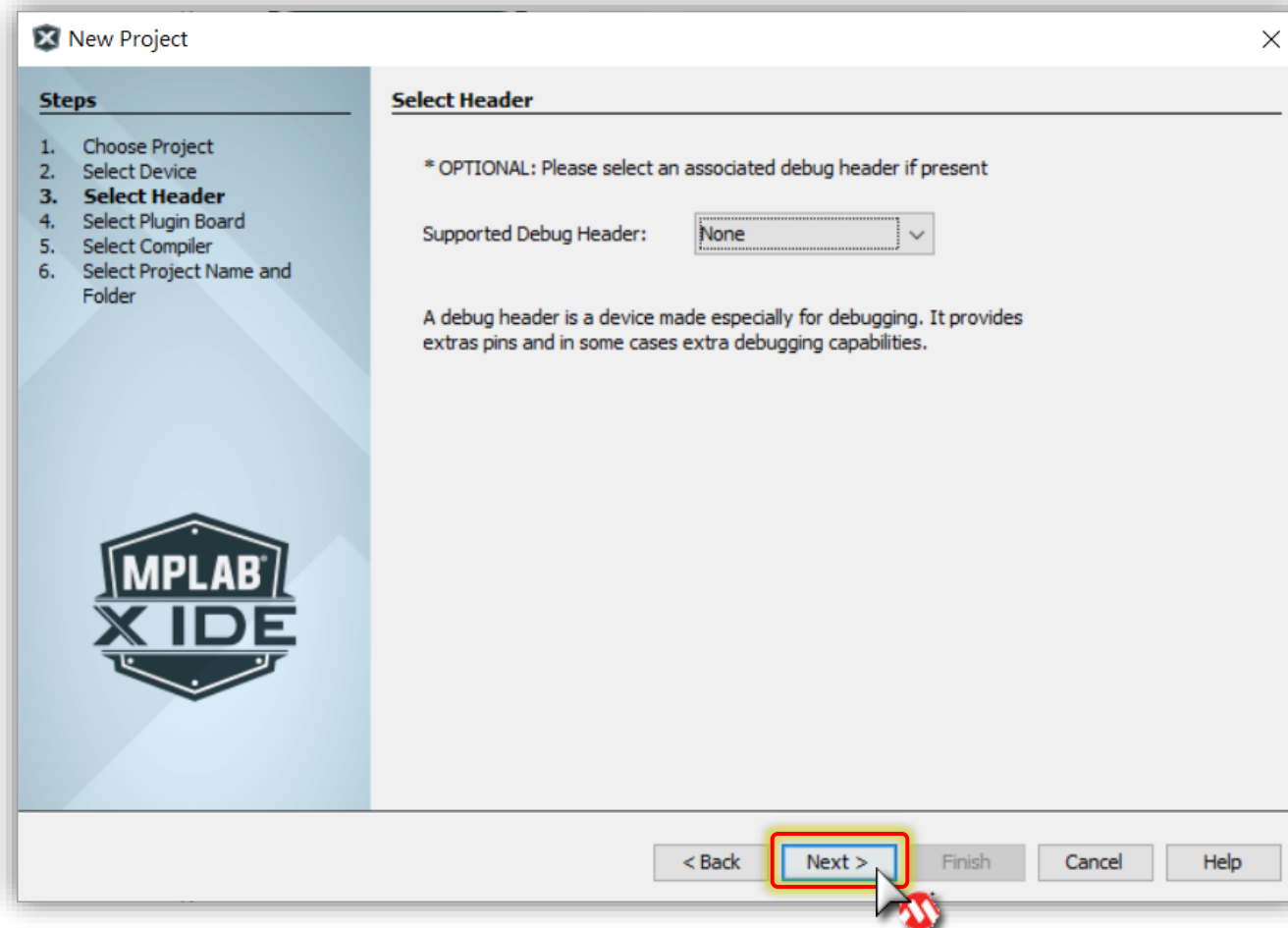
- Family : **Mid-Range 8-bit MCUs (PIC10/12/16/MCP)** , Device : **PIC16F1937**
- Tool : **Snap-SN : BURXXXXXXXXXX**



# Lab0 First Project

## Step 4

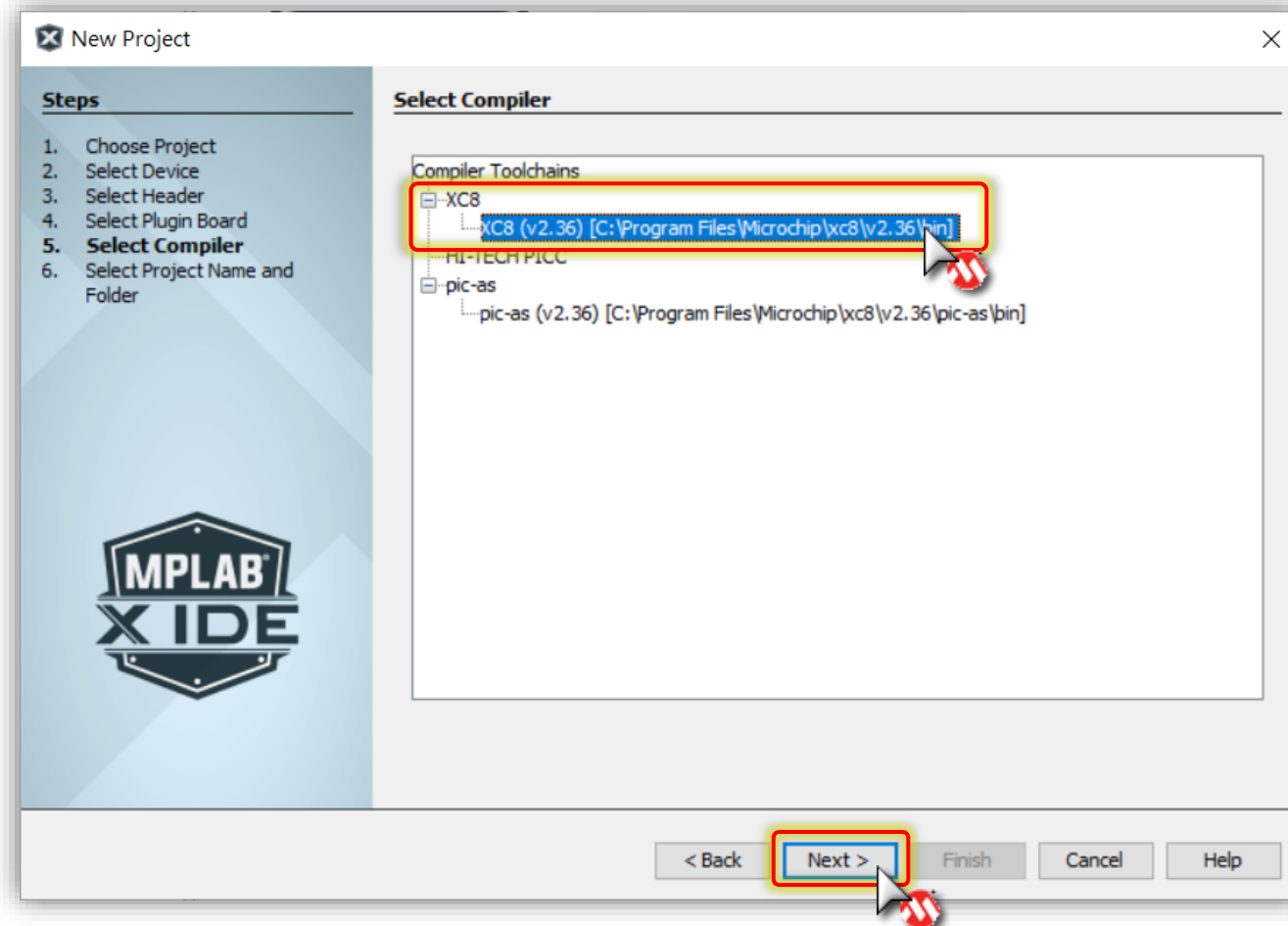
- Supported Debug Header : **None**



# Lab0 First Project

## Step 5 (Use Bare metal C and MCC)

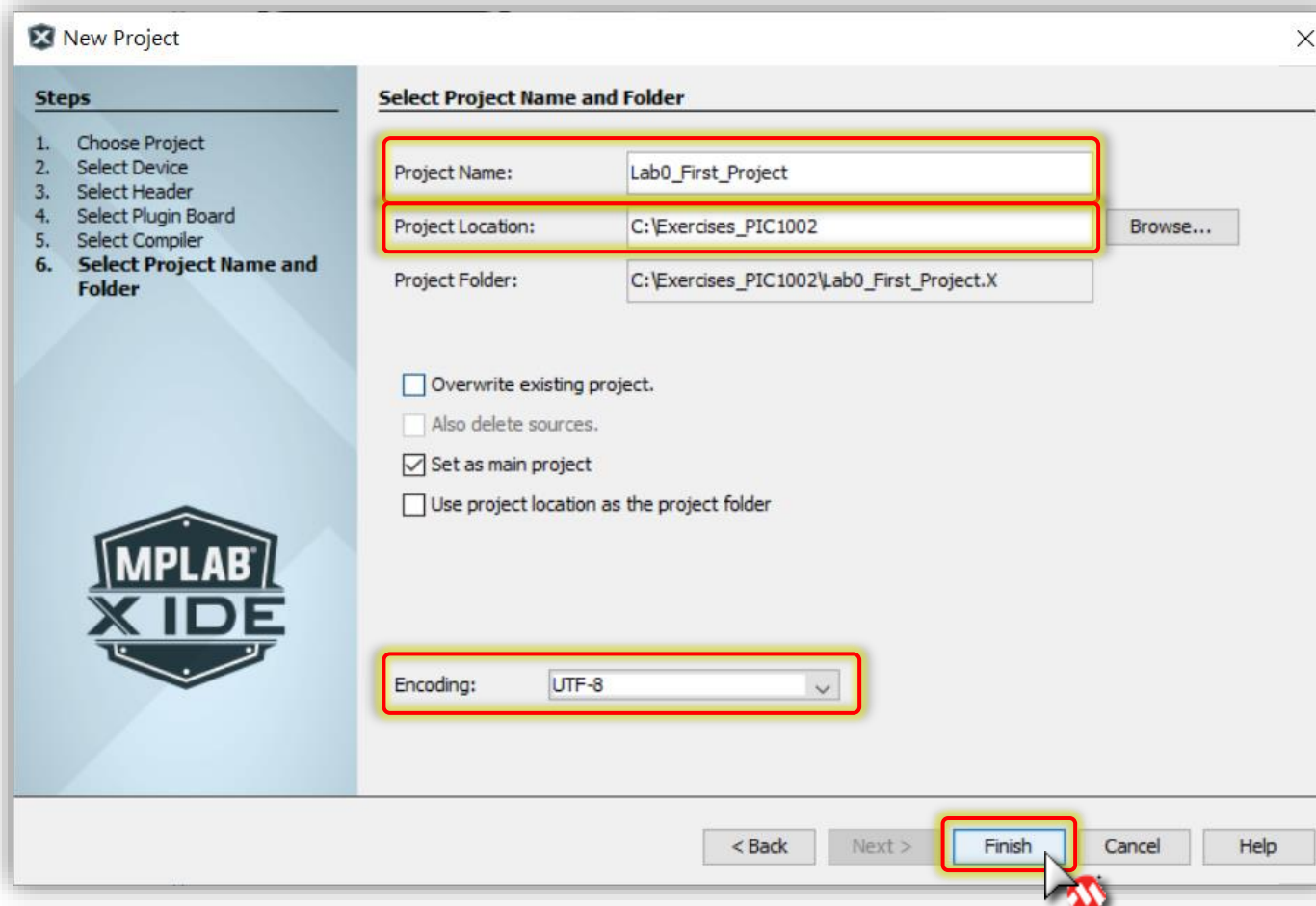
- **Compiler Toolchains : XC8 (v2.36) [C:\Program Files\Microchip\xc8\v2.36\bin]**



# Lab0 First Project

## Step 6

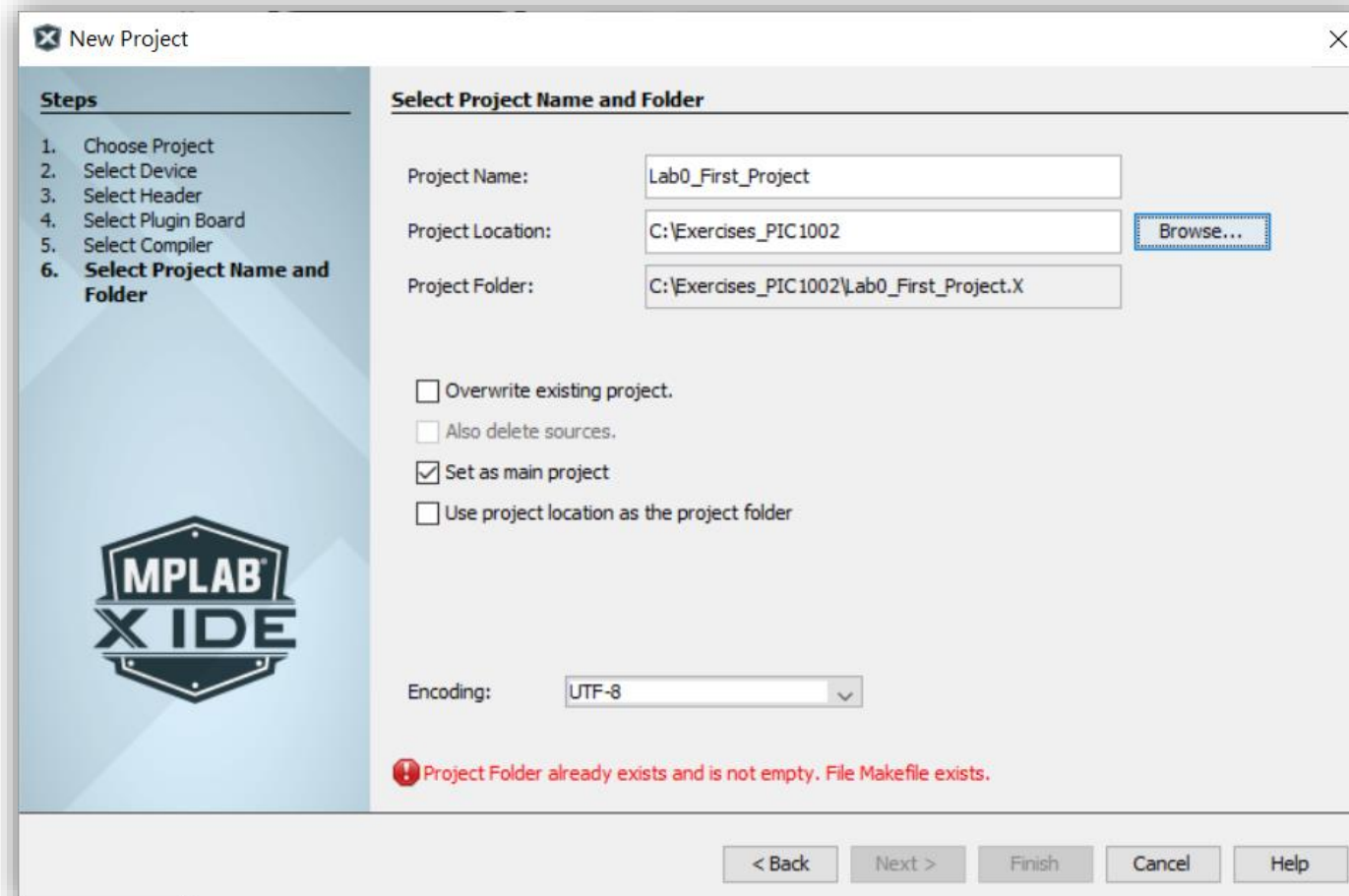
- Project Name : **Lab0\_First\_Project** , Project Location : **C:\Exercises\_PIC1002**
- Encoding : **UTF-8**



# 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.



**New Project**

**Steps**

1. Choose Project
2. Select Device
3. Select Header
4. Select Plugin Board
5. Select Compiler
6. **Select Project Name and Folder**

**Select Project Name and Folder**

Project Name:

Project Location:

Project Folder:

☐ Overwrite existing project.

☐ Also delete sources.

☒ Set as main project

☐ Use project location as the project folder

Encoding:

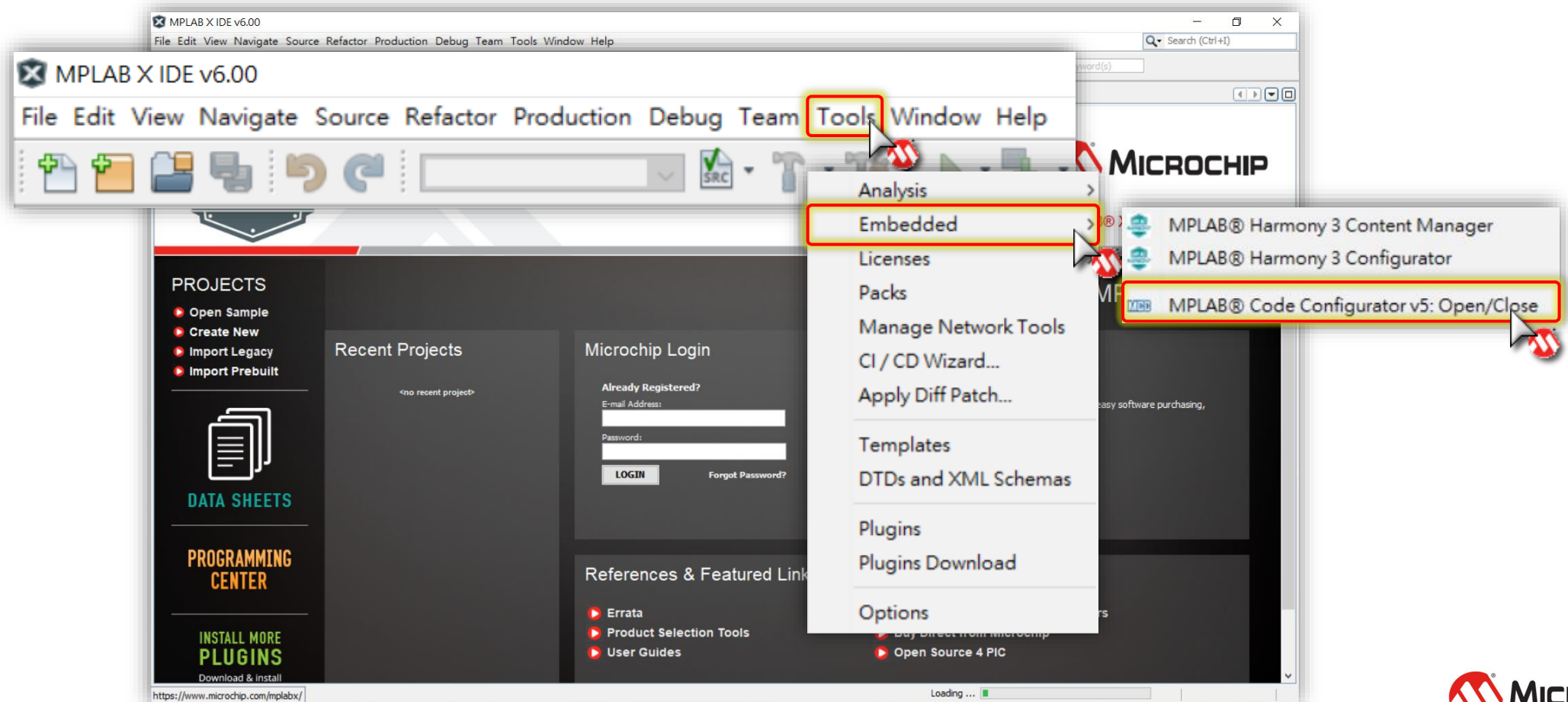
 Project Folder already exists and is not empty. File Makefile exists.

< Back   Next >   Finish   Cancel   Help

# Lab0 First Project

## Step 7

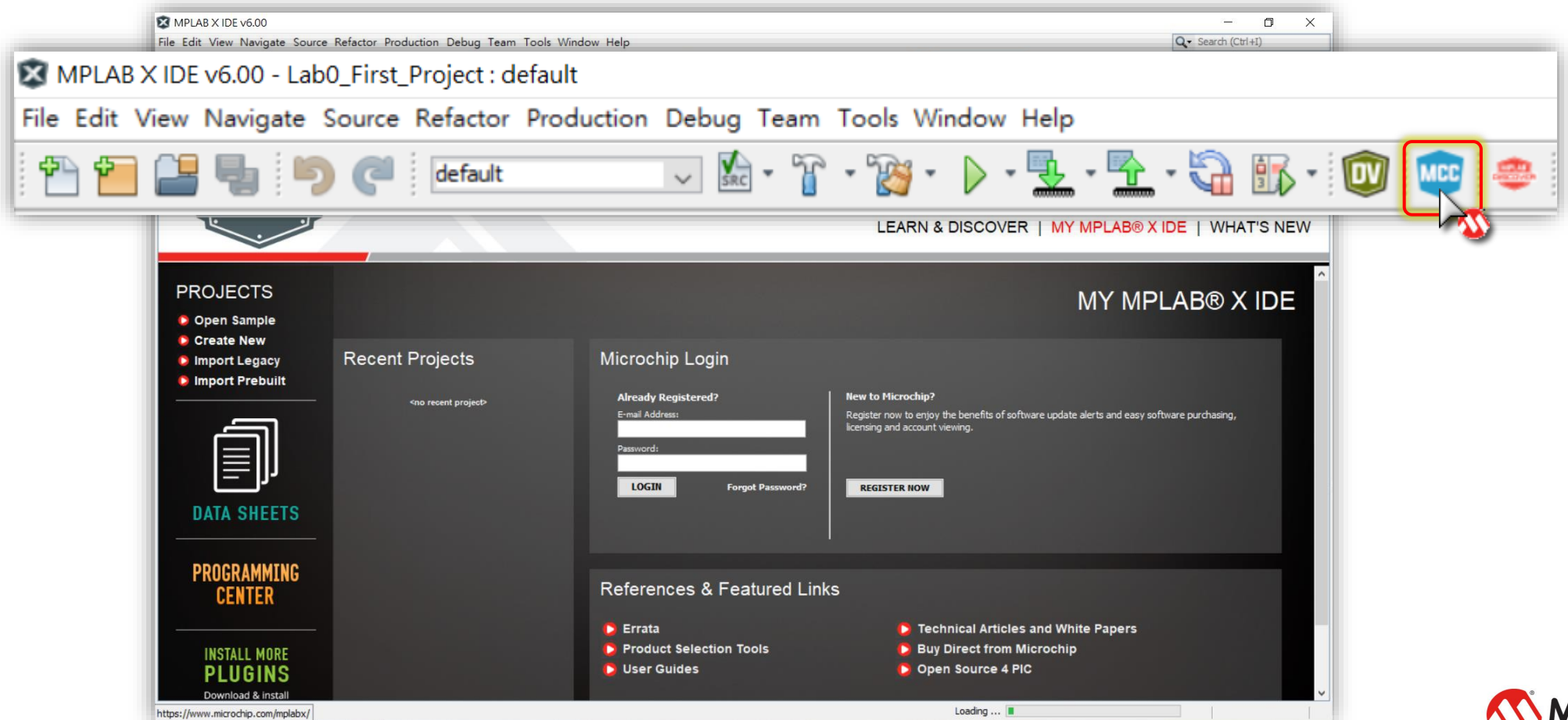
- Menu **Tools** ► **Embedded** ► **MPLAB Code Configurator v5: Open/Close** or Click



# Lab0 First Project

## Step 7

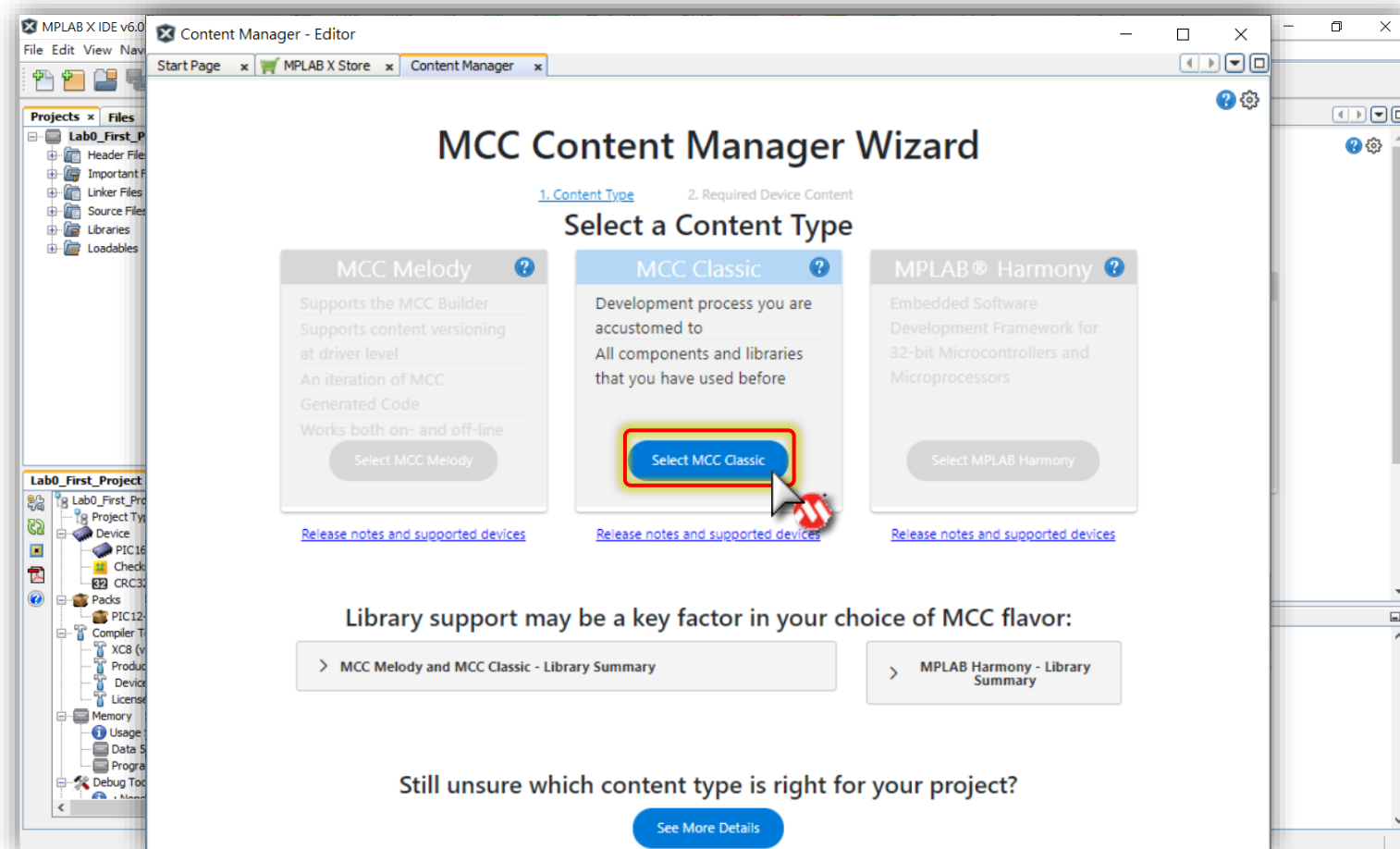
- Menu **Tools** ► **Embedded** ► **MPLAB Code Configurator v5: Open/Close** or **Click**



# Lab0 First Project

## Step 8

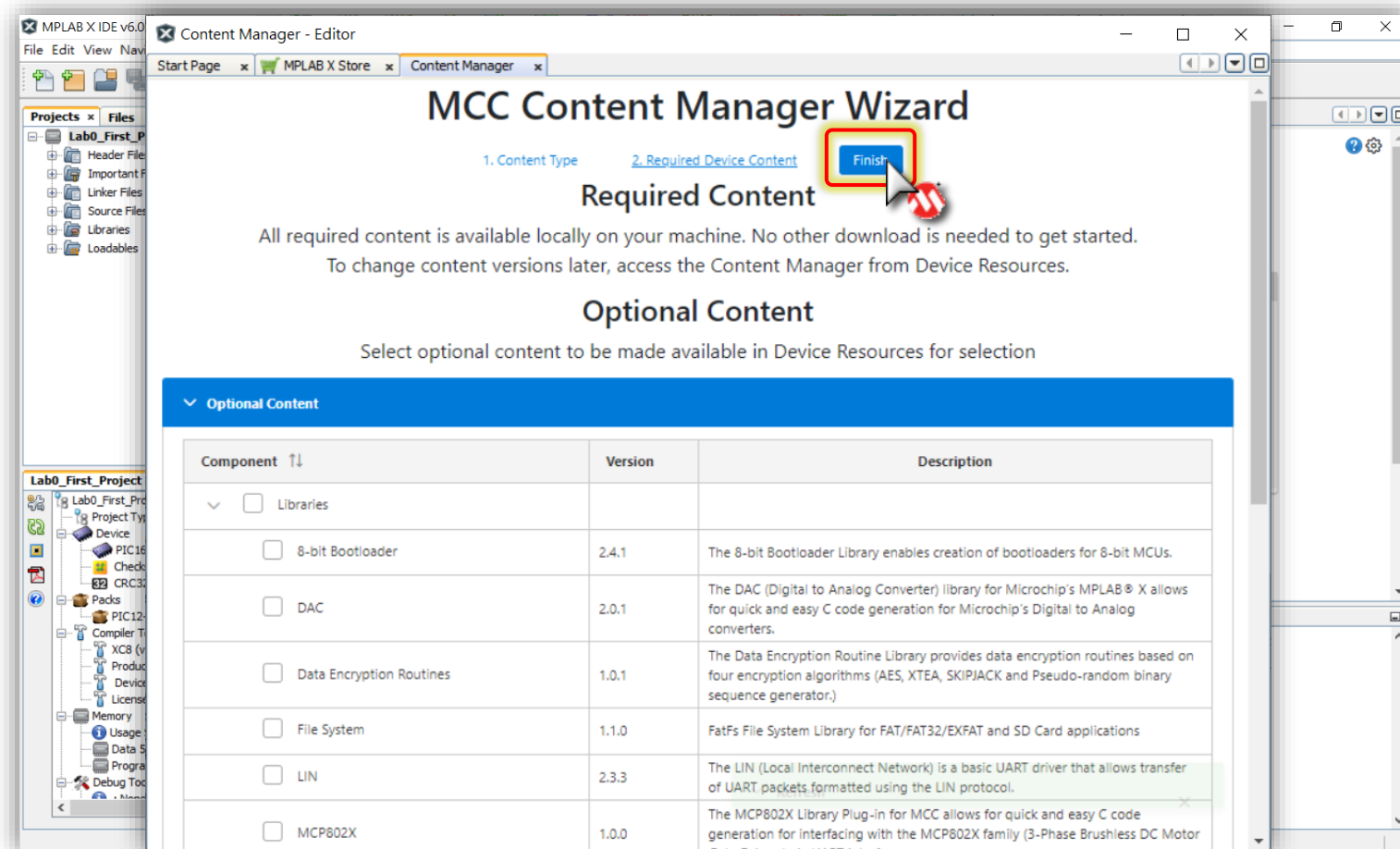
- Select a Content Type : **Select MCC Classic.**



# Lab0 First Project

## Step 9

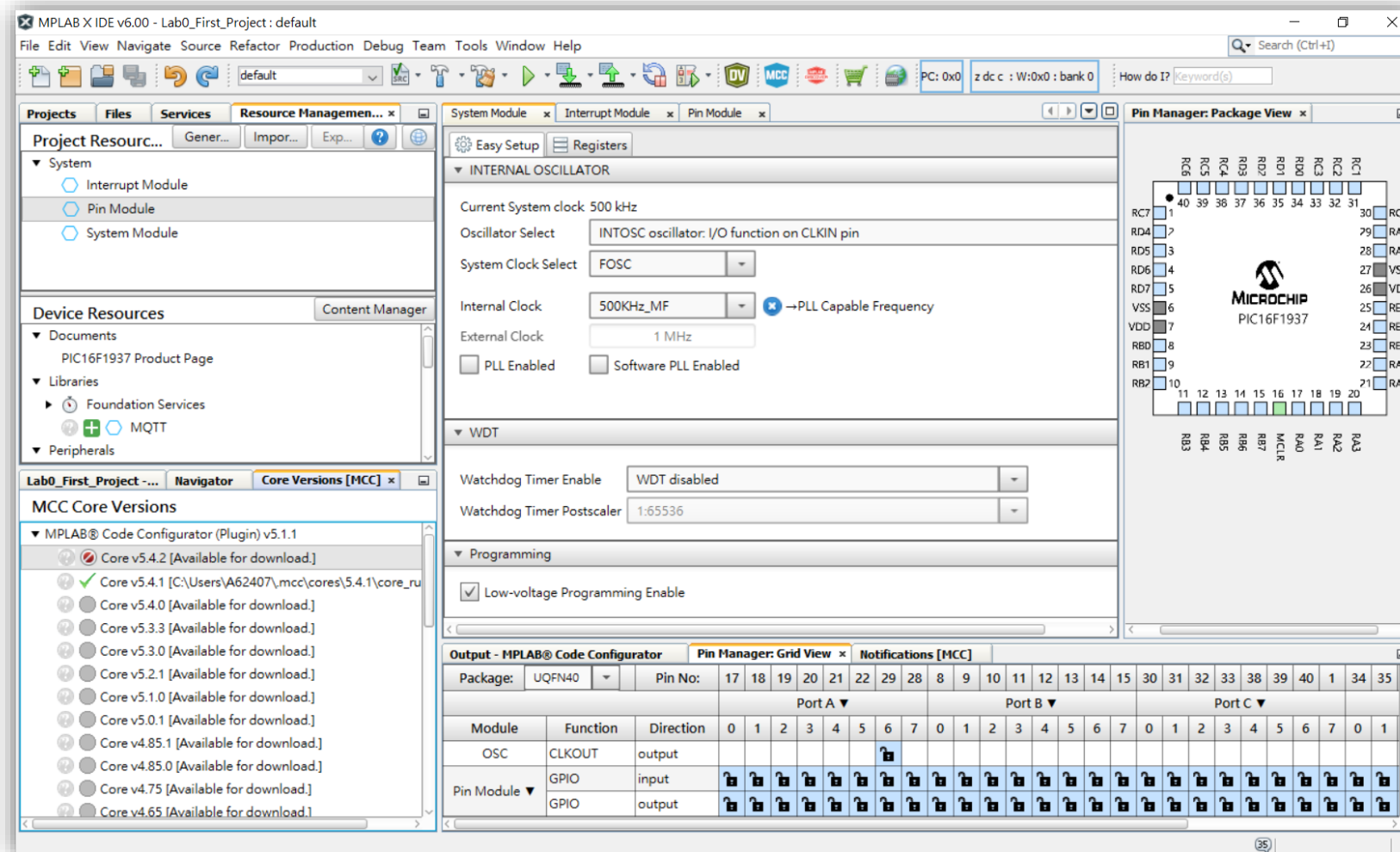
- You can add the available Libraries provided in the Device resource to the project or click **Finish**.



# Lab0 First Project

## Step 10

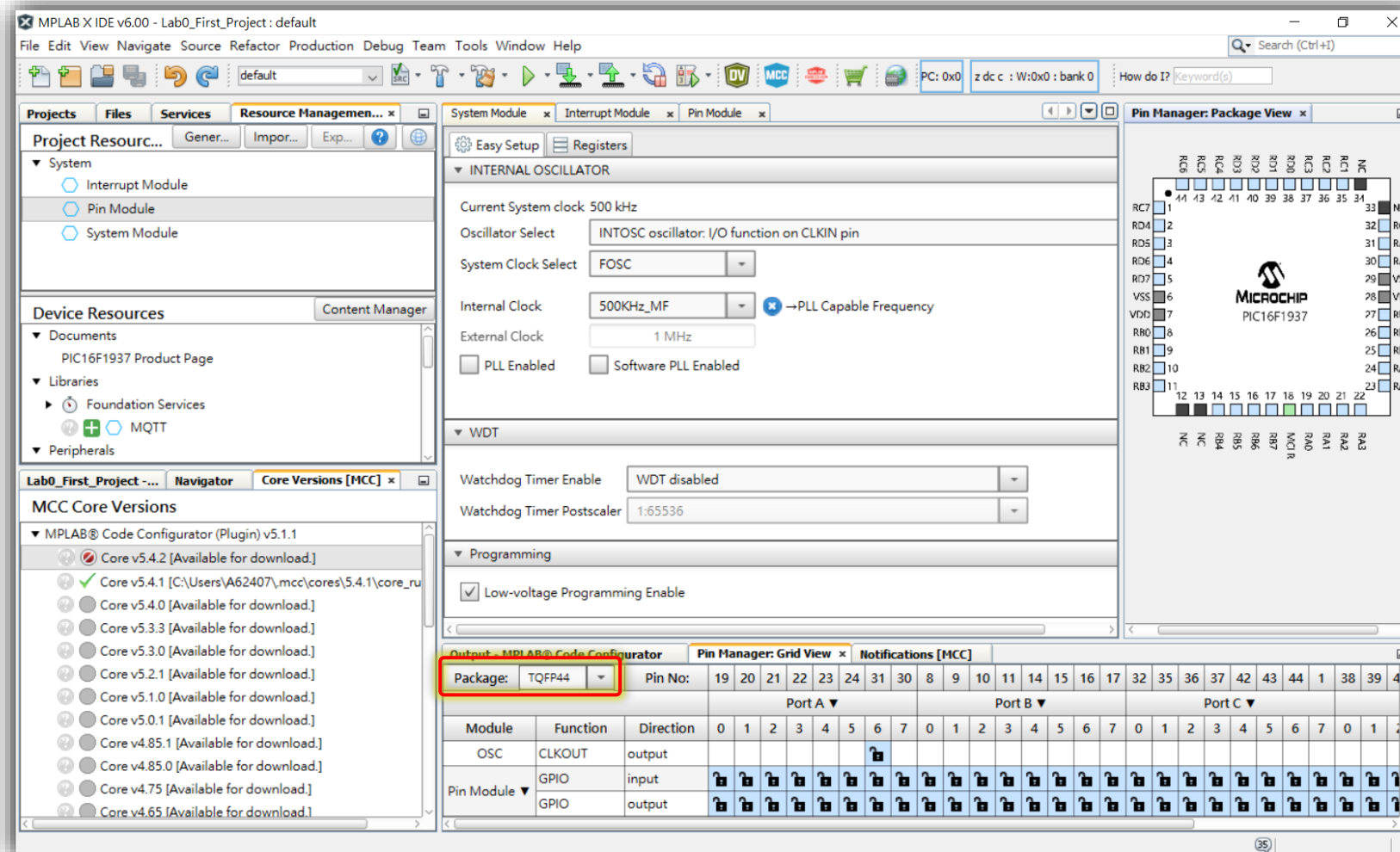
- MCC Configurator interface show up.



# Lab0 First Project

## Step 11

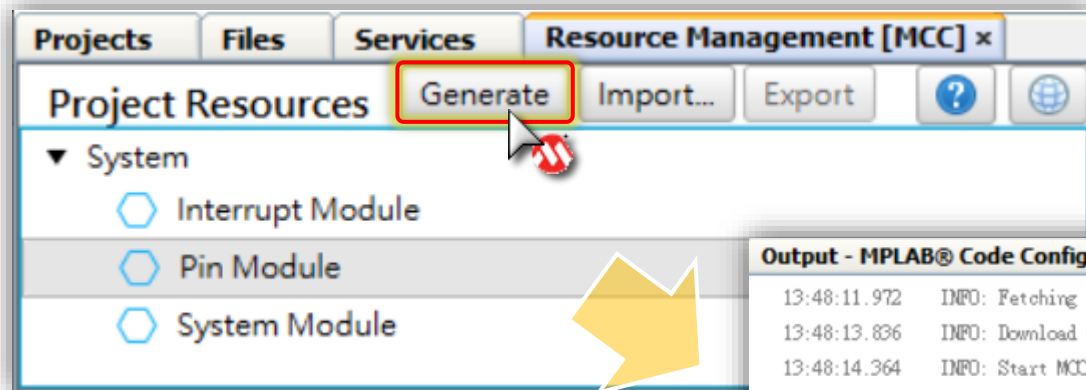
- Change package : **UQFN40** ► **TQFP44**



# Lab0 First Project

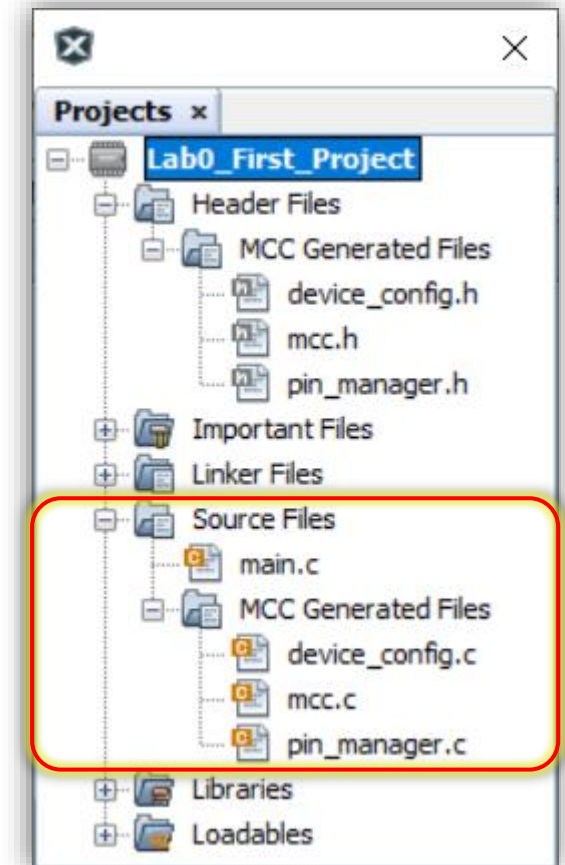
## Step 12

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



Output - MPLAB® Code Configurator

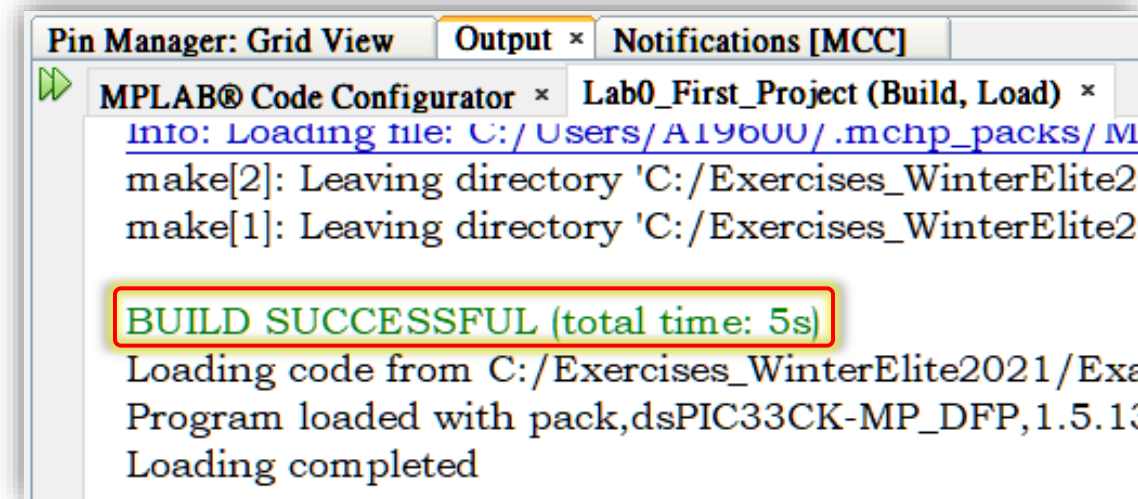
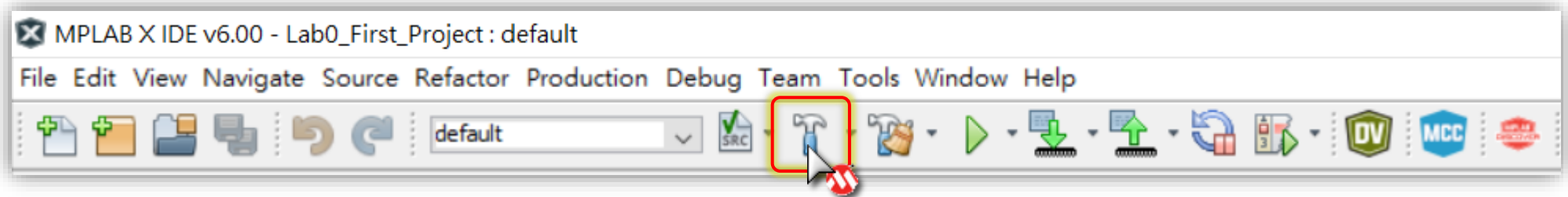
```
13:48:11.972 INFO: Fetching list of available libraries.
13:48:13.836 INFO: Download Complete: C:\Users\A62407\.mcc\mcc_libraries.xml
13:48:14.364 INFO: Start MCC
13:48:14.383 INFO: Core v5.4.1 loaded.
14:07:22.545 INFO: *****
14:07:22.546 INFO: Generation Results
14:07:22.547 INFO: *****
14:07:22.555 INFO: main.c Success. New file.
14:07:22.556 INFO: mcc_generated_files\device_config.c Success. New file.
14:07:22.558 INFO: mcc_generated_files\device_config.h Success. New file.
14:07:22.559 INFO: mcc_generated_files\mcc.c Success. New file.
14:07:22.559 INFO: mcc_generated_files\mcc.h Success. New file.
14:07:22.560 INFO: mcc_generated_files\pin_manager.c Success. New file.
14:07:22.561 INFO: mcc_generated_files\pin_manager.h Success. New file.
14:07:22.626 INFO: *****
14:07:22.627 INFO: Generation complete (total time: 928 milliseconds)
14:07:22.628 INFO: *****
14:07:22.632 INFO: Generation complete.
```



# Lab0 First Project

## Step 13

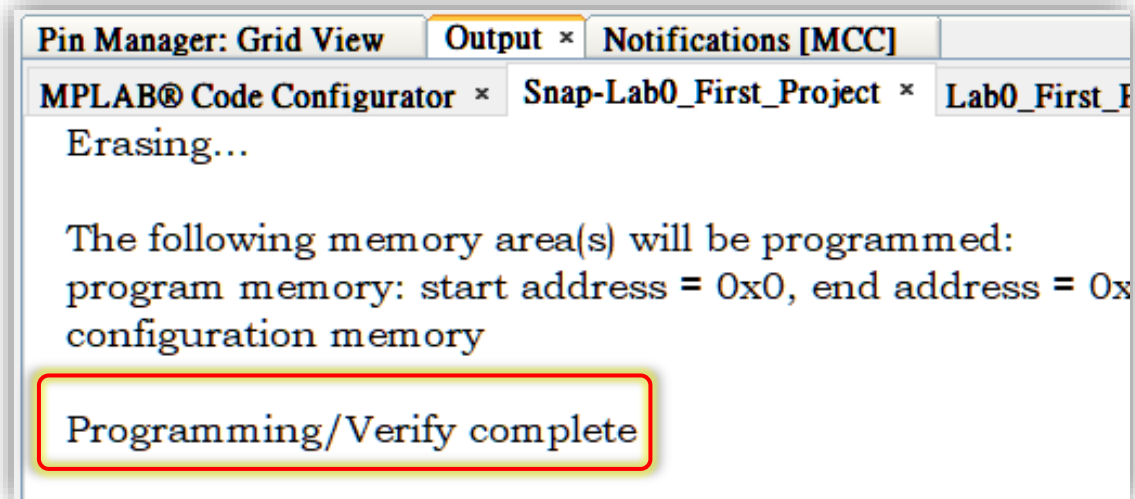
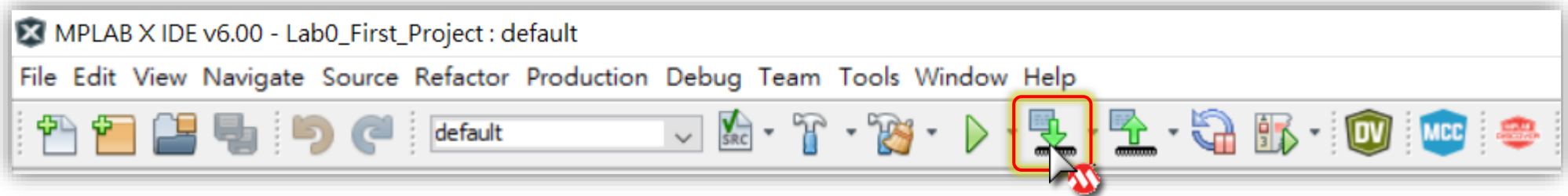
- Select **Build Main Project** icon 
- Make sure compiled result is **BUILD SUCCESSFUL**.



# Lab0 First Project

## Step 14

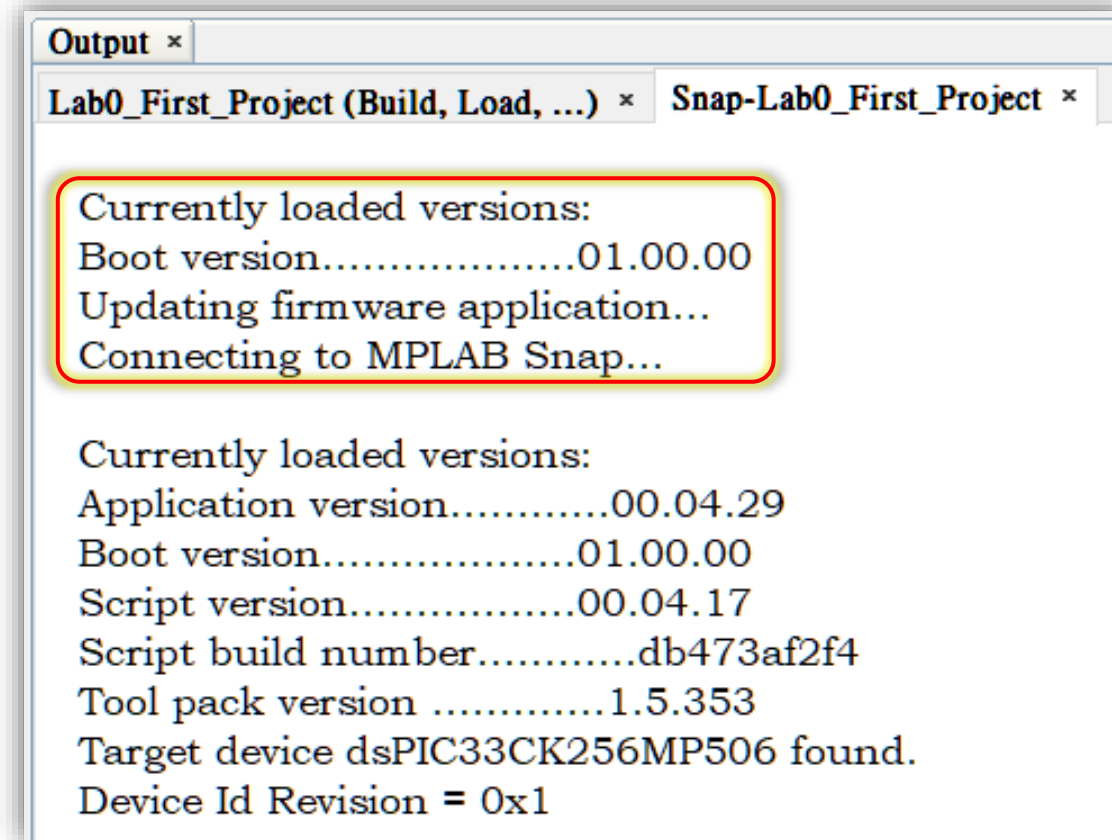
- Select **Make & Program Device Main Project** icon 
- Make sure the result is **Programming/Verify complete**.



# Lab0 First Project

## Notice

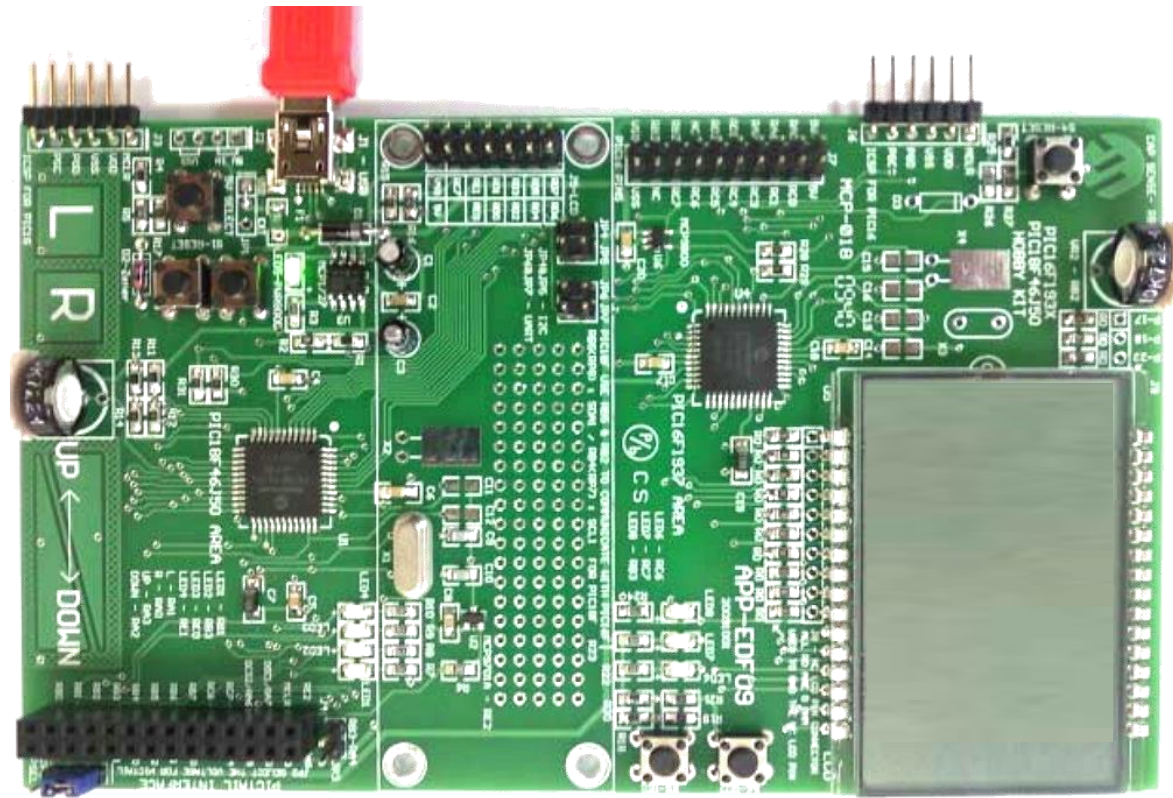
- MPLAB X IDE will auto upgrade the firmware of SNAP to keep it up to date, it might take couple minutes to wait for finish. Your project will start programming after this.



# Lab0 First Project

## Result

Nothing happen ????

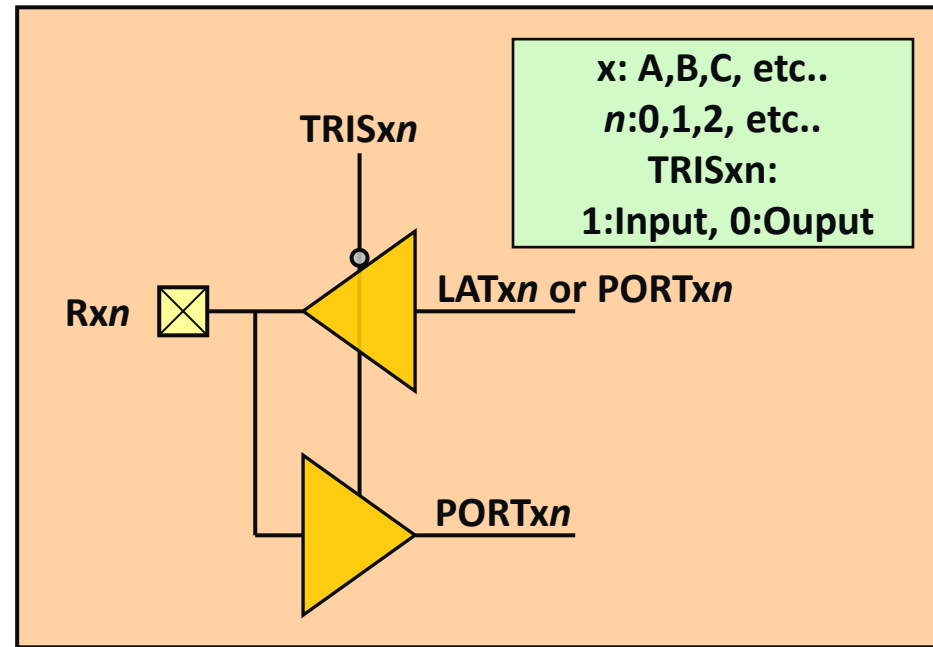
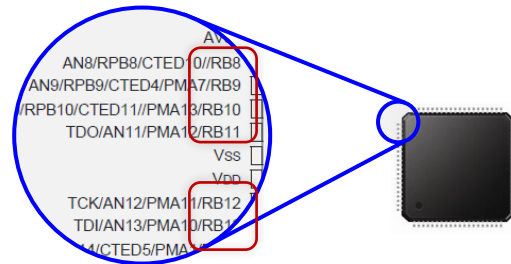


# GPIO Architecture

---

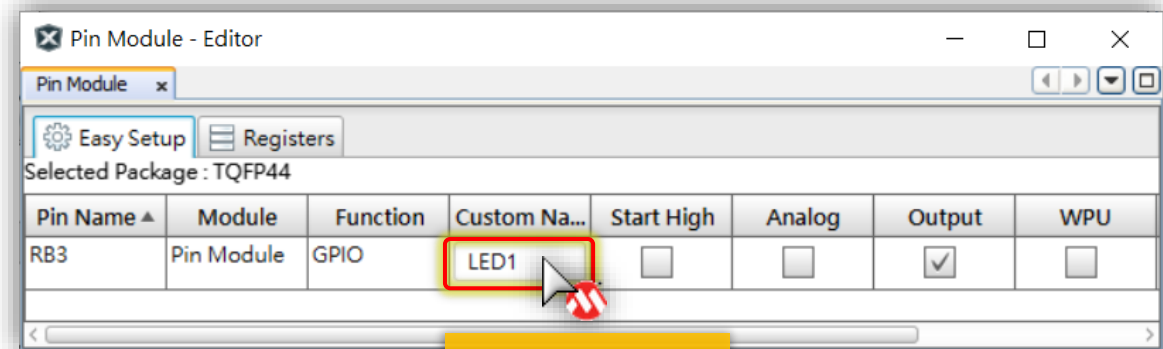
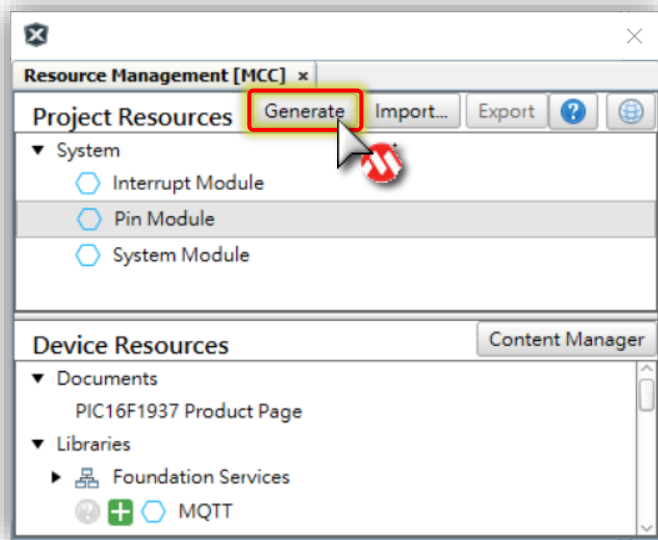
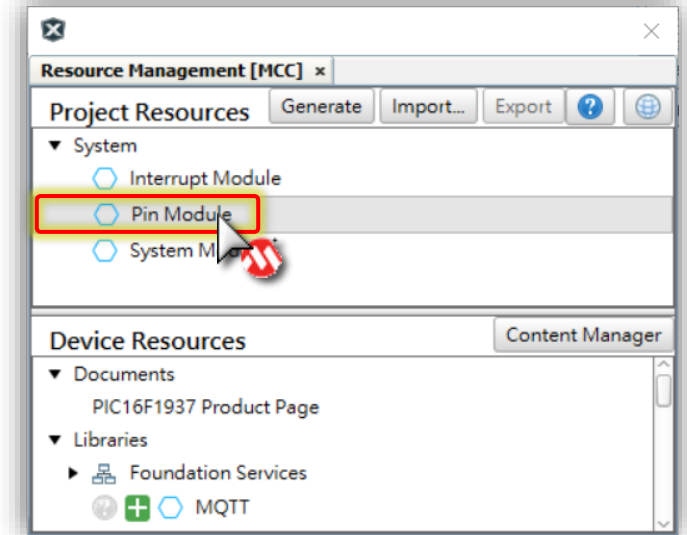
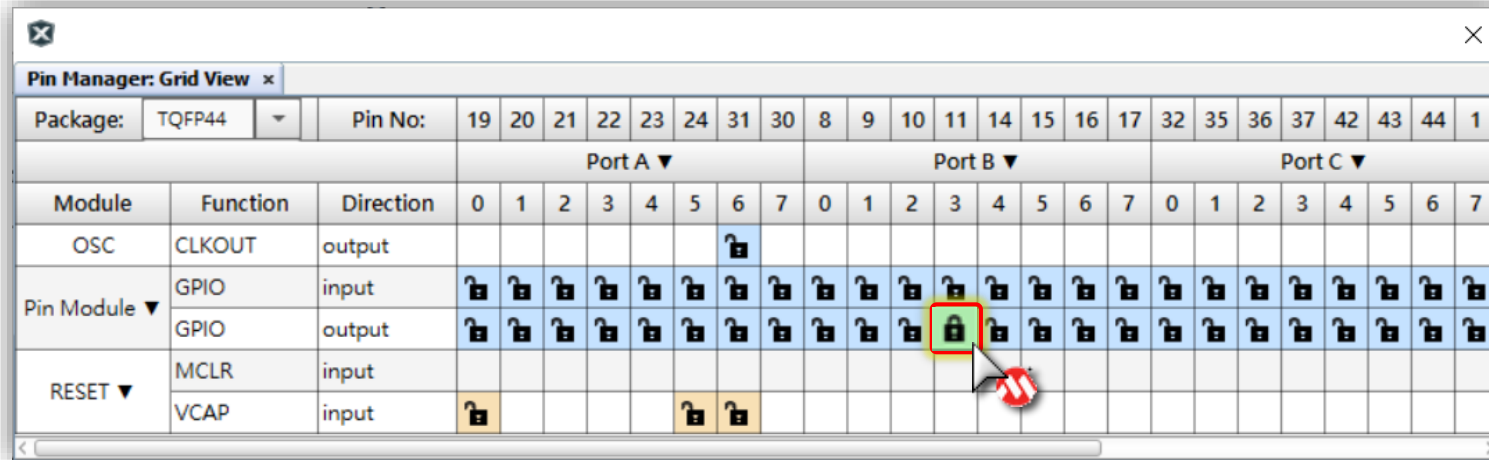
# GPIO Block Diagram

- 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 GPIO Output

---

# Lab1 GPIO Output

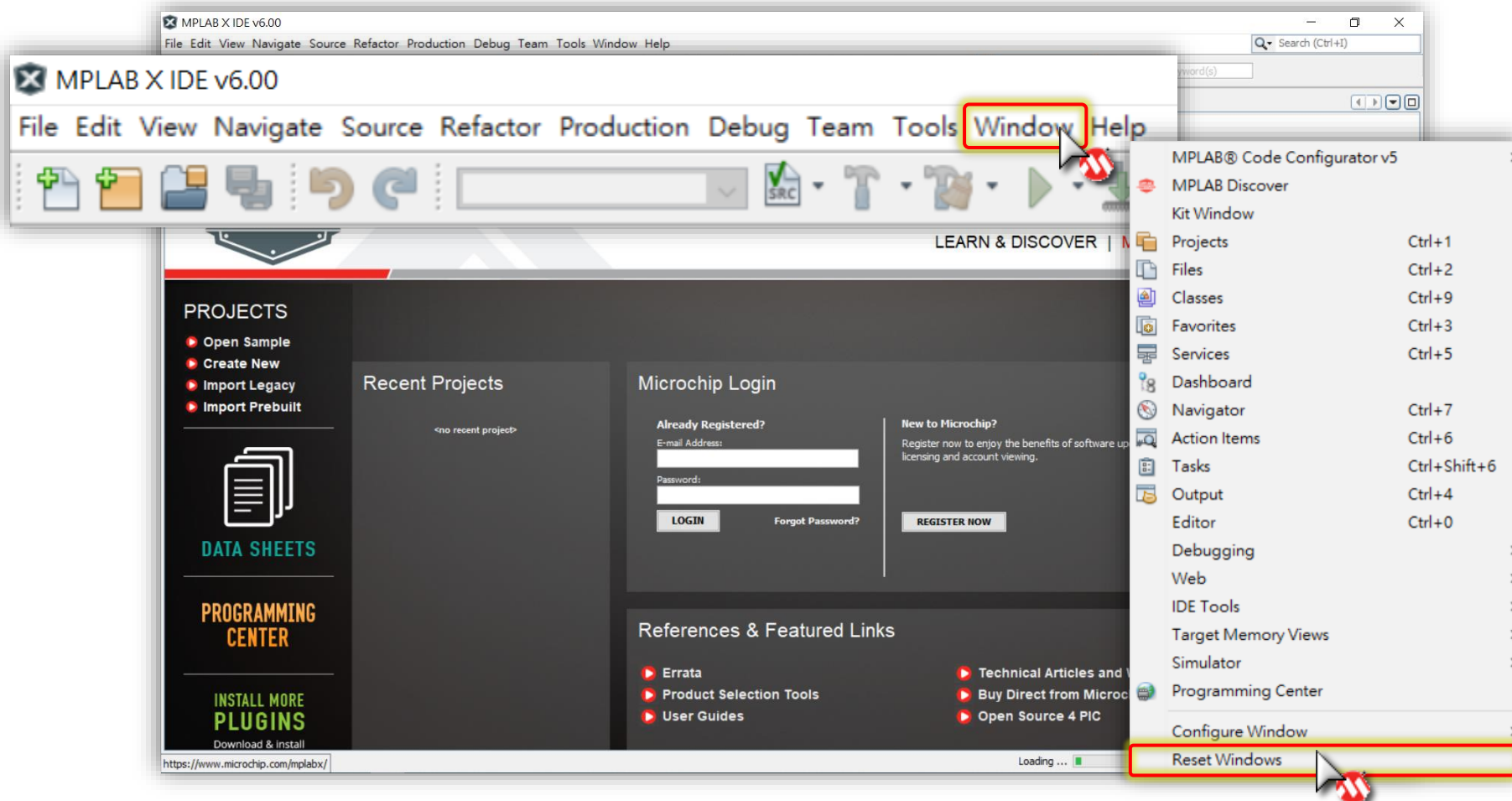
- Base on current project or Open `.\Lab1.2_XXX.X` to do exercises
- Try to use MCC to generate your first MCC style code and control GPIO base on MCC style.
- Try to set the four GPIO pins to digital output mode and toggle levels (period around 500ms ~ 1S), individually.
- **RB03** ▶ **LED(LED1)**.

## How to start ?

# MPLAB X IDE

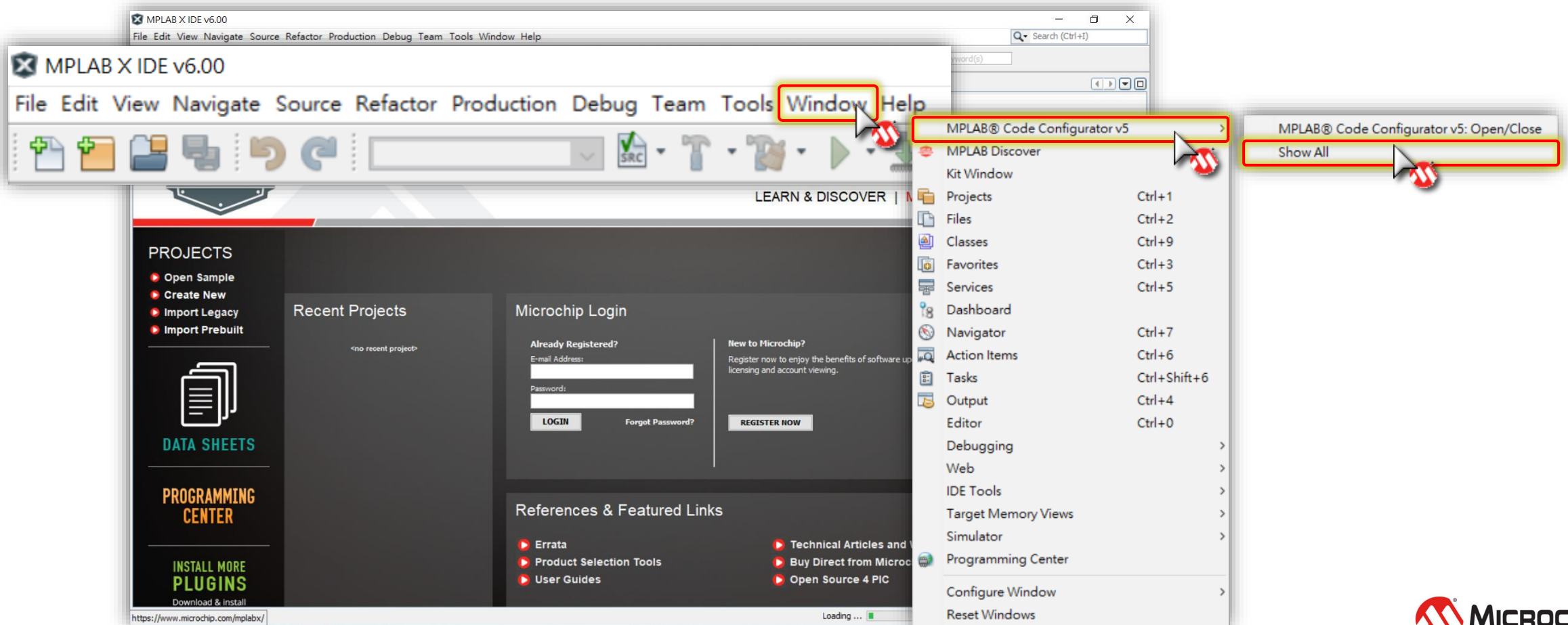
## Tips

- 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

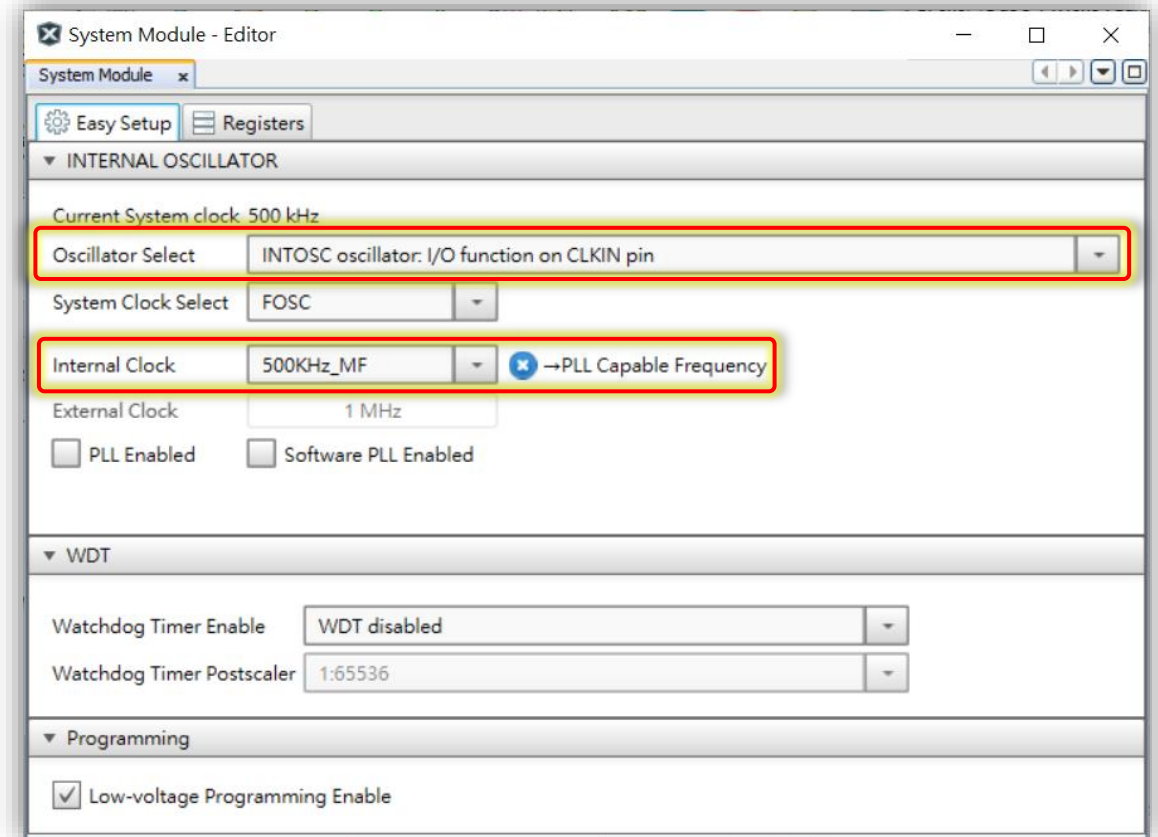
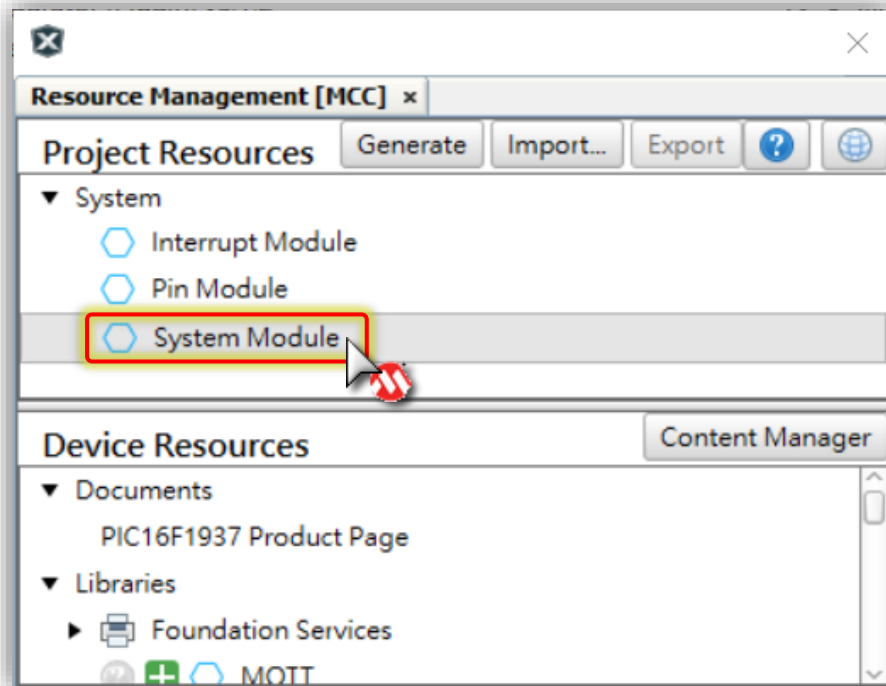
- You may can't find the MCC window you want also. MCC provide named **Show All** to show all MCC windows.



# Lab1 GPIO Output

## Step 1

- Set System Clock.
  - Oscillator Select : INTOSC oscillator. I/O function on CLKIN pin
  - Internal Clock : 500KHz\_MF



# Lab1 GPIO Output

## Step 2

- Change package
  - UQFN40 ► TQFP44

Pin Manager: Grid View

Package: TQFP44

Pin No:

19

20

21

22

23

24

31

30

8

9

10

11

14

15

16

17

32

35

36

37

42

43

44

1

Port A

Port B

Port C

Module

Function

Direction

0

1

2

3

4

5

6

7

0

1

2

3

4

5

6

7

0

1

2

3

4

5

6

7

OSC

CLKOUT

output

Pin Module

GPIO

input

GPIO

output

RESET

MCLR

input

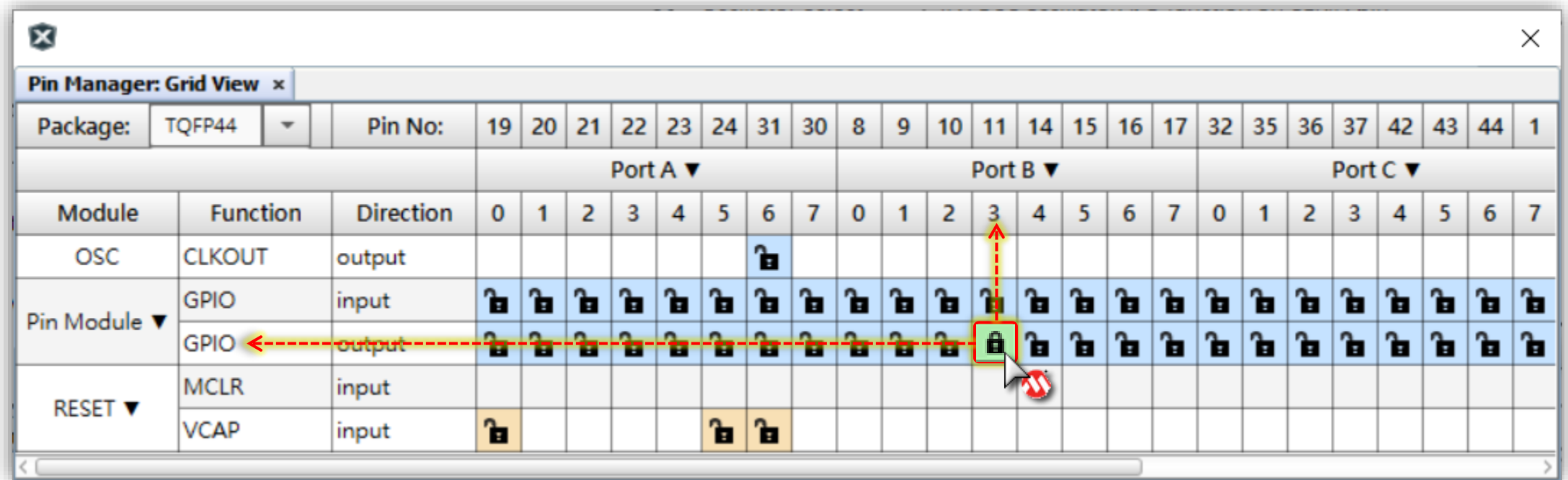
VCAP

input

# Lab1 GPIO Output

## Step 3

- Set **PORTB, Pin3 (RB3)** to digital output mode.
- **Pin Manager : Grid View** : **Click RB3 lock to GPIO output.**



Pin Manager: Grid View

Package: TQFP44

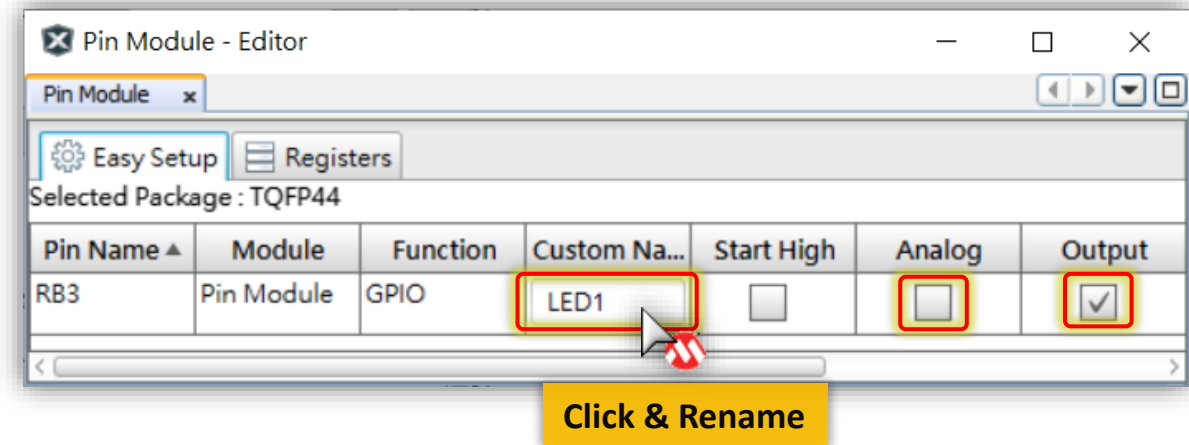
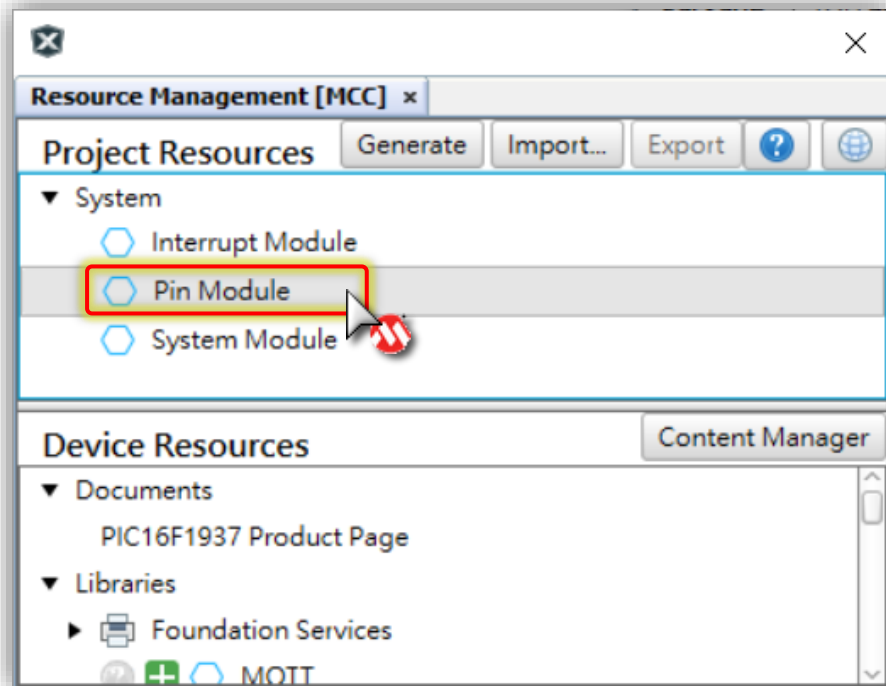
Pin No: 19 20 21 22 23 24 31 30 8 9 10 11 14 15 16 17 32 35 36 37 42 43 44 1

|              |          |           | Port A ▼ |   |   |   |   |   |   |   | Port B ▼ |   |   |   |   |   |   |   | Port C ▼ |   |   |   |   |   |   |   |
|--------------|----------|-----------|----------|---|---|---|---|---|---|---|----------|---|---|---|---|---|---|---|----------|---|---|---|---|---|---|---|
| Module       | Function | Direction | 0        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0        | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| OSC          | CLKOUT   | output    |          |   |   |   |   |   | 🔒 |   |          |   |   |   |   |   |   |   |          |   |   |   |   |   |   |   |
| Pin Module ▼ | GPIO     | input     | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |   |
|              | GPIO     | output    | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |   |
| RESET ▼      | MCLR     | input     |          |   |   |   |   |   |   |   |          |   |   |   |   |   |   |   |          |   |   |   |   |   |   |   |
|              | VCAP     | input     | 🔒        |   |   |   |   | 🔒 | 🔒 |   |          |   |   |   |   |   |   |   |          |   |   |   |   |   |   |   |

# Lab1 GPIO Output

## Step 4

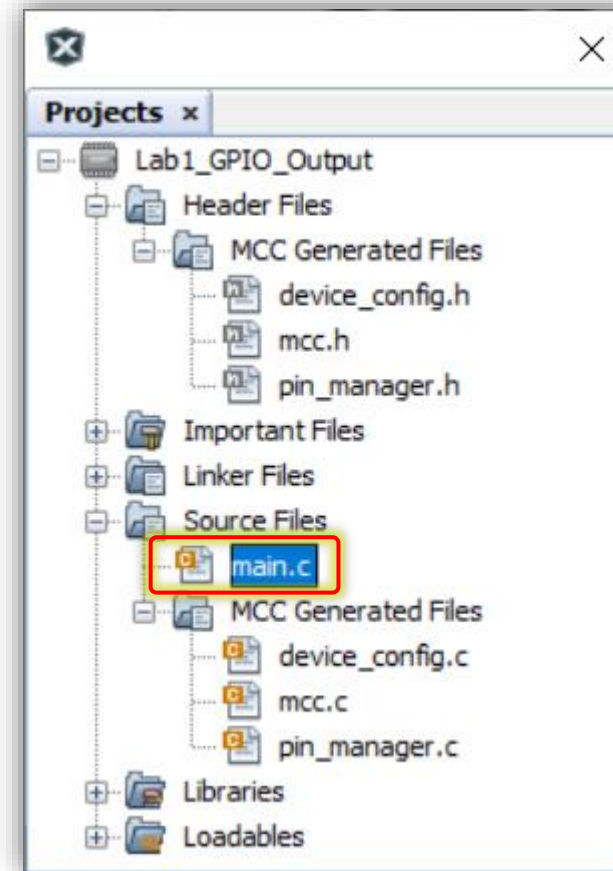
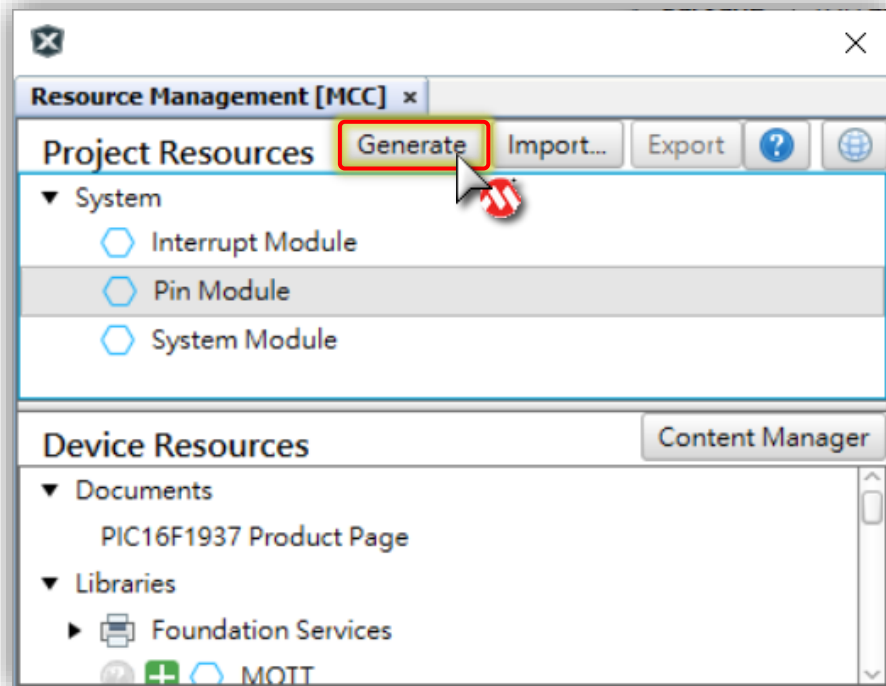
- Set Alias for RB3.
  - **Pin Module** : uncheck Analog, Check Output, Custom Name ► LED1.



# Lab1 GPIO Output

## Step 5

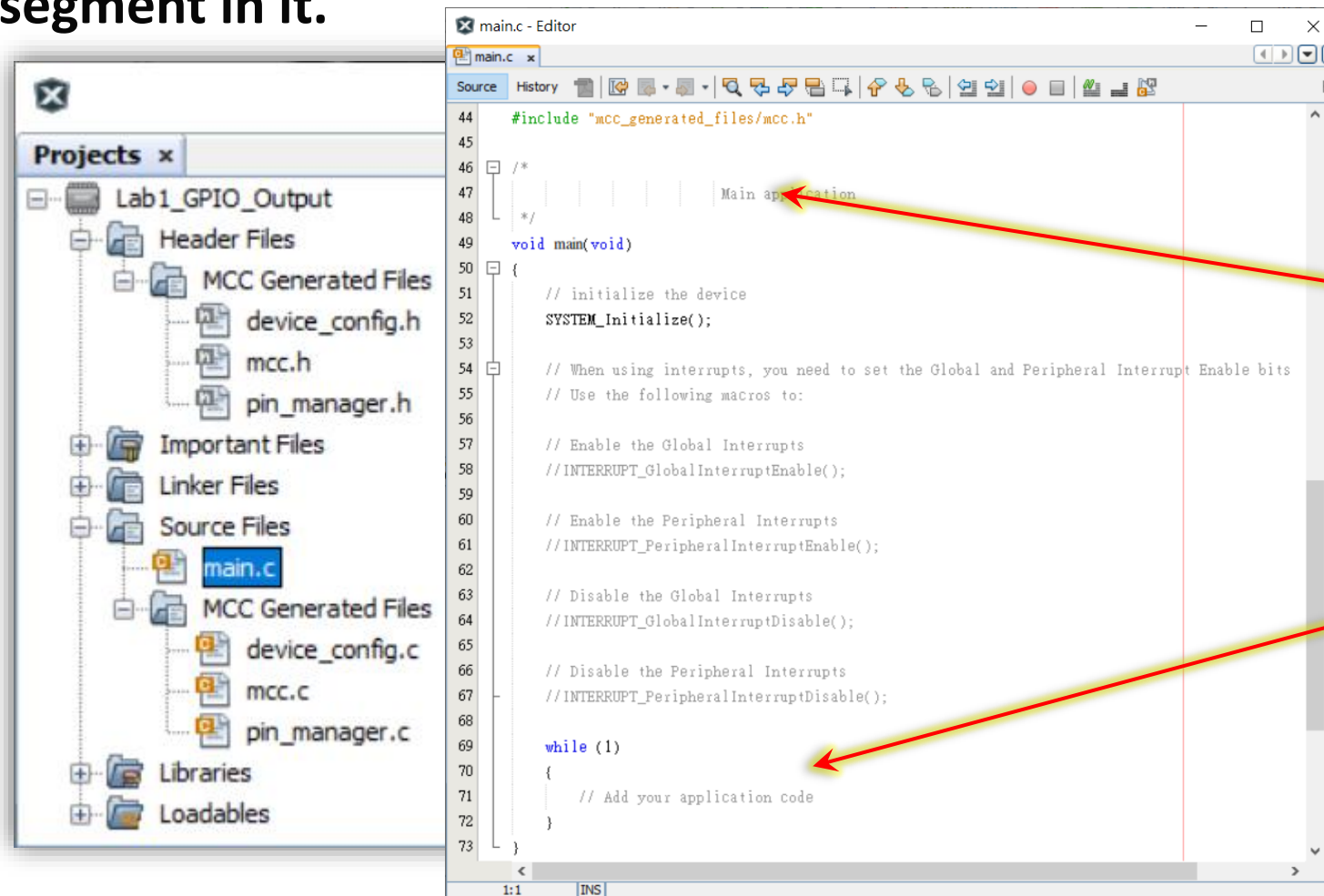
- Click **Generate** Button to generate your code.
- All files generate by MCC include **main.c** automatically.



# Lab1 GPIO Output

## Step 6

- 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/mcc.h"
```

```
uint32_t i = 0;
```

```
int main(void)
```

```
{  
    // initialize the device  
    SYSTEM_Initialize();
```

```
    while (1)
```

```
{  
    // Add your application code  
    LED1_Toggle();  
    for (i = 0; i < 2800; i++);
```

```
}
```

# MPLAB X IDE Hints

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

```
51 {
52     // initialize the device
53     SYSTEM_Initialize();
54
55     // When using interrupts, you need to set
56     // Use the following macros to:
57
58     // Enable the Global Interrupts
59     //INTERRUPT_GlobalInterruptEnable();
60
61     // Enable the Peripheral Interrupts
62     //INTERRUPT_PeripheralInterruptEnable();
63
64     // Disable the Global Interrupts
65     //INTERRUPT_GlobalInterruptDisable();
66
67     // Disable the Peripheral Interrupts
68     //INTERRUPT_PeripheralInterruptDisable();
69
70     while (1)
71     {
72         // Add your application code
73         LED1
74     }
75 }
```

44:12 | INS | Unable to resolve identifier LED1.



```
51 {
52     // initialize the device
53     SYSTEM_Initialize();
54
55     // When using interrupts, you need to set
56     // Use the following macros to:
57
58     // Enable
59     //INTERRUPT
60
61     // Enable
62     //INTERRUPT
63
64     // Disabl
65     //INTERRUPT
66
67     // Disabl
68     //INTERRUPT
69
70     while (1)
71     {
72         // Ad
73         LED1
74     }
75 }
```

44:12 | INS | Unable to resolve identifier LED1.



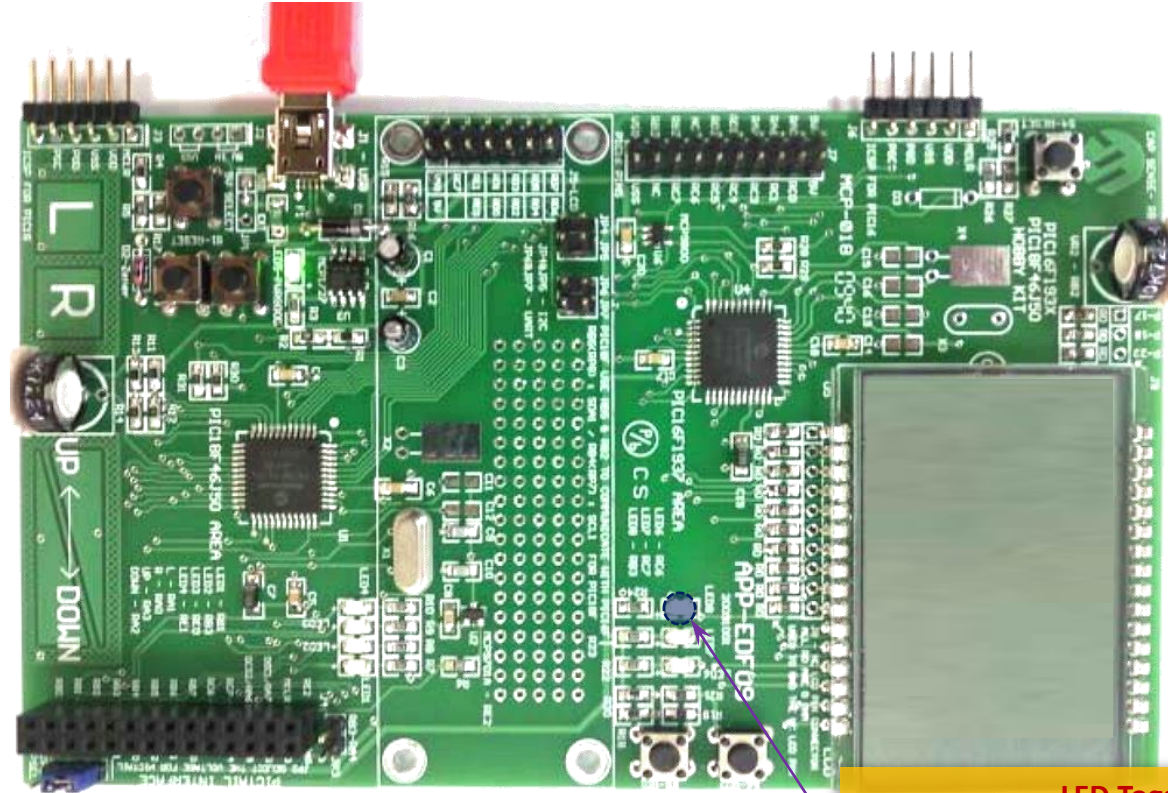
```
51 {
52     // initialize the device
53     SYSTEM_Initialize();
54
55     // When using interrupts, you need to set
56     // Use the following macros to:
57
58     // Enable the Global Interrupts
59     //INTERRUPT_GlobalInterruptEnable();
60
61     // Enable the Peripheral Interrupts
62     //INTERRUPT_PeripheralInterruptEnable();
63
64     // Disable the Global Interrupts
65     //INTERRUPT_GlobalInterruptDisable();
66
67     // Disable the Peripheral Interrupts
68     //INTERRUPT_PeripheralInterruptDisable();
69
70     while (1)
71     {
72         // Add your application code
73         LED1_Toggle;
74     }
75 }
```

42:3 | INS | unexpected token: ;

# Lab1 GPIO Output

## Step 7 & Result

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



LED Toggle,  
Interval Period use "for()" loop delay.

# Lab1 GPIO Output

## Code Example

```
#include "mcc_generated_files/mcc.h"

uint32_t i = 0;

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

    // When using interrupts, you need to set the Global and Peripheral Interrupt Enable bits
    // Use the following macros to:

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    while (1)
    {
        // Add your application code
        LED1_Toggle();
        for (i = 0; i < 2800; i++);
    }
}
```

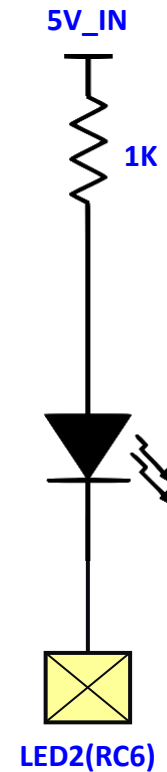
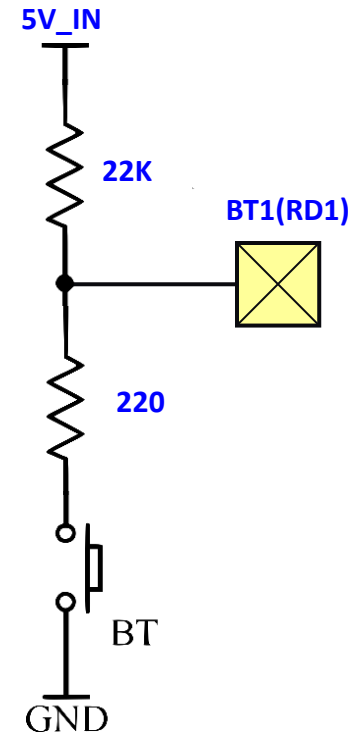
# Lab2 GPIO Input

---

# Lab2 GPIO Input

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

```
if(BT1_GetValue())  
{  
    LED2_SetHigh();  
}  
else  
{  
    LED2_SetLow();  
}
```



# Lab2 GPIO Input

- Base on current project or Open `.\Lab2.2_xxx.X` to do exercises
- Try to set **RD01**, **RD02** to digital input as get button status.
- Try use button to control LED light or dark.
  - BT1 Pressed -> LED2 Light, BT1 Released -> LED2 Dark
  - BT2 Pressed -> LED3 Light, BT2 Released -> LED3 Dark
- **RB03** ▶ **LED(LED1)**.
- **RD01** ▶ **Button(BT1)**.
- **RD02** ▶ **Button(BT2)**.
- **RC06** ▶ **LED(LED2)**.
- **RC07** ▶ **LED(LED3)**.

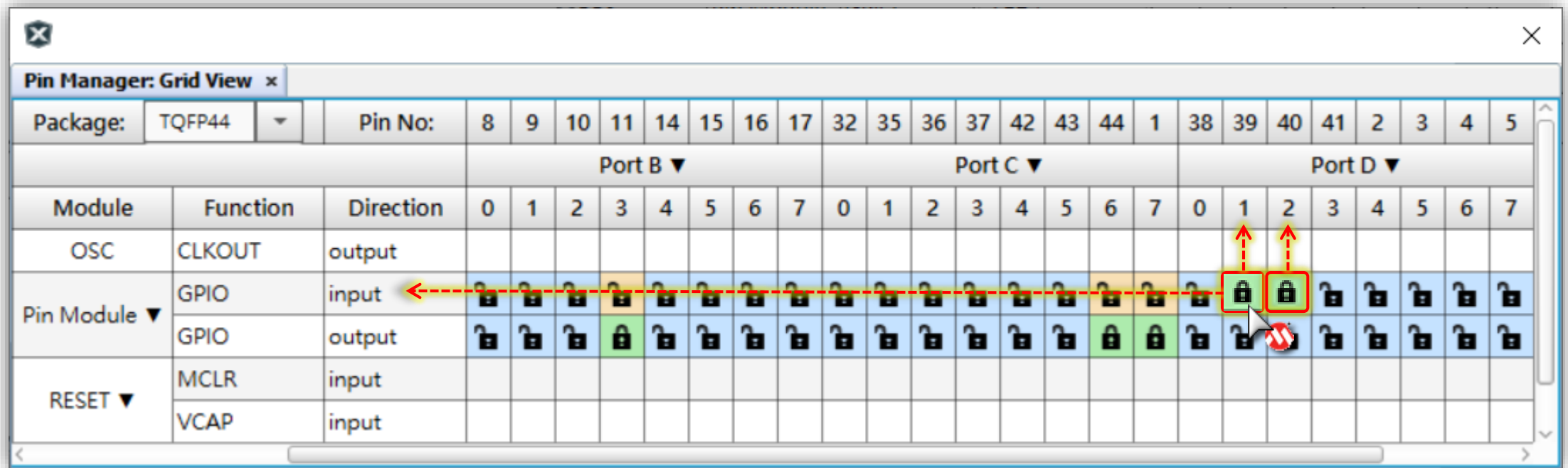
# Let's go!



# Lab2 GPIO Input

## Step 2

- Set PORTD, Pin1 (RD1) , Pin2 (RD2) to digital input mode.
- Pin Manager : Grid View : Click RD1, RD2 lock to GPIO input.



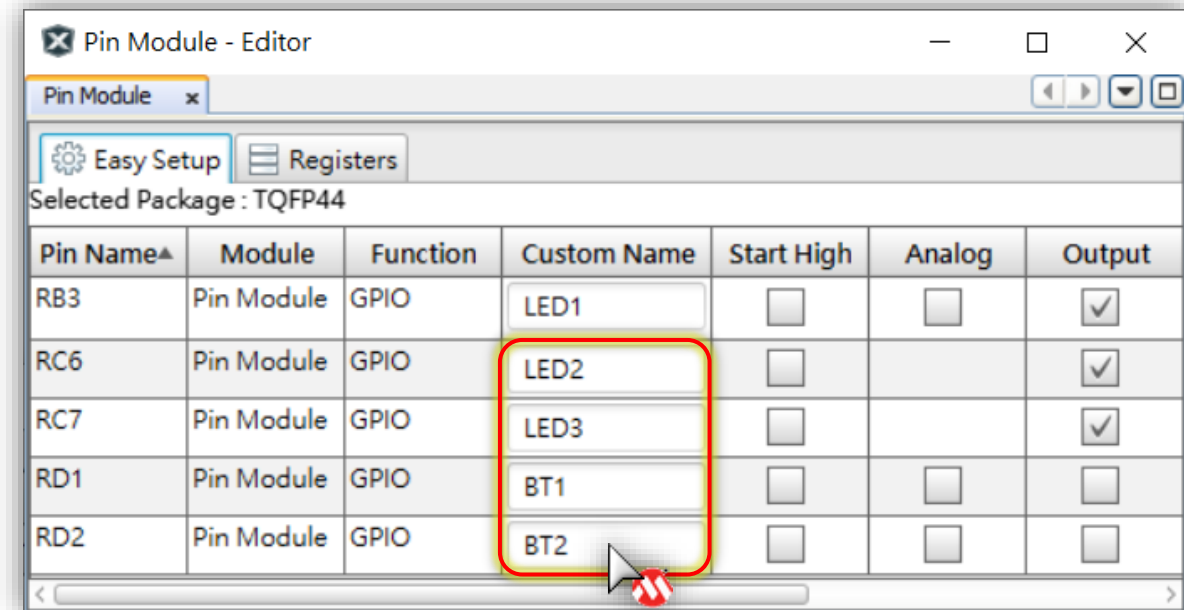
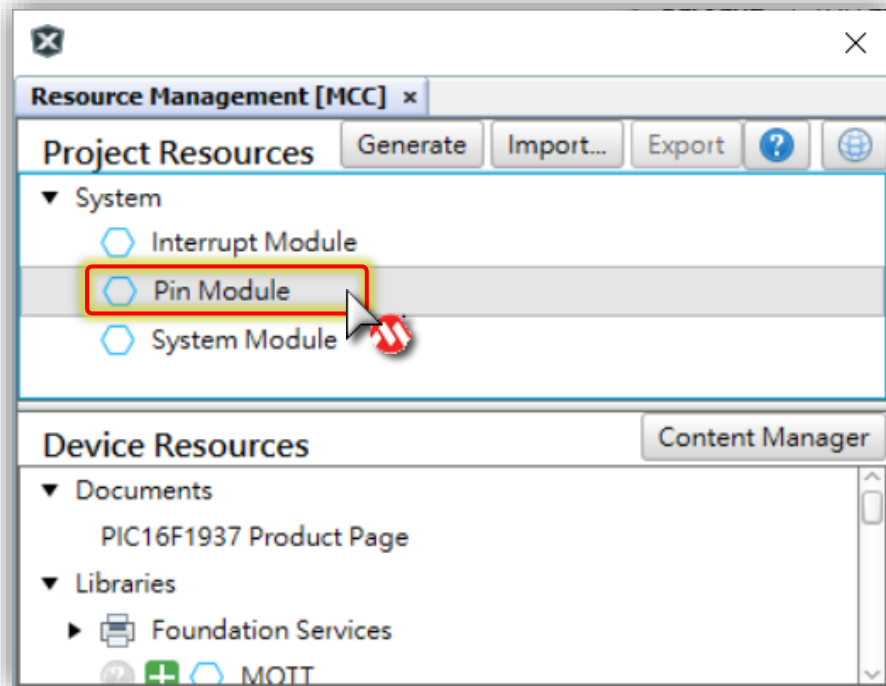
The screenshot shows the 'Pin Manager: Grid View' window for a TQFP44 package. The grid displays various modules and their functions across different ports. The 'Pin Module' section is highlighted, showing the configuration for PORTD pins 1 and 2. Red dashed arrows point to the lock icons for these pins, indicating they should be clicked to set them to digital input mode.

| Package:     | TQFP44   | Pin No:   | 8        | 9 | 10 | 11 | 14 | 15 | 16 | 17 | 32       | 35 | 36 | 37 | 42 | 43 | 44 | 1 | 38       | 39 | 40 | 41 | 2 | 3 | 4 | 5 |
|--------------|----------|-----------|----------|---|----|----|----|----|----|----|----------|----|----|----|----|----|----|---|----------|----|----|----|---|---|---|---|
|              |          |           | Port B ▼ |   |    |    |    |    |    |    | Port C ▼ |    |    |    |    |    |    |   | Port D ▼ |    |    |    |   |   |   |   |
| Module       | Function | Direction | 0        | 1 | 2  | 3  | 4  | 5  | 6  | 7  | 0        | 1  | 2  | 3  | 4  | 5  | 6  | 7 | 0        | 1  | 2  | 3  | 4 | 5 | 6 | 7 |
| OSC          | CLKOUT   | output    |          |   |    |    |    |    |    |    |          |    |    |    |    |    |    |   |          |    |    |    |   |   |   |   |
| Pin Module ▼ | GPIO     | input     | 🔒        | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒 | 🔒        | 🔒  | 🔒  | 🔒  | 🔒 | 🔒 | 🔒 | 🔒 |
|              | GPIO     | output    | 🔒        | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒 | 🔒        | 🔒  | 🔒  | 🔒  | 🔒 | 🔒 | 🔒 | 🔒 |
| RESET ▼      | MCLR     | input     |          |   |    |    |    |    |    |    |          |    |    |    |    |    |    |   |          |    |    |    |   |   |   |   |
|              | VCAP     | input     |          |   |    |    |    |    |    |    |          |    |    |    |    |    |    |   |          |    |    |    |   |   |   |   |

# Lab2 GPIO Input

## Step 3

- Set Alias for RC6, RC7, RD1, RD2.
  - **Pin Module** : uncheck Analog, Check Output, Custom Name ► LED1.

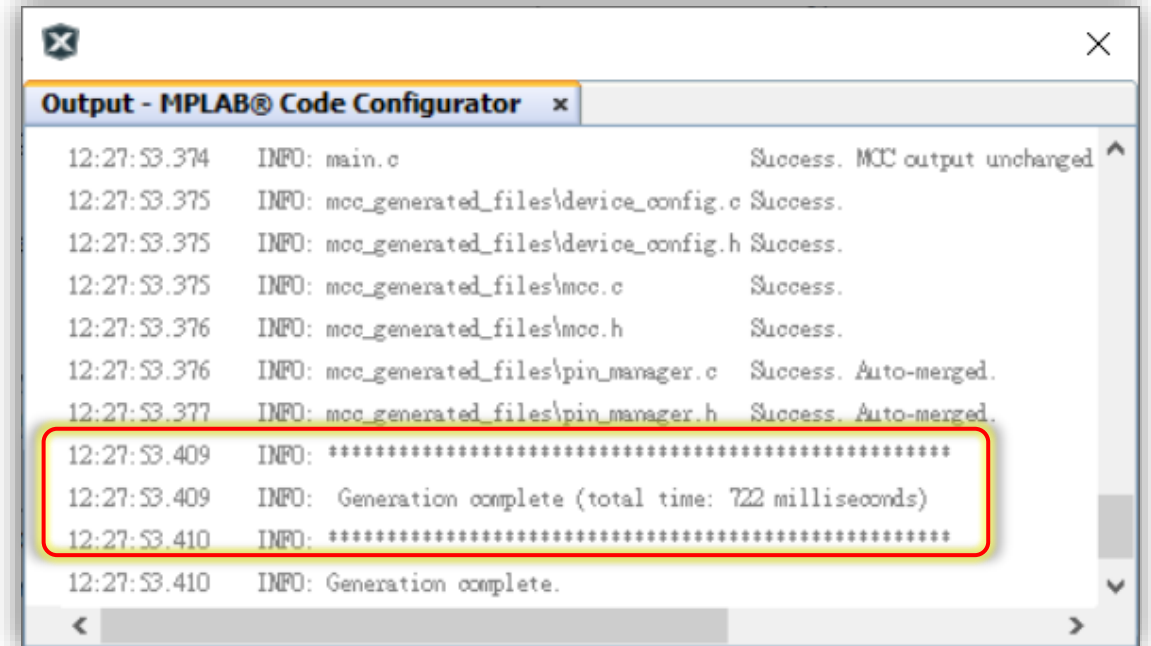
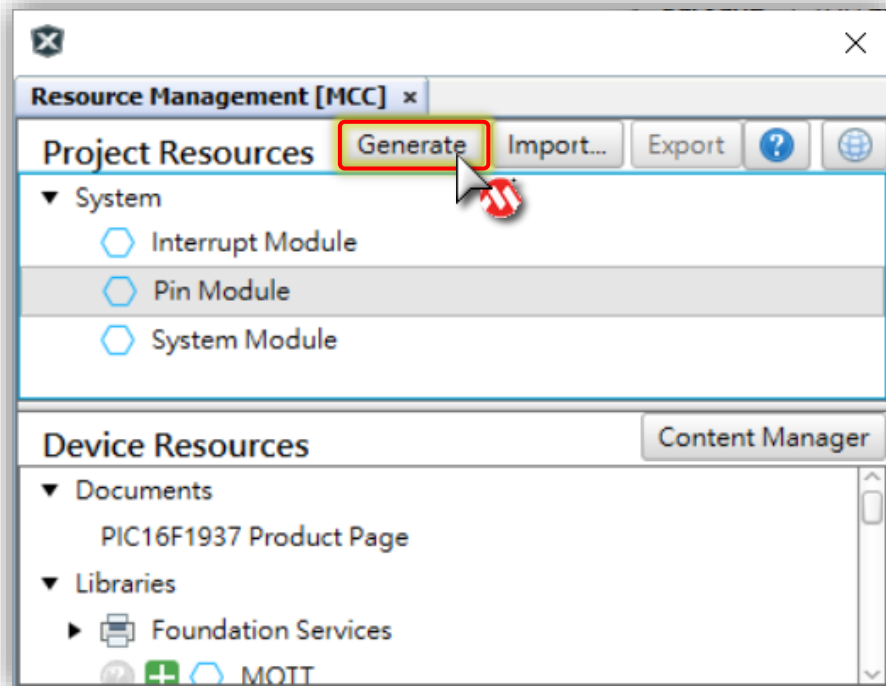


Click & Rename

# Lab2 GPIO Input

## Step 4

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



# Lab2 GPIO Input

## Code Example

```
#include "mcc_generated_files/mcc.h"

uint32_t i = 0;

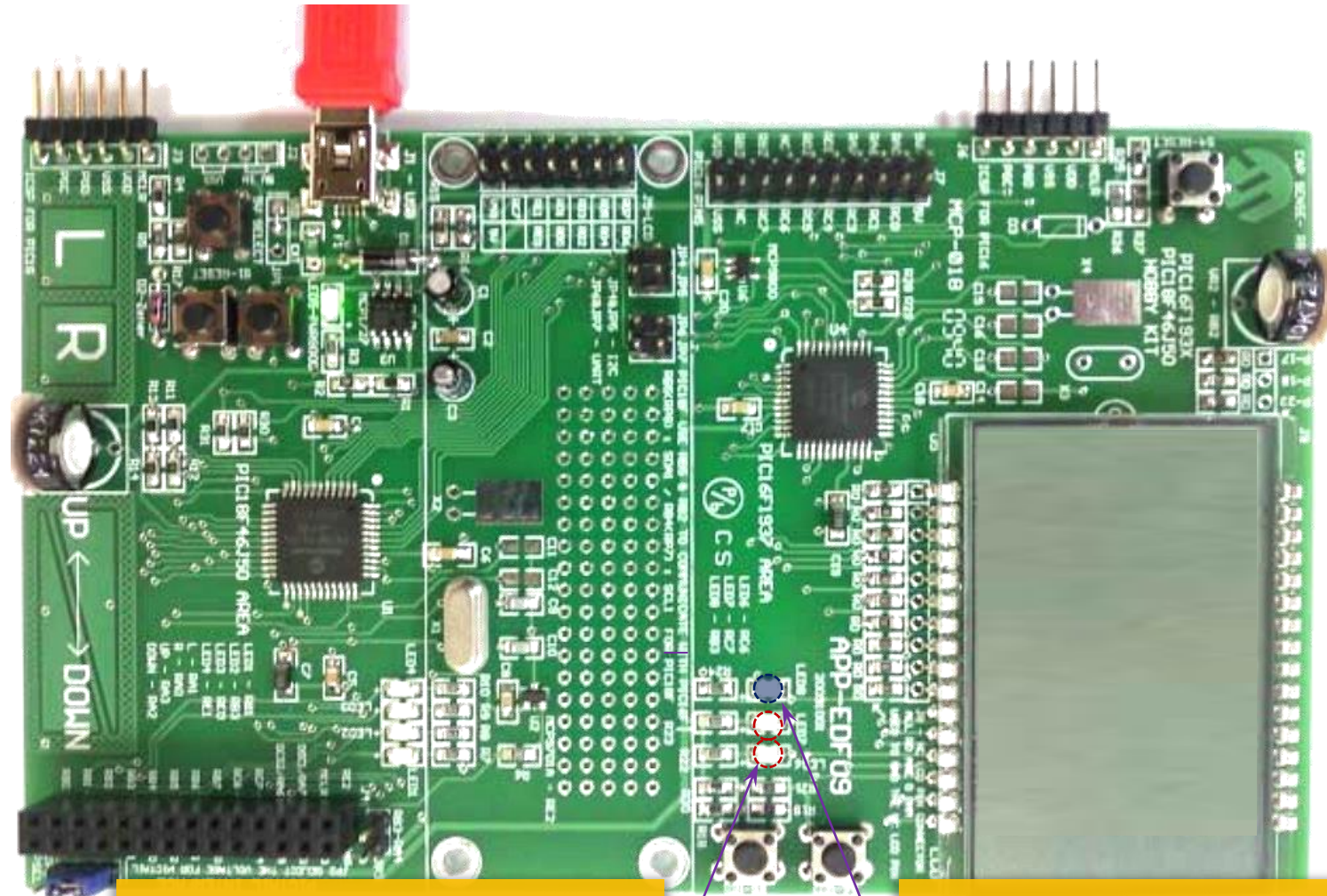
int main(void)
{
    ...

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    while (1)
    {
        // Add your application code
        LED1_Toggle();
        for (i = 0; i < 2800; i++);

        if(!BT1_GetValue())
        {
            LED2_SetLow();
        }
        else if(!BT2_GetValue())
        {
            LED3_SetLow();
        }
        else
        {
            LED2_SetHigh();
            LED3_SetHigh();
        }
    }
}
```

## Result



**LED2 Turn On, When BT1 Pressed.  
LED3 Turn On, When BT2 Pressed.**

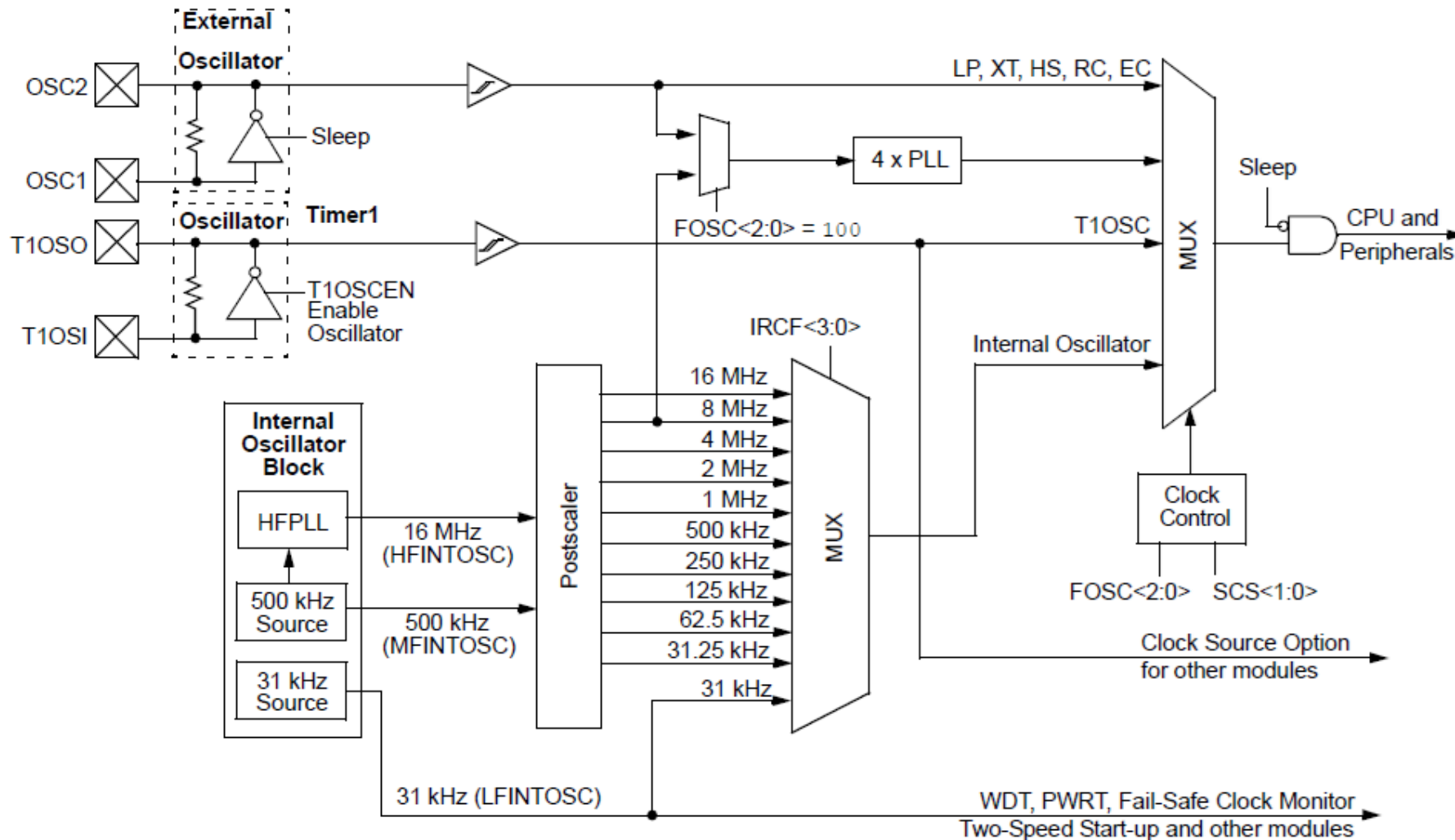
**LED Toggle,  
Interval Period use "for()" loop delay.**

# Oscillator Architecture

---

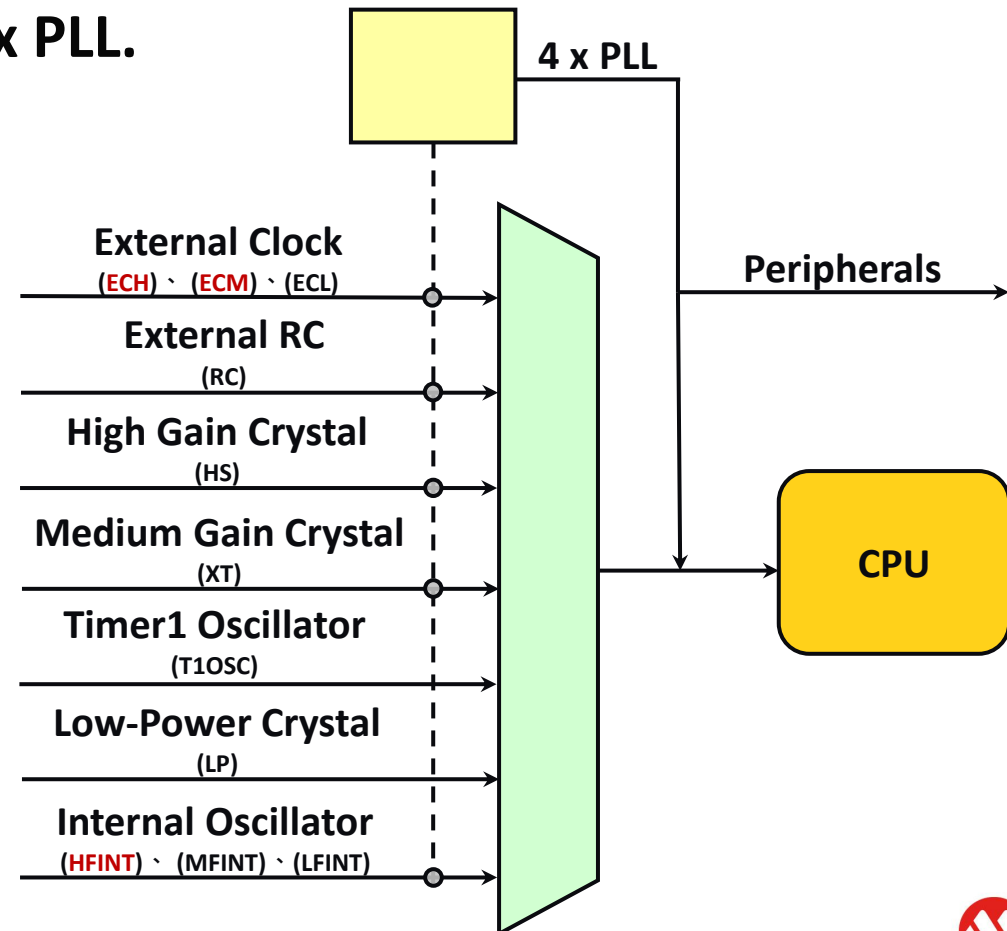
# Phase Locked Loop

- A block diagram of the PLL module.



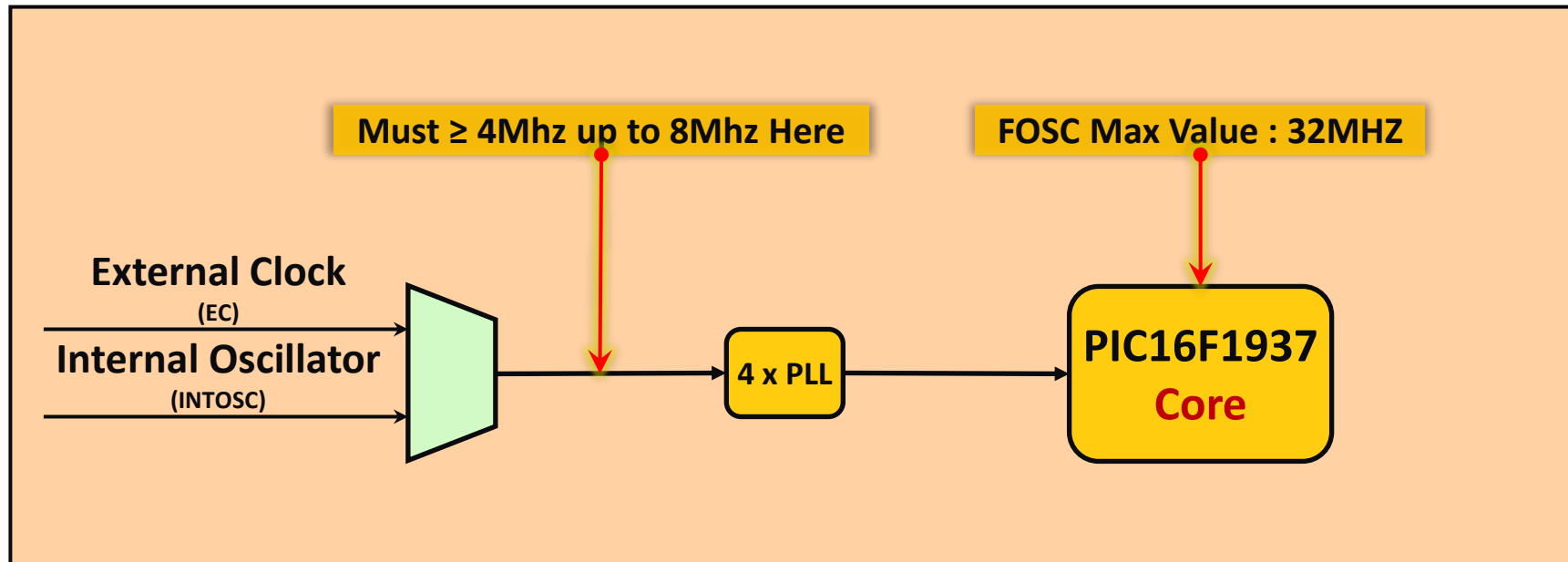
# PIC16F Series Clocks

- Microchip PIC16F1937 MCU available clock sources are Internal Oscillator, Low Power Crystal and External Clock and RC or etc..
- Software-controllable switching between various clock sources.
- The clocks can be divided or multiply by 4 x PLL.
  - ECL : 0MHz to 0.5MHz
  - ECM : 0.5MHz to 4MHz
  - ECH : 4MHz to 8MHz
  - LP : 32kHz
  - XT : up to 4MHz
  - HS : 4MHz to 20MHz
  - INTOSC : 31kHz to 32MHz

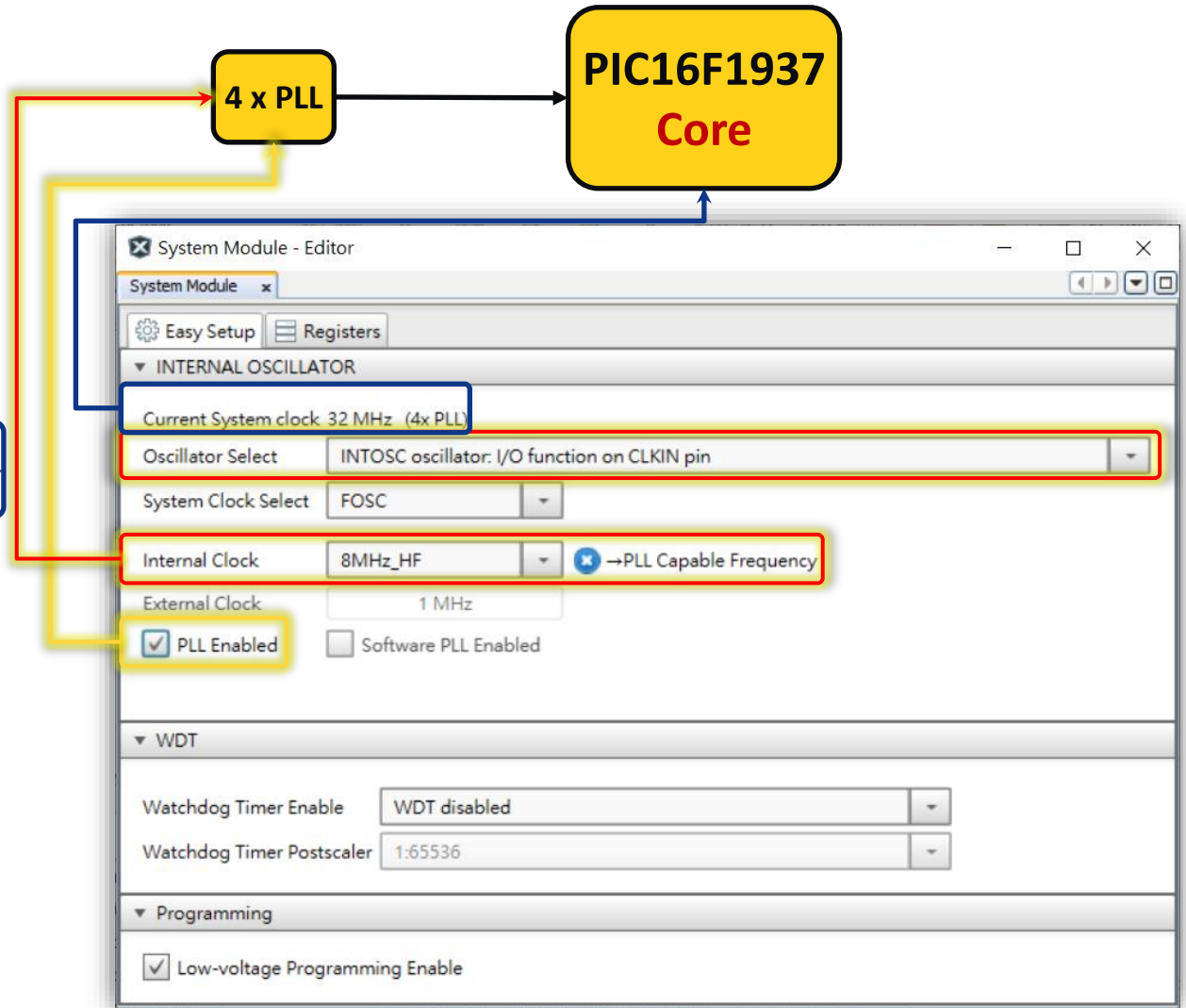
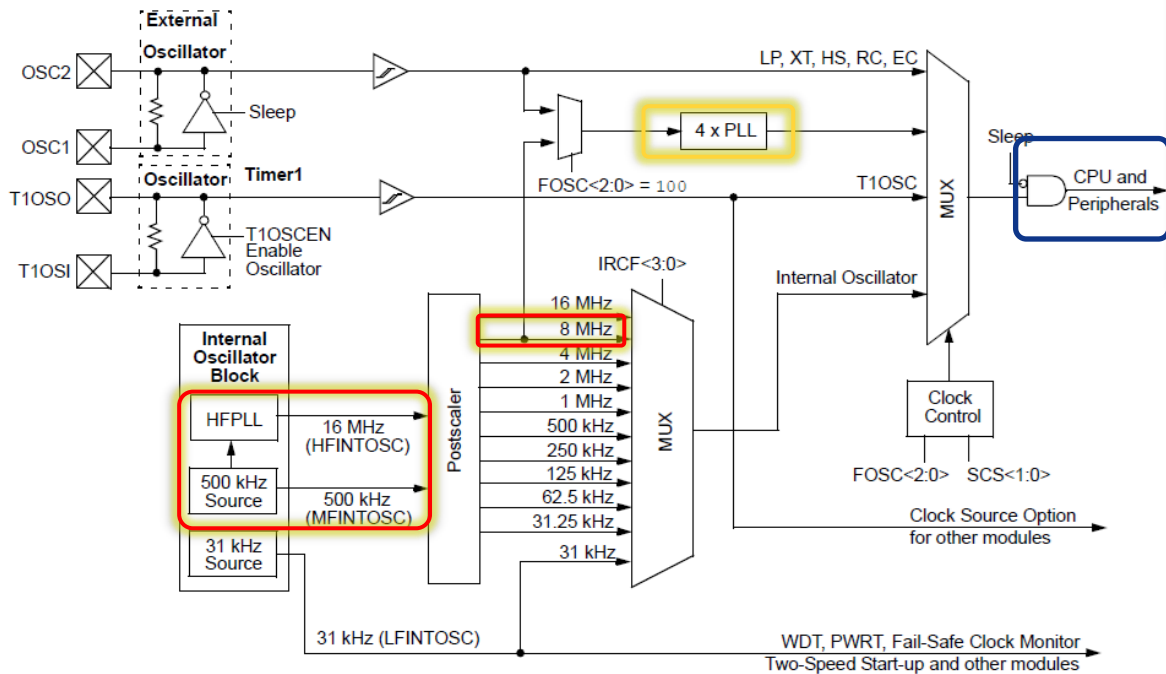


# Phase Locked Loop

- Clock input must 4MHz to 8MHz after PLLPRE, which can use **External Clock** (Crystal or External Clock) or **Internal Fast RC** as clock source.



# Oscillator Setting of MCC



# Lab3 Clock PRIPLL

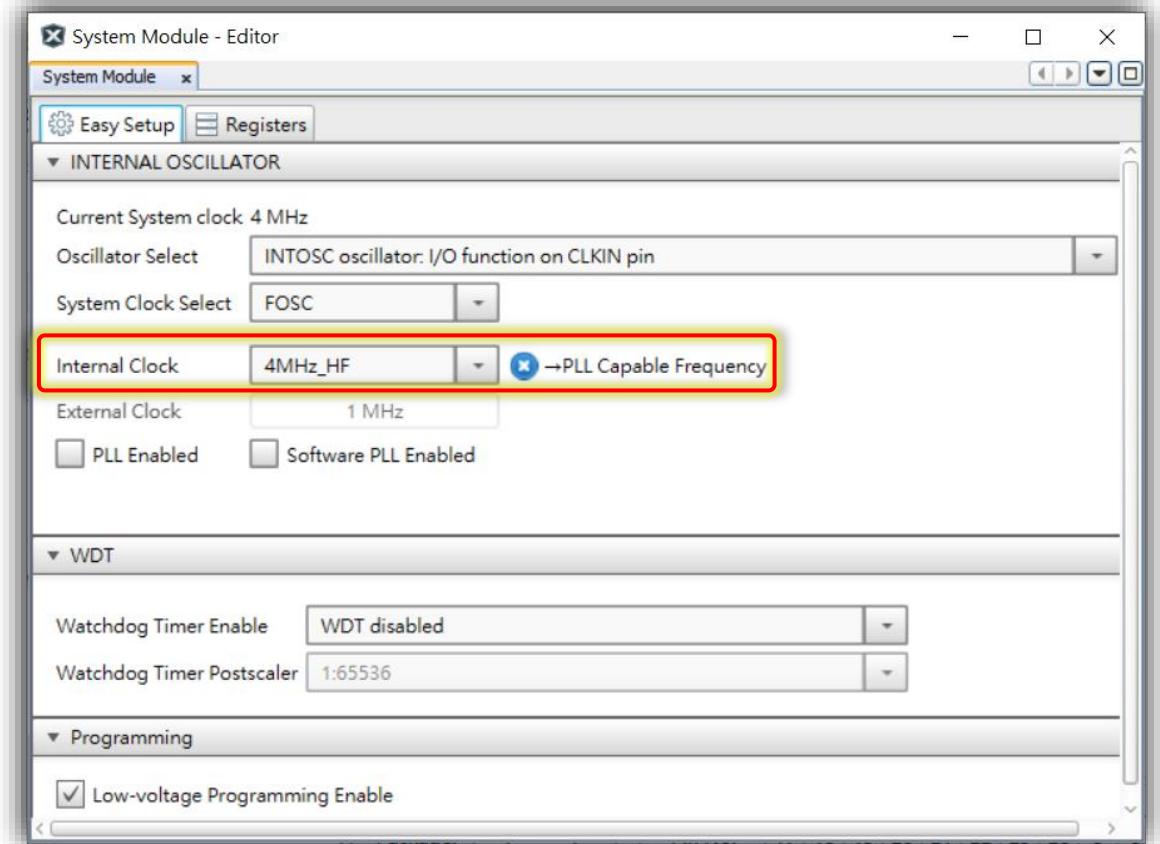
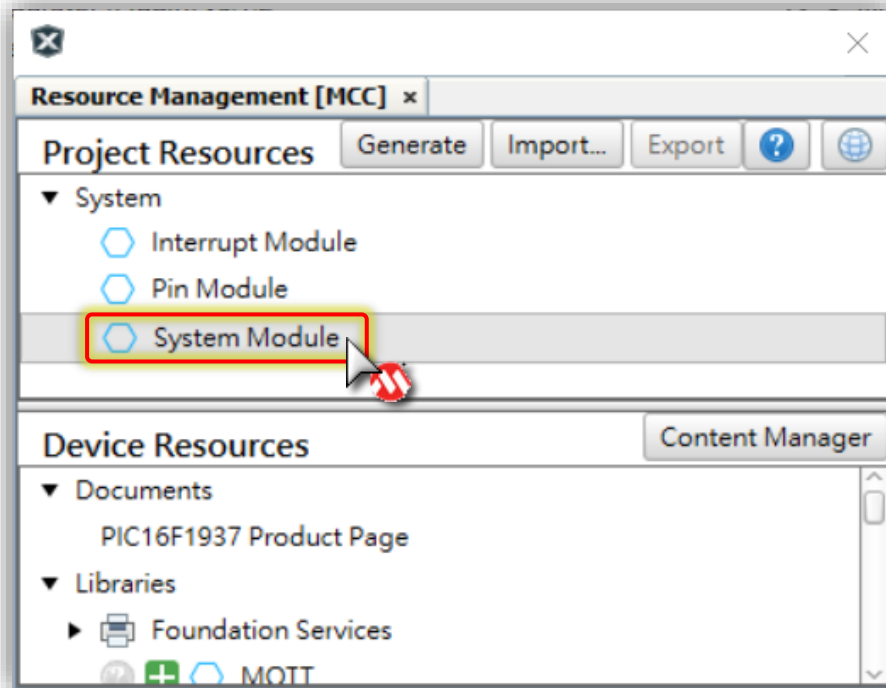
- Base on current project or Open . \Lab3.2\_xxx.X to do exercises
- Try provide maximum frequency (4MHz) to MCU.

Let's go!

# Lab3 Clock PRIPLL

## Step 1

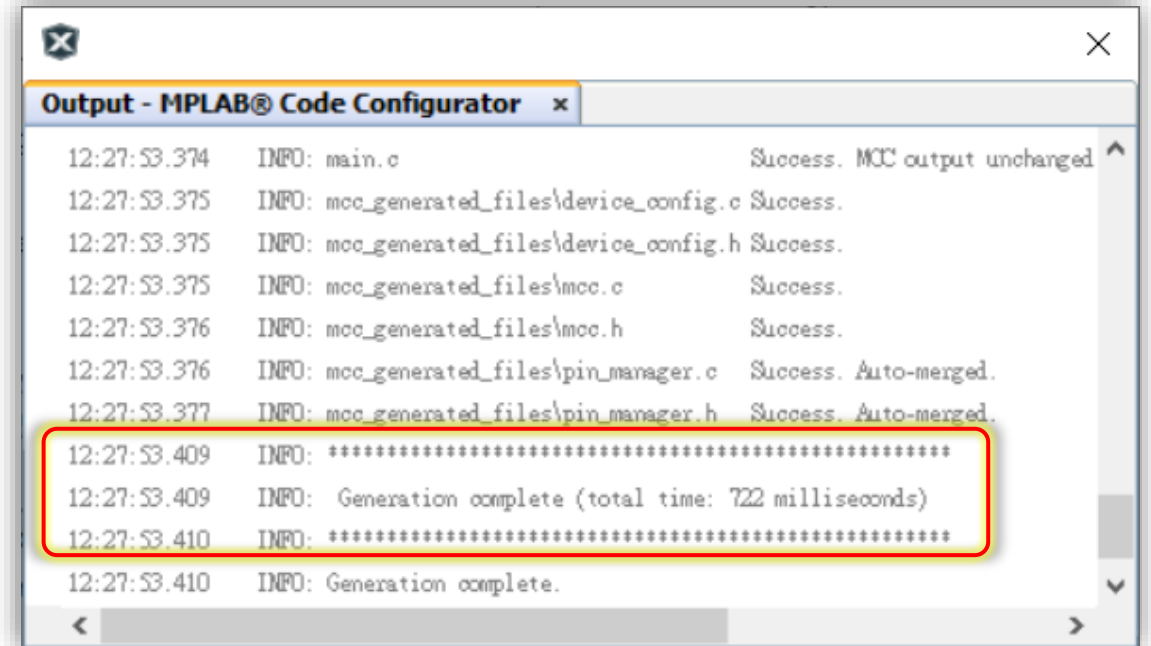
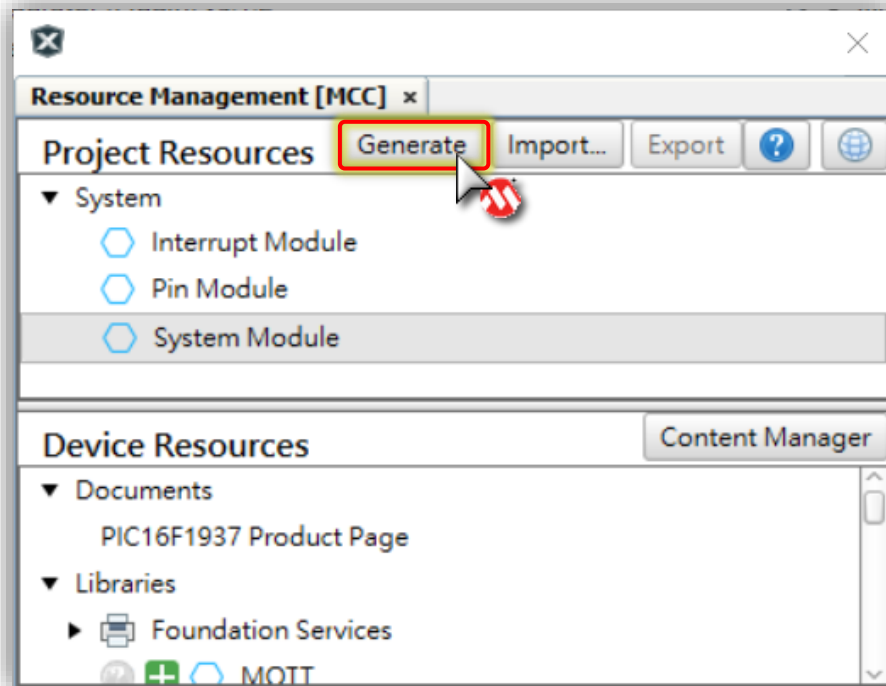
- Set Internal Clock.
  - Internal Clock : 500KHz\_MF ► 4MHz\_HF



# Lab3 Clock PRIPLL

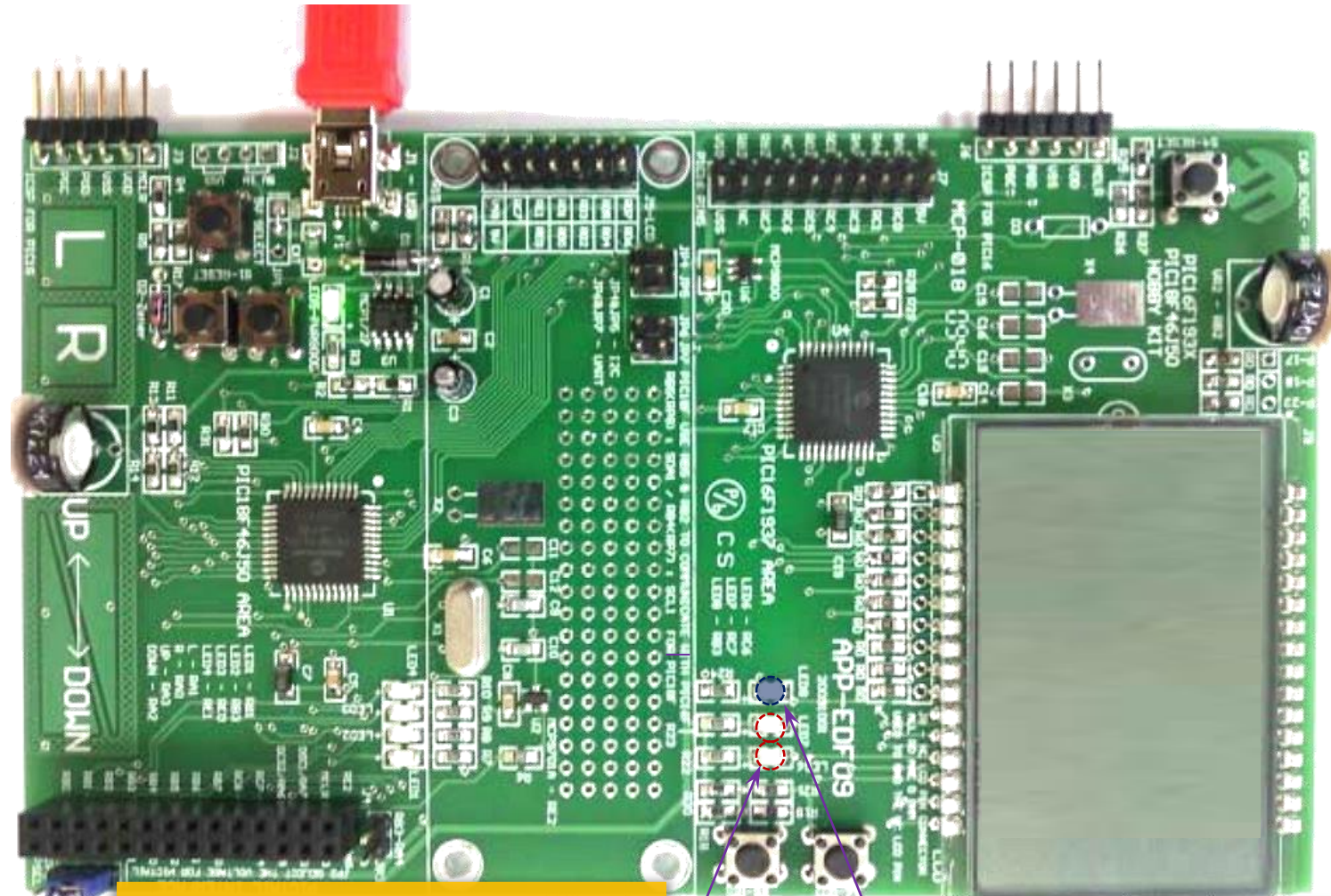
## Step 2

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



# Lab3 Clock PRIPLL

## Result



LED2 Turn On, When BT1 Pressed.  
LED3 Turn On, When BT2 Pressed.

LED Blink Speed More Quickly.

# Interrupt Architecture

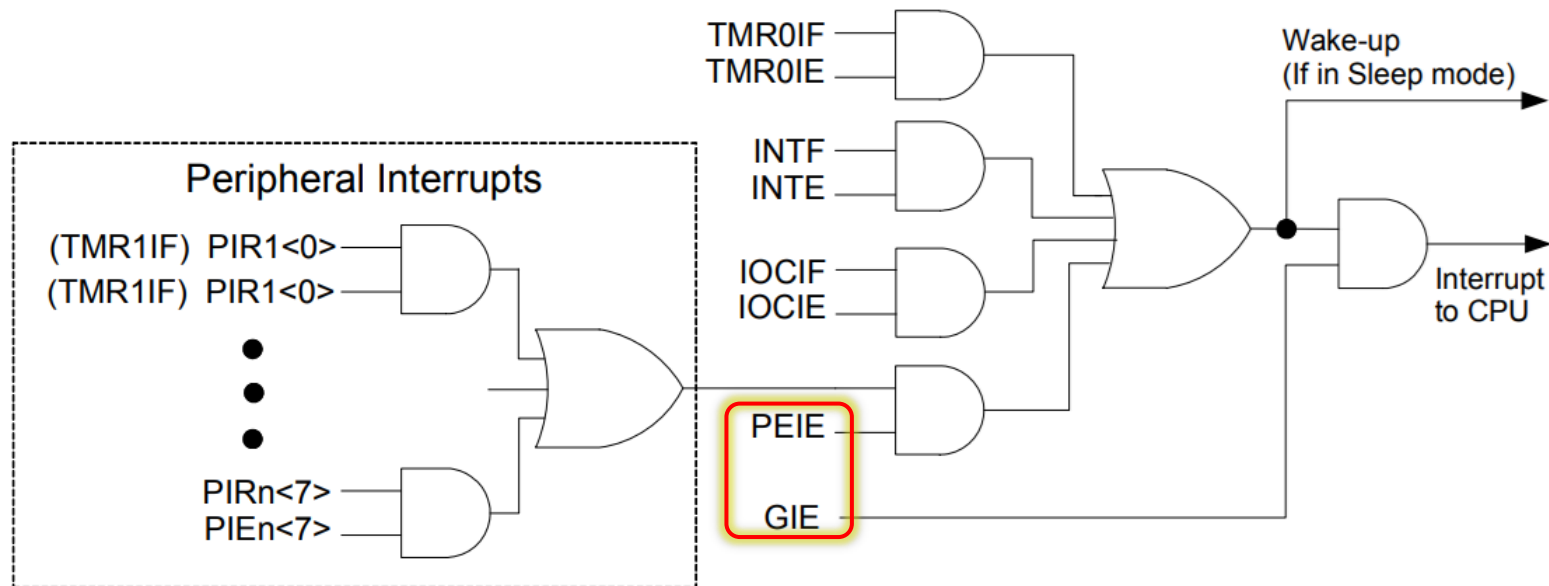
---

# 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.

# PIC16F Series Interrupt Architecture

- The interrupt enabled by setting the **GIE**, **PEIE**, and Interrupt Enable bit(s).
- PIC16F has only one interrupt entry point, address is **0x0004**.
- Support multi-interrupt event processing, but **no priority function**.
- Interrupt priority is determined by **polling** in sequence.
- PIC16F background storage adopts Shadow mode.



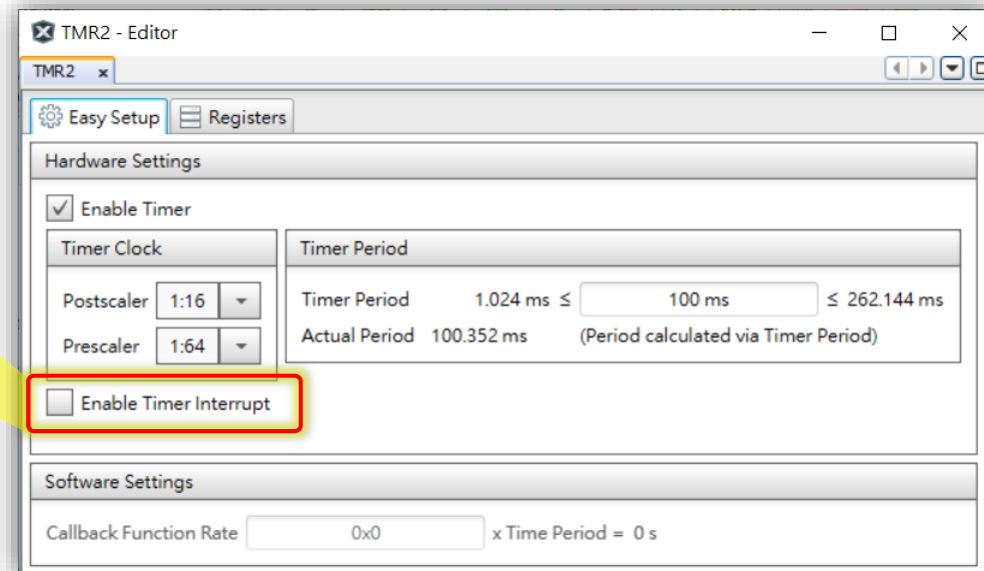
**Interrupt logic**

# Coding for Interrupt

- **A complete interrupt code segment includes below 3 parts.**
  - Interrupt Service Routine (ISR)
  - Enable Interrupt
  - 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.**

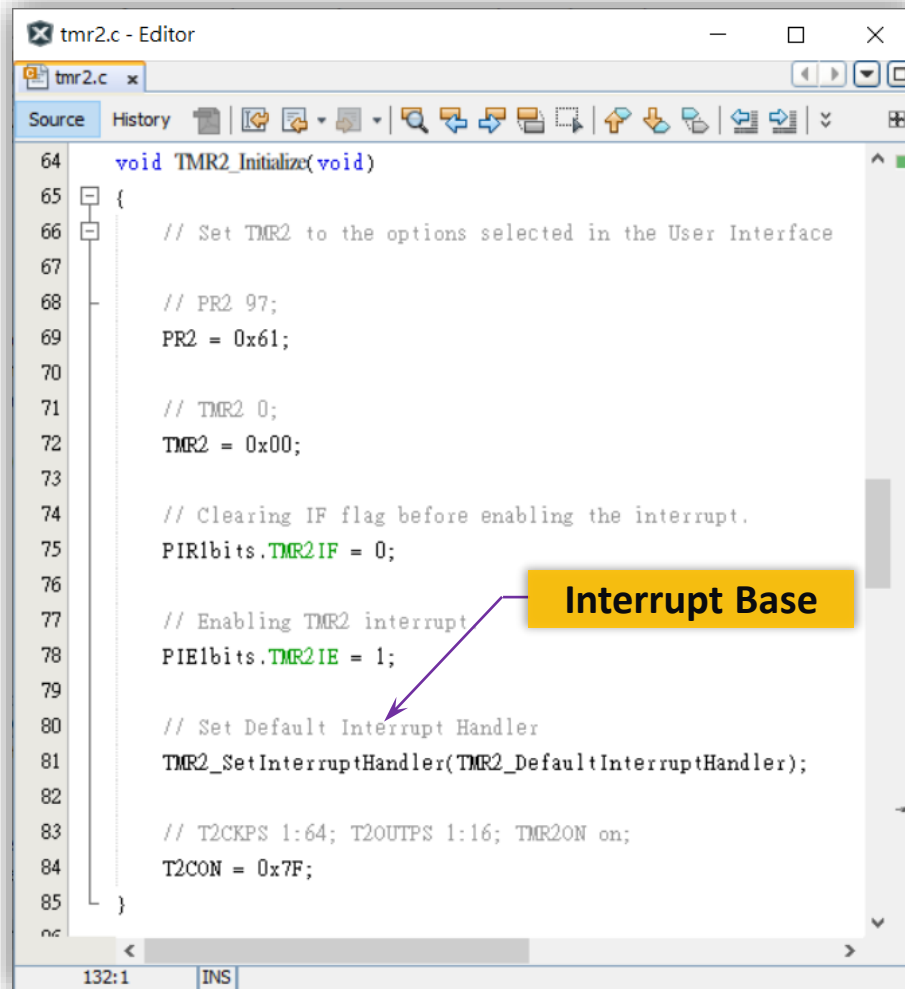
☐ Enable Timer Interrupt

**Just One Click !  
Enable Interrupt**



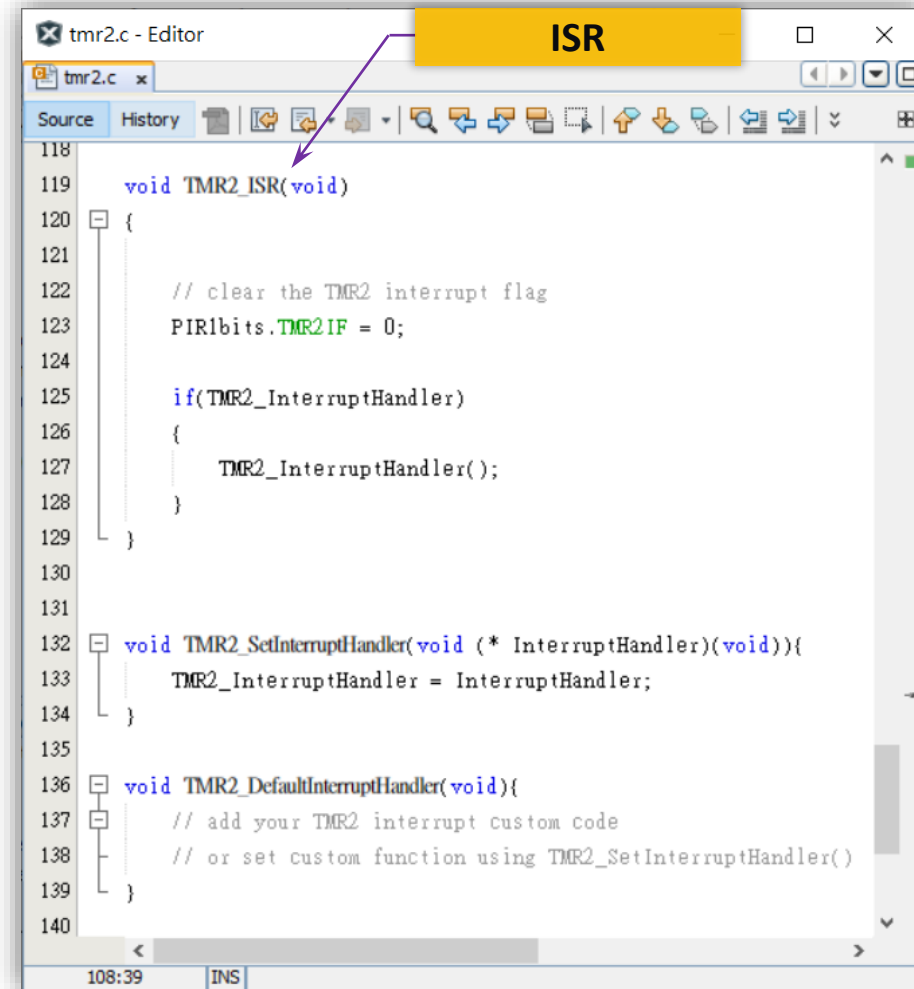
# MCC's Interrupt Function

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



```
64 void TMR2_Initialize(void)
65 {
66     // Set TMR2 to the options selected in the User Interface
67
68     // PR2 97;
69     PR2 = 0x61;
70
71     // TMR2 0;
72     TMR2 = 0x00;
73
74     // Clearing IF flag before enabling the interrupt.
75     PIR1bits.TMR2IF = 0;
76
77     // Enabling TMR2 interrupt
78     PIR1bits.TMR2IE = 1;
79
80     // Set Default Interrupt Handler
81     TMR2_SetInterruptHandler(TMR2_DefaultInterruptHandler);
82
83     // T2CKPS 1:64; T2OUTPS 1:16; TMR2ON on;
84     T2CON = 0x7F;
85 }
```

132:1 | INS

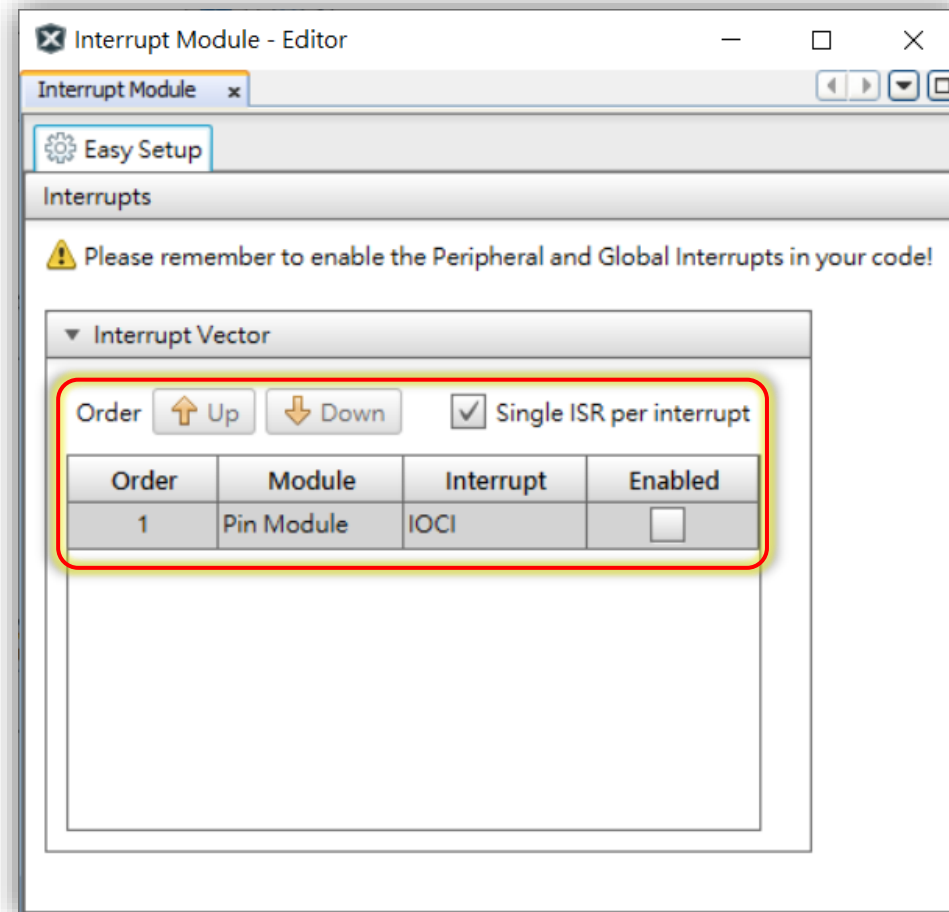


```
118
119 void TMR2_ISR(void)
120 {
121
122     // clear the TMR2 interrupt flag
123     PIR1bits.TMR2IF = 0;
124
125     if(TMR2_InterruptHandler)
126     {
127         TMR2_InterruptHandler();
128     }
129 }
130
131
132 void TMR2_SetInterruptHandler(void (* InterruptHandler)(void)){
133     TMR2_InterruptHandler = InterruptHandler;
134 }
135
136 void TMR2_DefaultInterruptHandler(void){
137     // add your TMR2 interrupt custom code
138     // or set custom function using TMR2_SetInterruptHandler()
139 }
140
```

108:39 | INS

# MCC's Interrupt Manager

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



# Timer Architecture

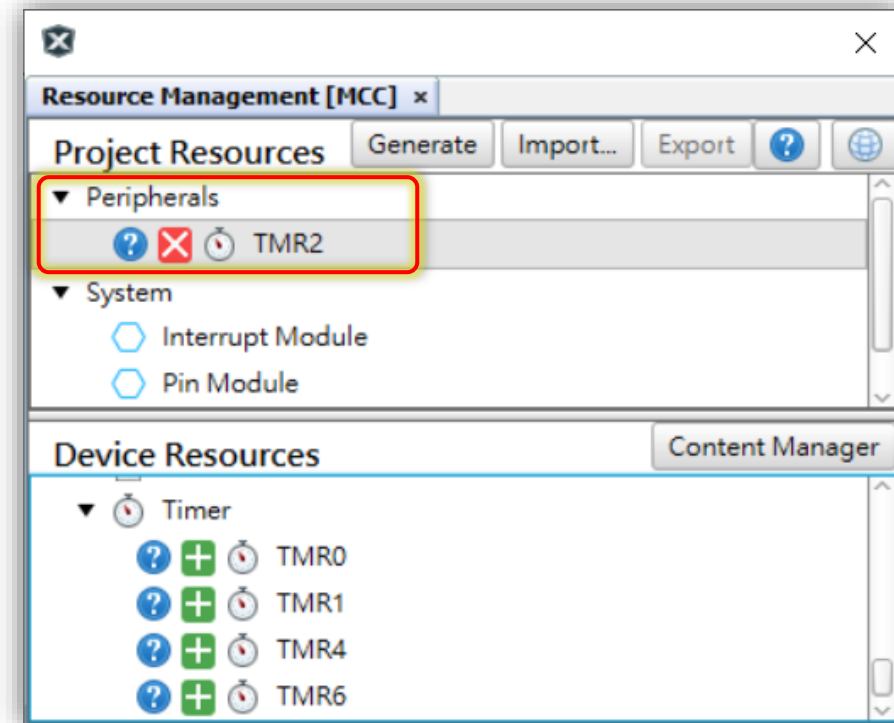
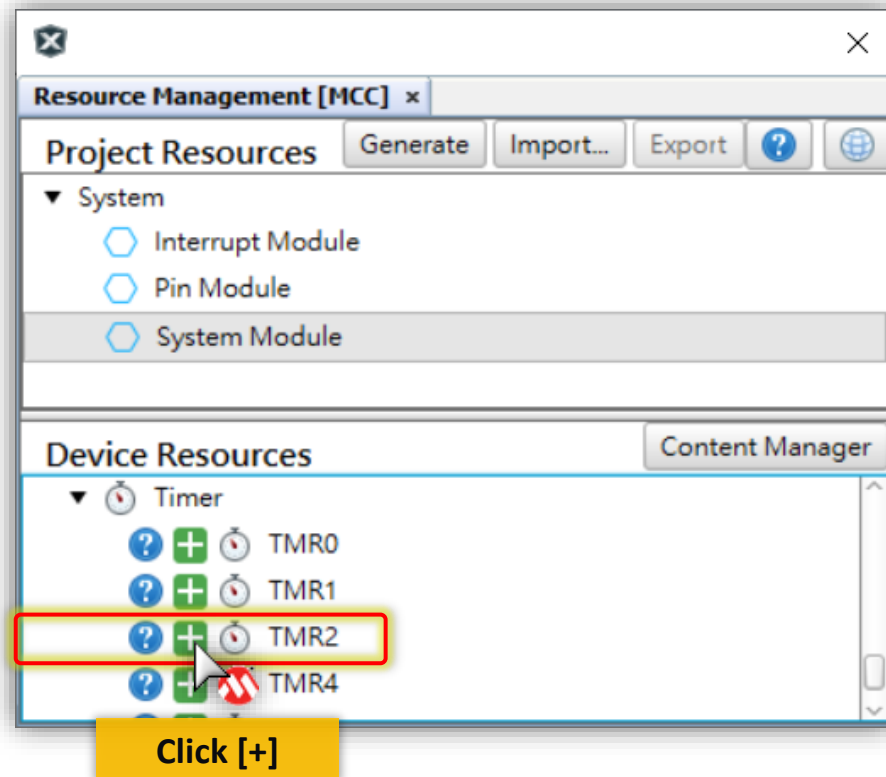
---

# 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.
- **PIC16F1937 provide one 16-bit timer (TMR1) and four 8-bits timers.**

# MCC's Timer Module

- Add Timer resource from Device Resources.



# MCC's Timer Module

The screenshot shows the TMR2 - Editor window with the following settings:

- Hardware Settings**
  - ☒ Enable Timer
  - Timer Clock**
    - Postscaler: 1:1
    - Prescaler: 1:1
  - Timer Period**
    - Timer Period:  $8\text{ us} \leq 2.048\text{ ms} \leq 2.048\text{ ms}$
    - Actual Period: 2.048 ms (Period calculated via Timer Period)
  - ☐ Enable Timer Interrupt
- Software Settings**
  - Callback Function Rate: 0x0 x Time Period = 0.0 ns

Callouts from yellow boxes point to the following settings:

- Module Enable** points to the ☒ Enable Timer checkbox.
- Postscaler** points to the Postscaler dropdown menu (1:1).
- Enable Interrupt** points to the ☐ Enable Timer Interrupt checkbox.
- Rate** points to the Callback Function Rate input field (0x0).
- Prescaler** points to the Prescaler dropdown menu (1:1).
- Period** points to the Timer Period input field (2.048 ms).

# Timer Functions of MCC

- MCC Provide below functions for Timer:

Prototype definition : "tmrn.h"

- `void TMRn_Initialize (void);`
- `void TMRn_Start (void);`
- `void TMRn_Stop (void);`
- `void TMRn_WriteTimer (uint8_t timerVal);`
- `uint8_t TMRn_ReadTimer (void);`
- `void TMRn_LoadPeriodRegister (uint8_t periodVal);`
- `void TMRn_SetInterruptHandler (void (* InterruptHandler)(void));`

# Function Rate

- 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 TMR2_ISR(void)
{
    static volatile unsigned int CountCallBack = 0;

    PIR1bits.TMR2IF = 0;

    if (++CountCallBack >= TMR2_INTERRUPT_TICKER_FACTOR)
    {
        TMR2_CallBack();
        CountCallBack = 0;
    }
}
```

```
void TMR2_CallBack(void)
{
    // Add your custom callback code here
    if(TMR2_InterruptHandler)
    {
        TMR2_InterruptHandler();
    }
}
```

```
void TMR2_SetInterruptHandler(void (* InterruptHandler) void))
{
    TMR2_InterruptHandler = InterruptHandler;
}
```

| Software Settings      |                                |
|------------------------|--------------------------------|
| Callback Function Rate | 0x2 x Time Period = 200.704 ms |

extend your timer period by software artificie.

# Lab4 Timer2 Interrupt

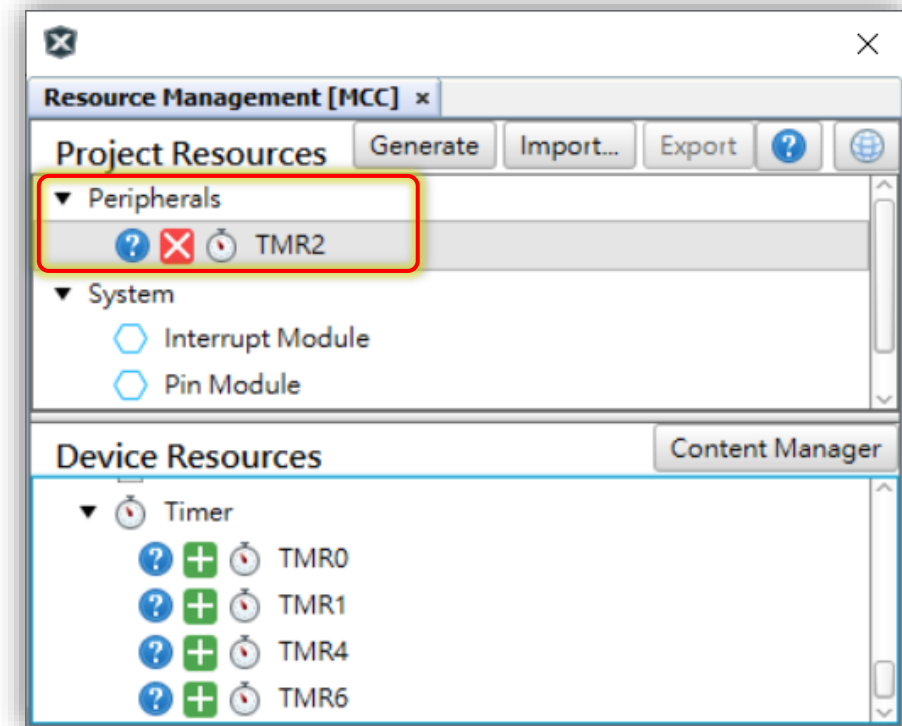
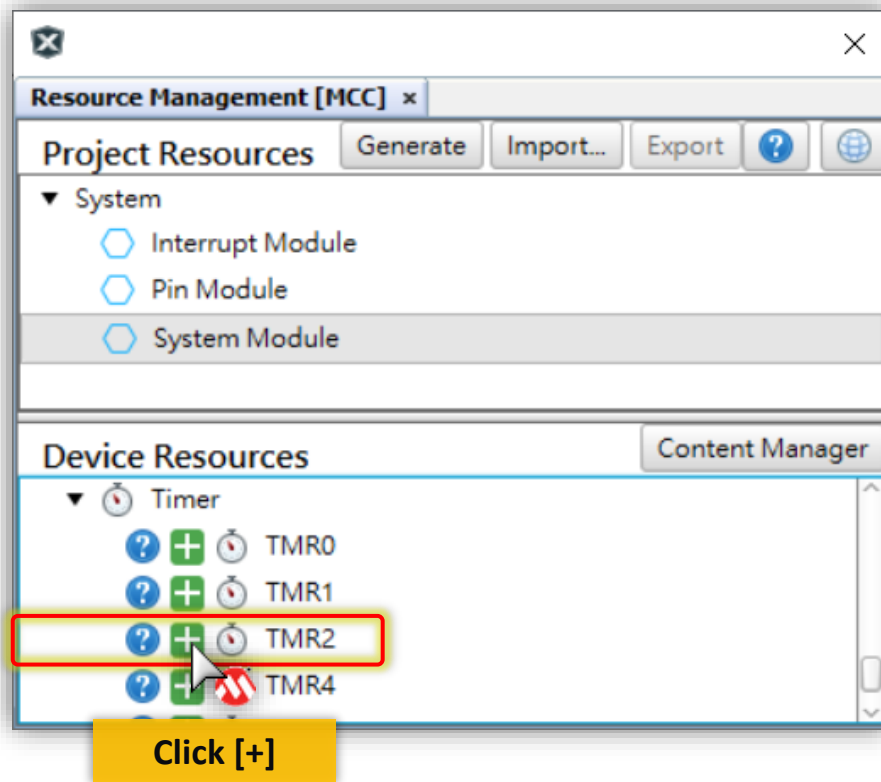
- Base on current project or Open `.\Lab4.2_XXX.X` to do exercises
- Try to add TMR2 to replace software delay to control LEDs toggle period.
- The Timer2 setup to timer mode, timer period is 200ms.

# Let's go!

# Lab4 Timer2 Interrupt

## Step 1

- Device Resources
  - **Timer ▶ TMR2**



# Lab4 Timer2 Interrupt

## Step 2

- Prescaler
  - 1:1 ► 1:64

TMR2 - Editor

TMR2 x

Easy Setup Registers

Hardware Settings

☒ Enable Timer

Timer Clock

Postscaler 1:1 ▼

Prescaler 1:64 ▼

Timer Period

Timer Period 64 us ≤ 256 us ≤ 16.384 ms

Actual Period 256 us (Period calculated via Timer Period)

☐ Enable Timer Interrupt

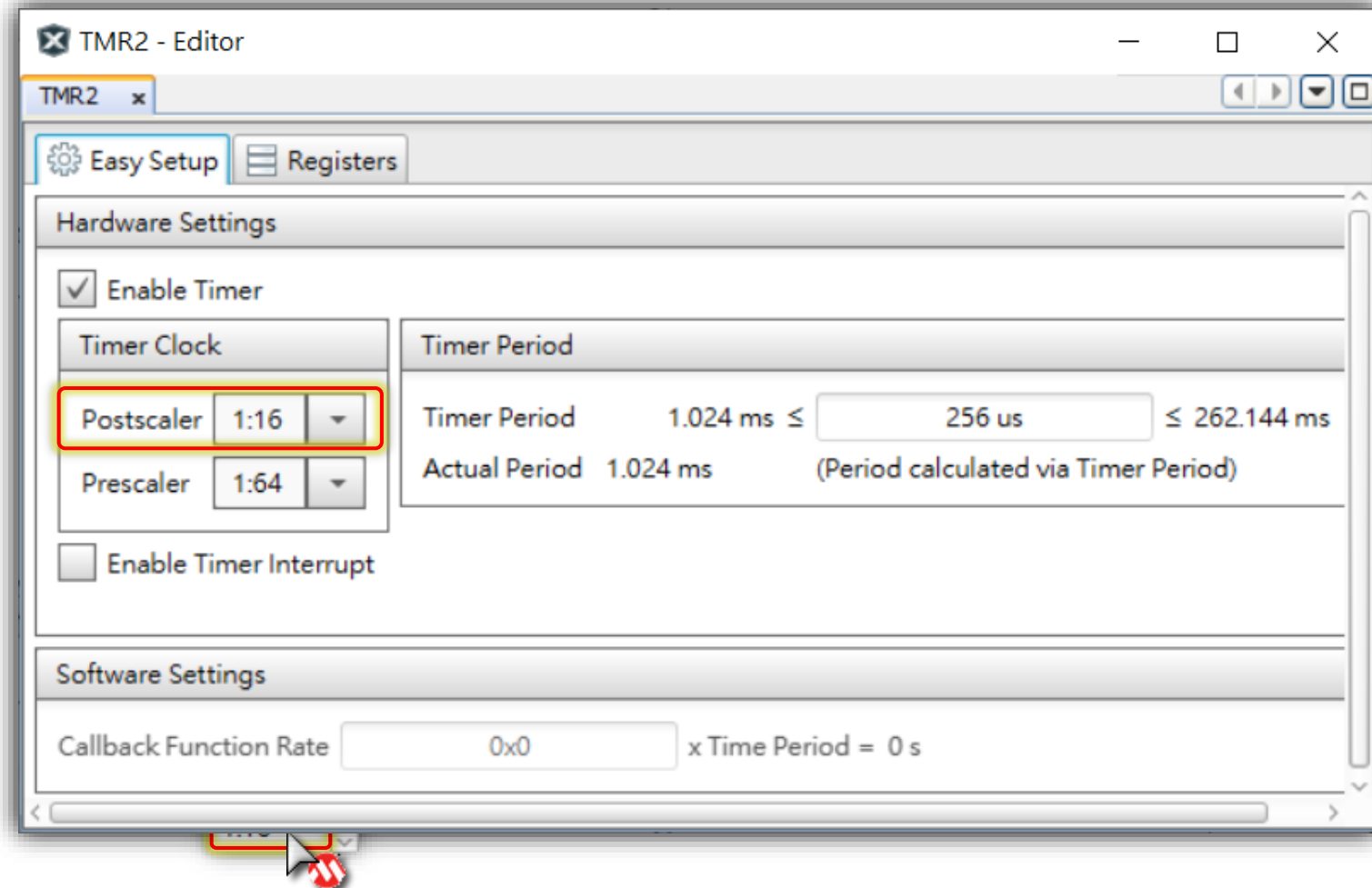
Software Settings

Callback Function Rate 0x0 x Time Period = 0 s

# Lab4 Timer2 Interrupt

## Step 3

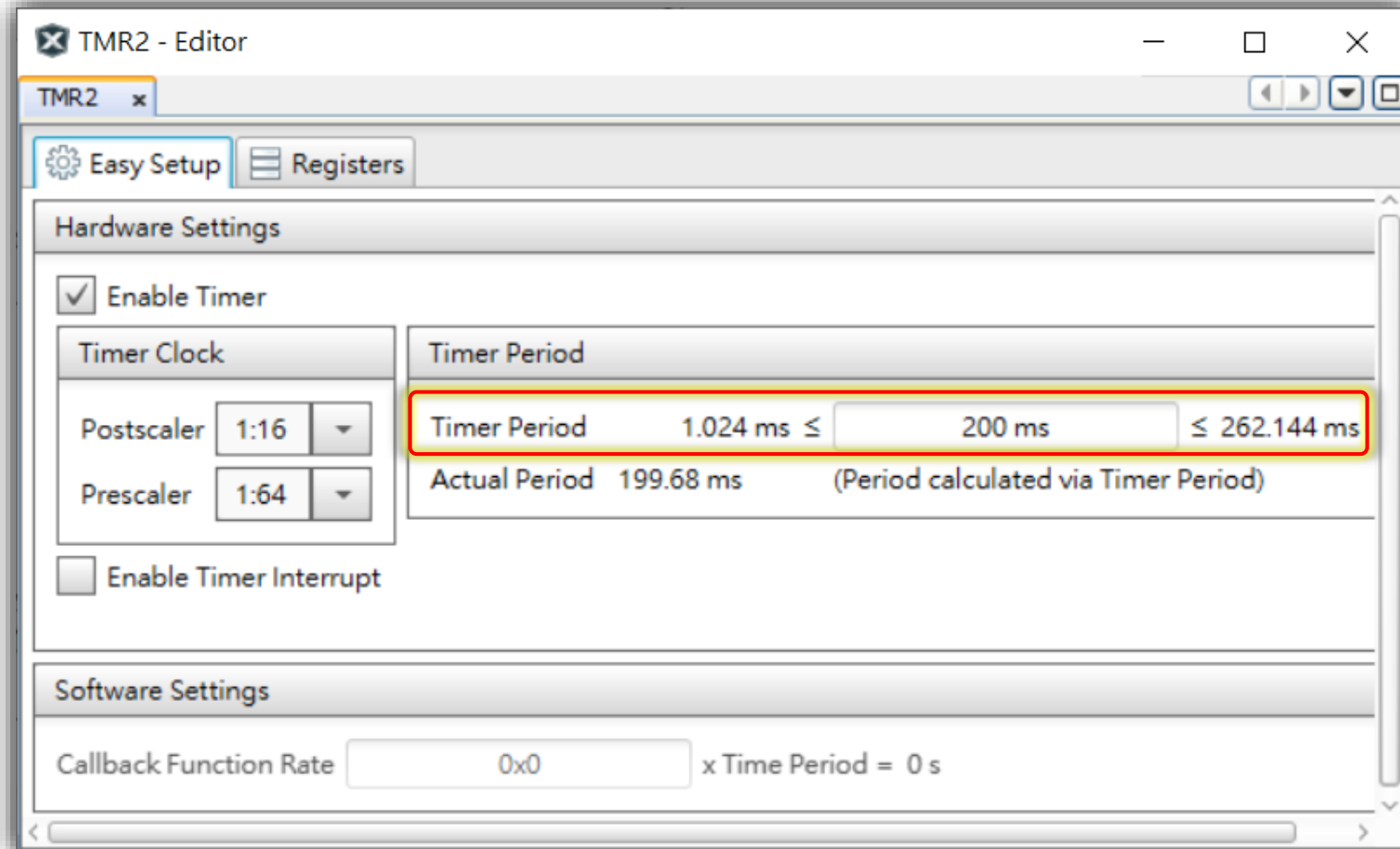
- Postscaler
  - 1:1 ► 1:16



# Lab4 Timer2 Interrupt

## Step 4

- Timer Period
  - 256us ► 200ms



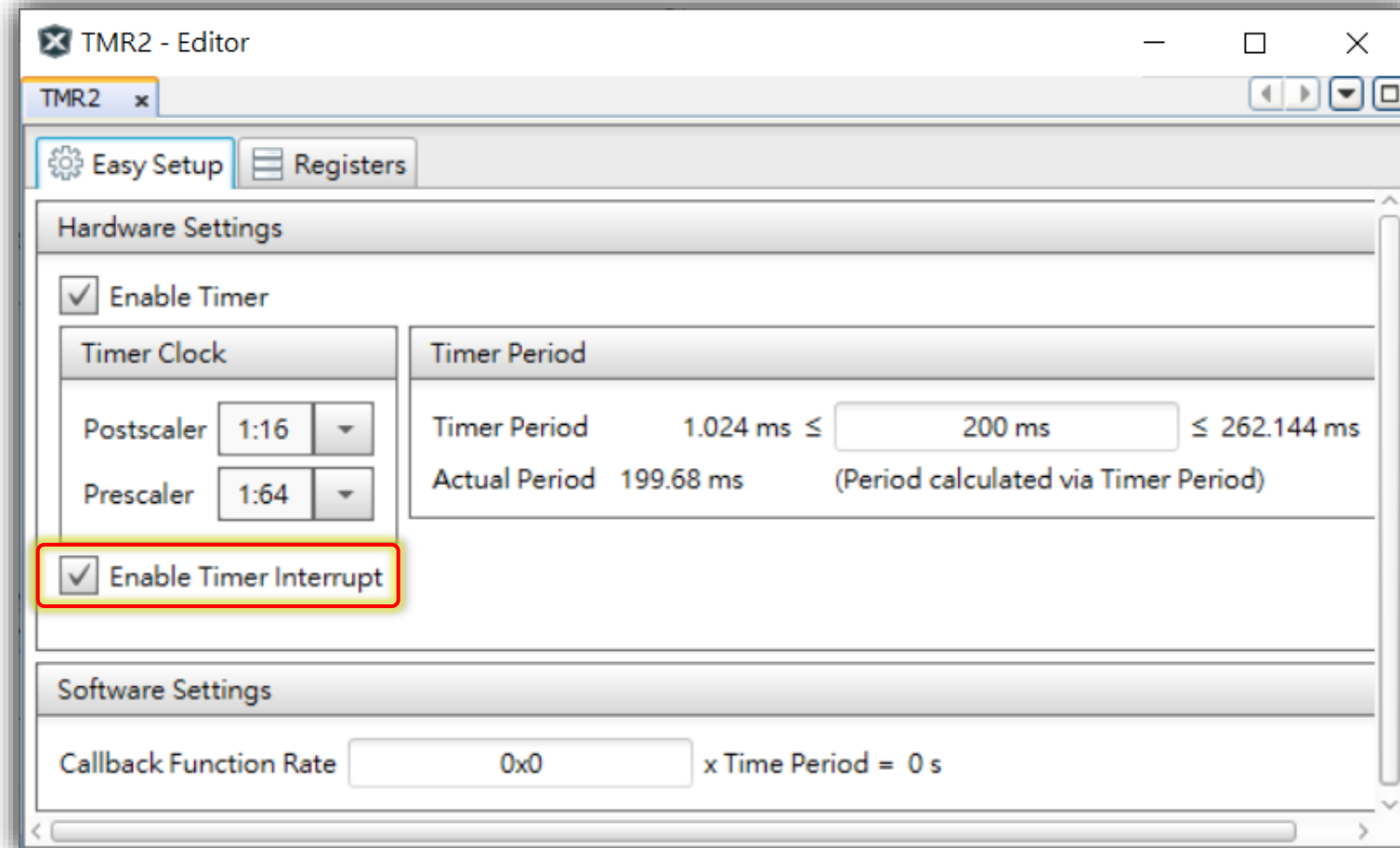
The screenshot shows the 'TMR2 - Editor' window with the 'Easy Setup' tab selected. Under 'Hardware Settings', the 'Enable Timer' checkbox is checked. The 'Timer Clock' section shows 'Postscaler' set to 1:16 and 'Prescaler' set to 1:64. The 'Timer Period' section is highlighted with a red box, showing a range from 1.024 ms to 262.144 ms, with a value of 200 ms entered. Below this, the 'Actual Period' is calculated as 199.68 ms. The 'Enable Timer Interrupt' checkbox is unchecked. The 'Software Settings' section shows the 'Callback Function Rate' set to 0x0, resulting in a time period of 0 s.

| Parameter                | Value                 |
|--------------------------|-----------------------|
| Timer Period Range       | 1.024 ms ≤ 262.144 ms |
| Timer Period (Set)       | 200 ms                |
| Actual Period            | 199.68 ms             |
| Callback Function Rate   | 0x0                   |
| Time Period (Calculated) | 0 s                   |

# Lab4 Timer2 Interrupt

## Step 5

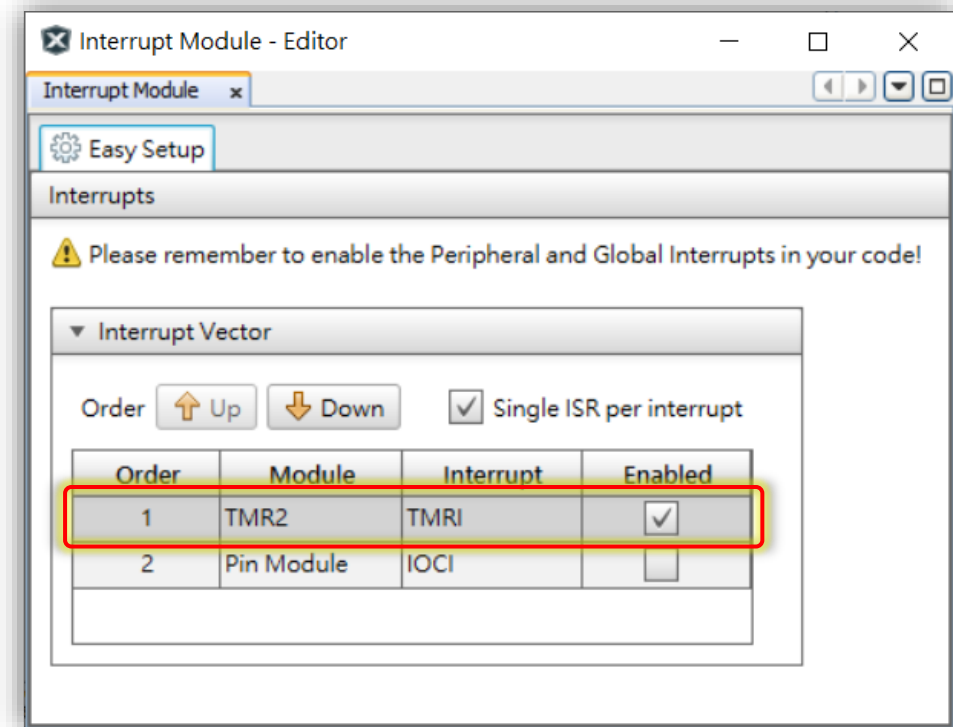
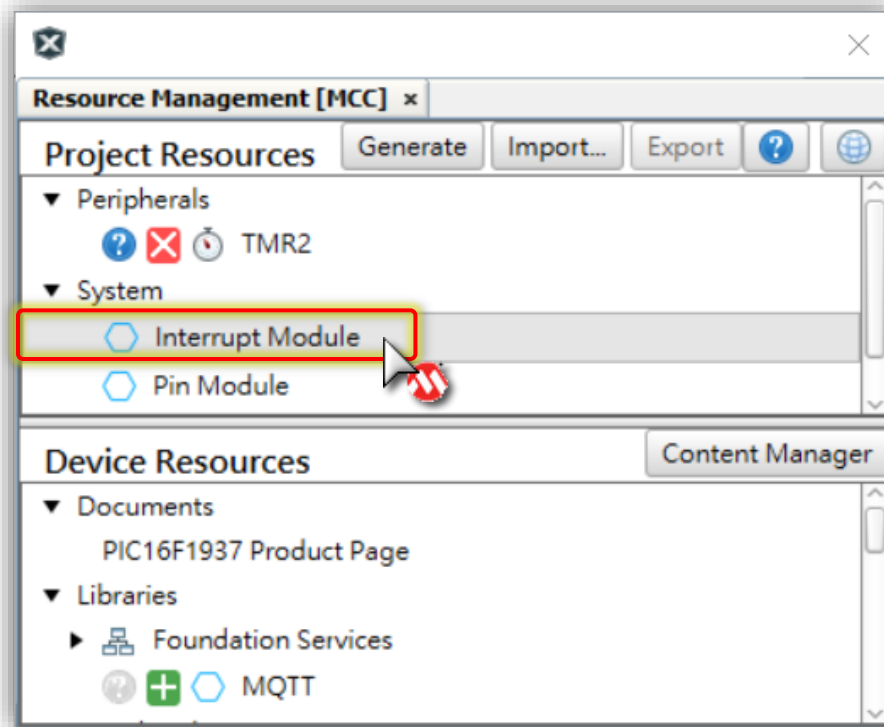
- **Enable Timer Interrupt.**



# Lab4 Timer2 Interrupt

## Step 6

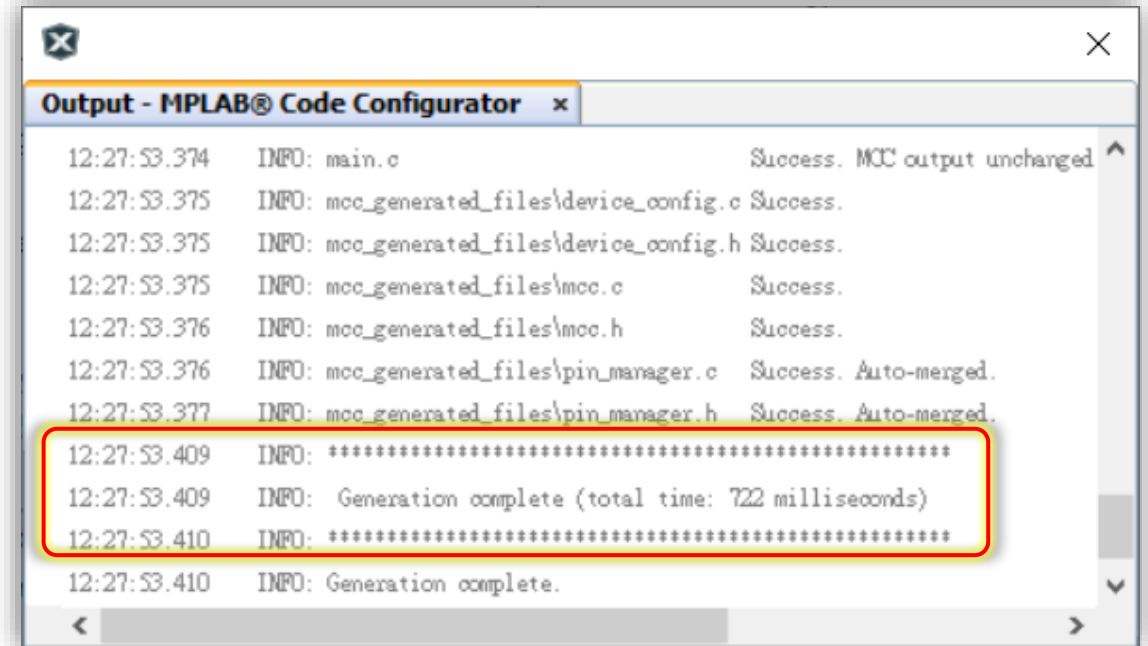
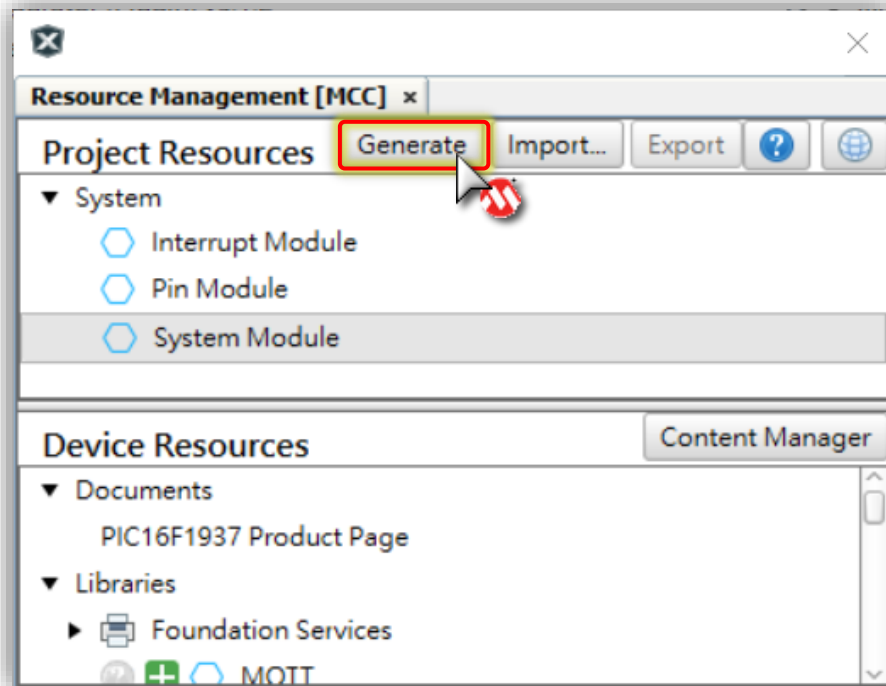
- **Confirm enable TMR2 Interrupt.**



# Lab4 Timer2 Interrupt

## Step 7

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



# Lab4 Timer2 Interrupt

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"

void TIM2_ISR(void);

uint32_t i = 0;

int main(void)
{
    ...

    // Enable the Global Interrupts
    INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    INTERRUPT_PeripheralInterruptEnable();
    ...
    //INTERRUPT_PeripheralInterruptDisable();

    TMR2_SetInterruptHandler(TIM2_ISR);
    while (1)
    {
        ...
    }
}

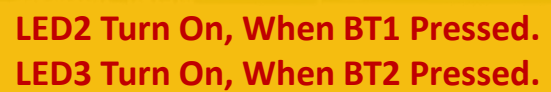
void TIM2_ISR(void)
{
    LED1_Toggle();
}
```

```
int main(void)
{
    ...
    //INTERRUPT_PeripheralInterruptDisable();

    TMR2_SetInterruptHandler(TIM2_ISR);
    while (1)
    {
        // Add your application code
        LED1_Toggle();
        for (i = 0; i < 2800; i++);

        if(!BT1_GetValue())
        {
            LED2_SetLow();
        }
        else if(!BT2_GetValue())
        {
            LED3_SetLow();
        }
        else
        {
            LED2_SetHigh();
            LED3_SetHigh();
        }
    }
}
```

## Result



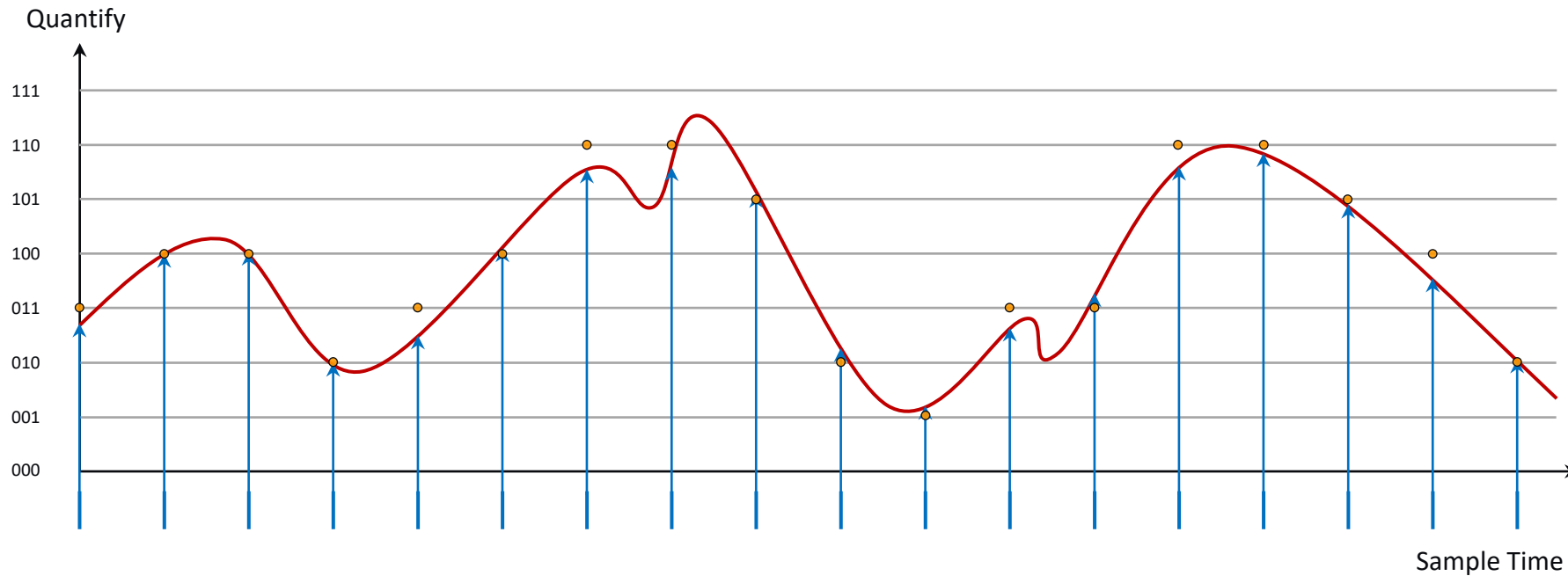
**LED1 Toggle,  
Interval Period use "Timer2" callback.**

# ADC Architecture

---

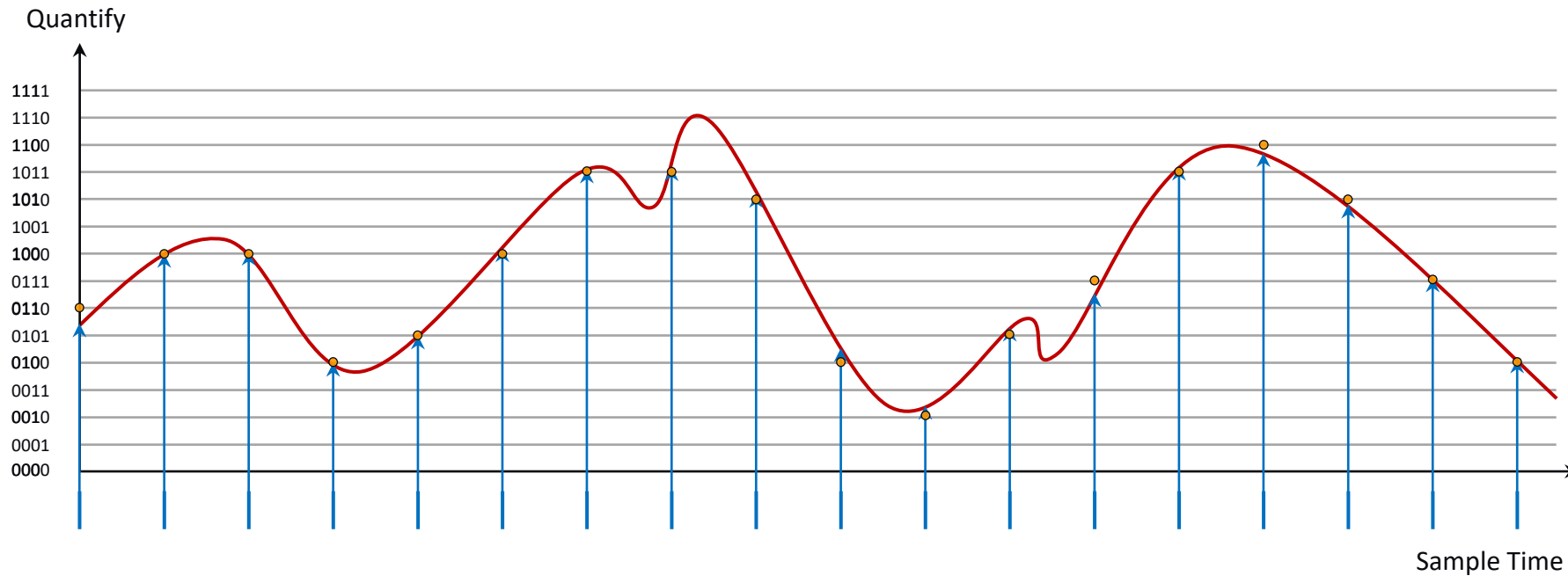
# What's ADC ?

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



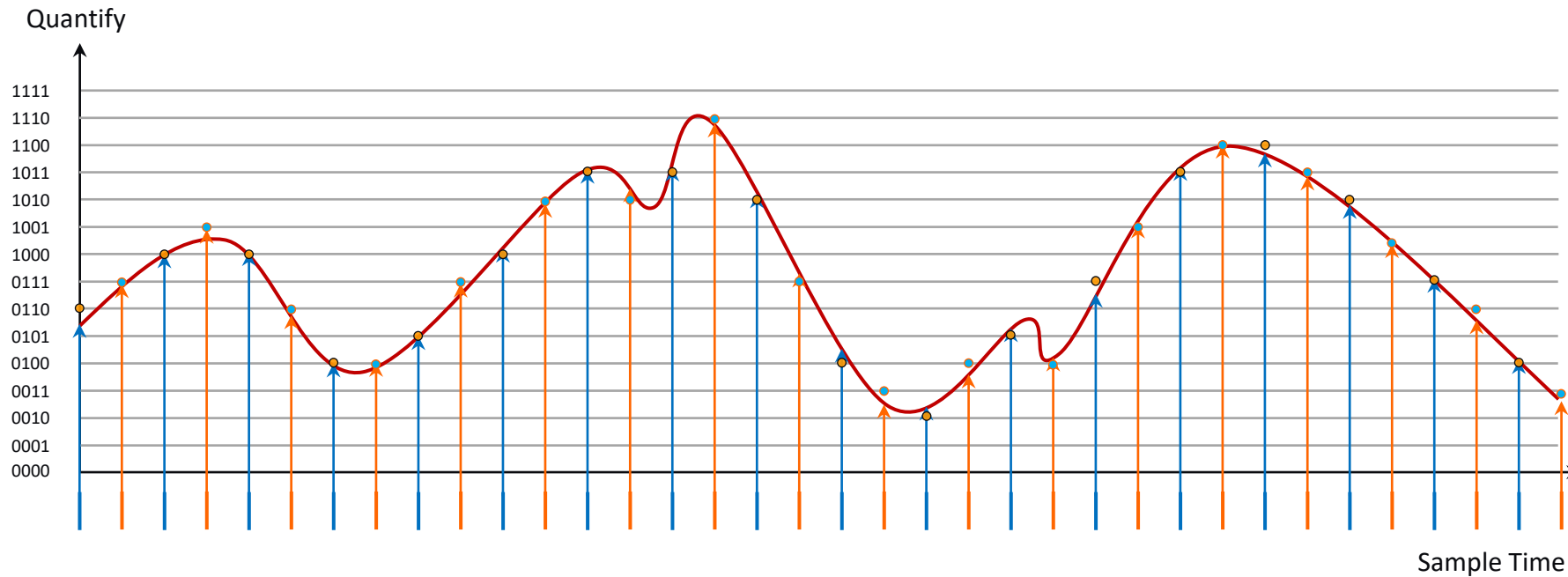
# Resolution

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



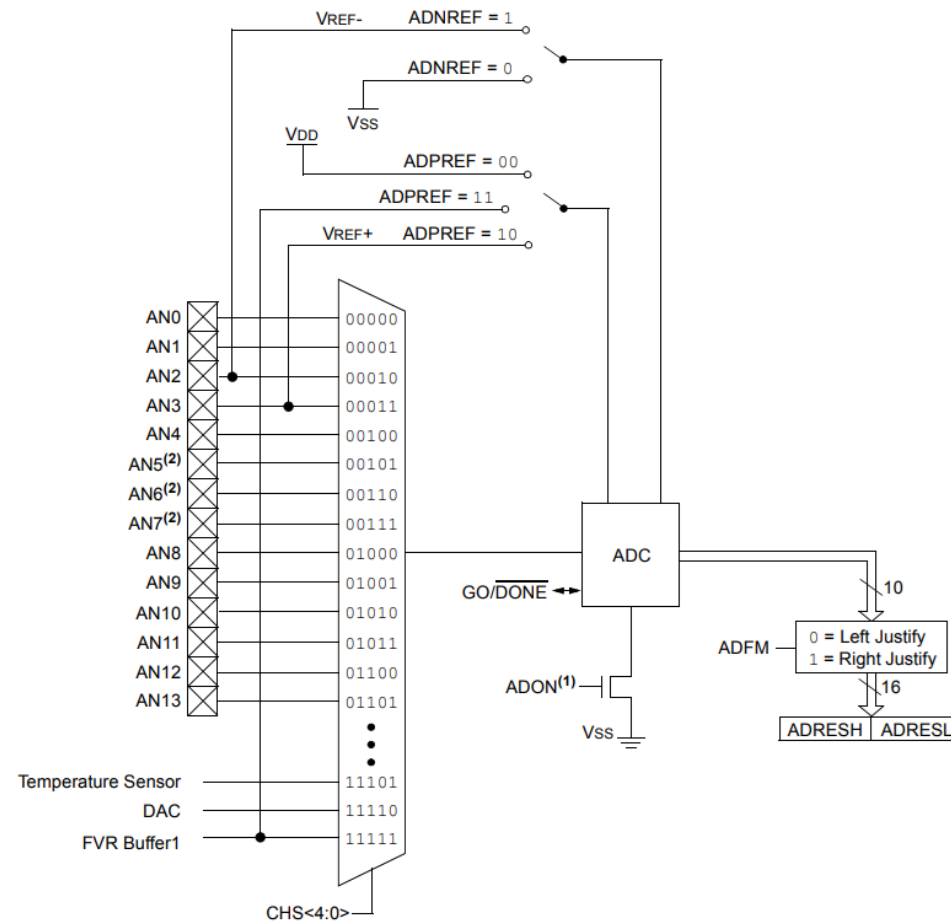
# Sample Rate

- 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)



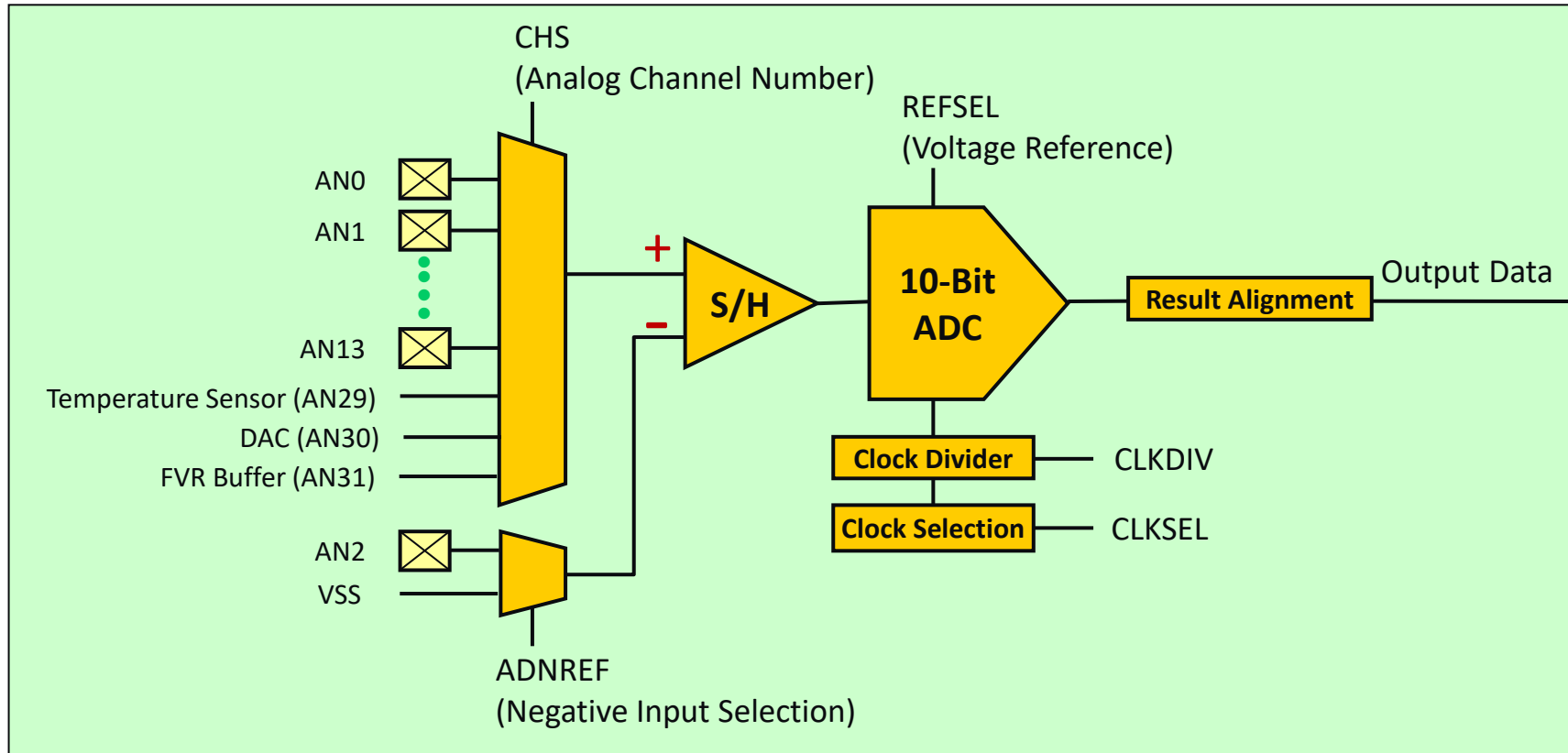
# PIC16F Series ADC Features

- Analog-to-Digital Converter (ADC) includes the following features:
  - **10-bit resolution** and up to **14 channels analog inputs** multiplexed into one A/D converter.
  - **5  $\mu$ s typical sampling time**, as fast as **11  $\mu$ s conversion time**.
  - A/D sampling frequency up to **62.5 kHz**.
  - External reference inputs, VREF+ & VREF-.
  - A/D Result can be left or right justified.
  - 10-bit resolution with  $\pm 1$  LSb accuracy.
  - A/D conversion during SLEEP.
  - A/D complete can wake processor.



# ADC Channel Selection

- 14 analog channel inputs to connect as Analog Positive input of S/H.
- AN2 or VSS as Analog Negative input of the S/H.
- The differential input voltage is **Positive – Negative**.



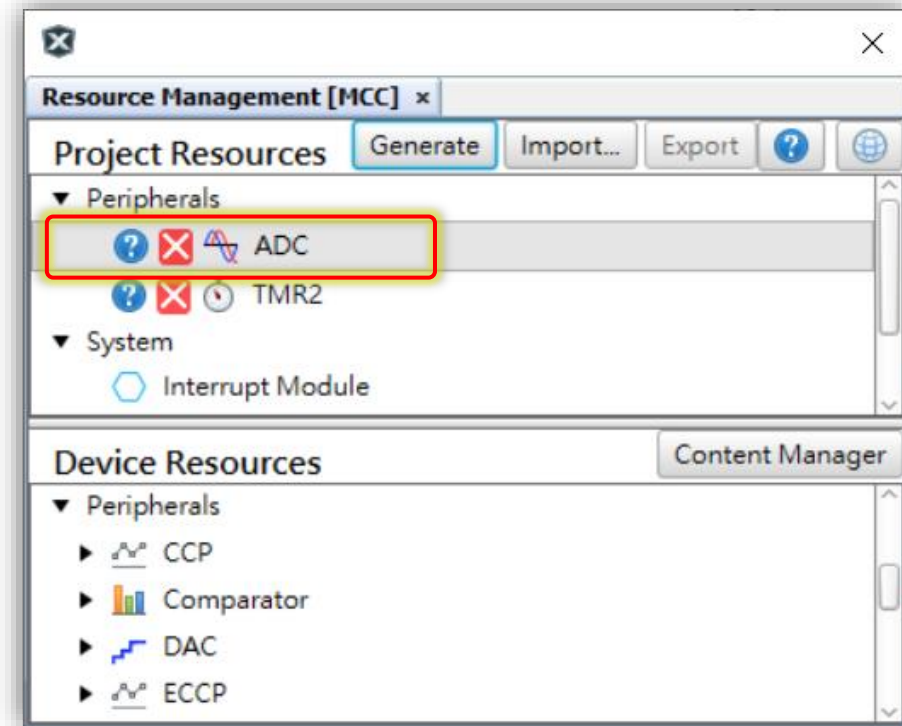
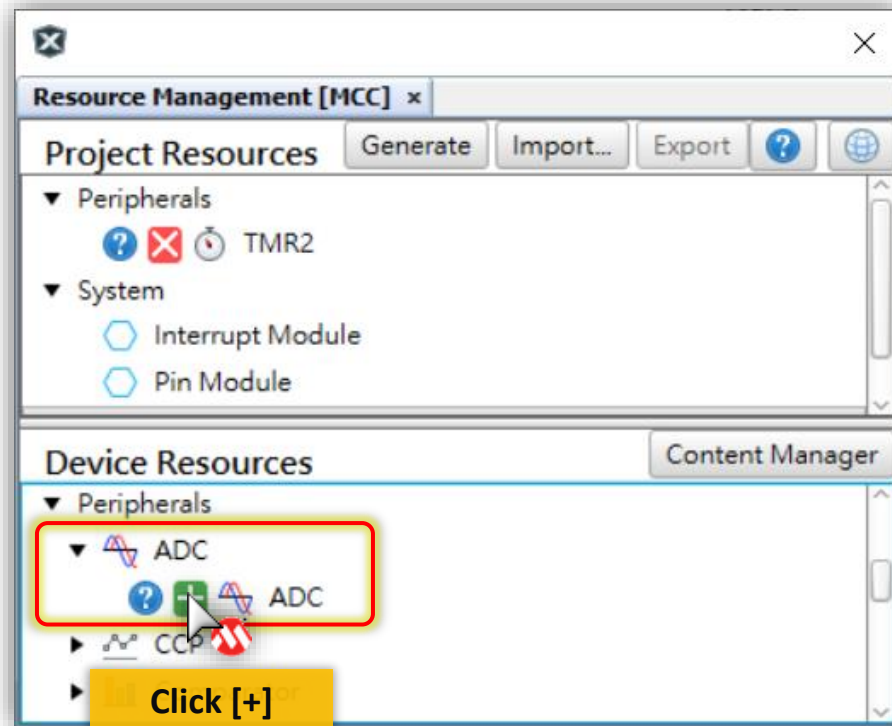
# Sample and Conversion Sequence

- The ADC sample and conversion time are requirements from the specification.
- The conversion time need 11 TAD, TAD time range 1us to 6us.

| ADC Clock Period (TAD) |           | Device Frequency (Fosc)<br>Device Frequency (Fosc) |                             |                             |                             |                             |                             |
|------------------------|-----------|----------------------------------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| ADC Clock Source       | ADCS<2:0> | 32 MHz                                             | 20 MHz                      | 16 MHz                      | 8 MHz                       | 4 MHz                       | 1 MHz                       |
| Fosc/2                 | 000       | 62.5ns <sup>(2)</sup>                              | 100 ns <sup>(2)</sup>       | 125 ns <sup>(2)</sup>       | 250 ns <sup>(2)</sup>       | 500 ns <sup>(2)</sup>       | 2.0 µs                      |
| Fosc/4                 | 100       | 125 ns <sup>(2)</sup>                              | 200 ns <sup>(2)</sup>       | 250 ns <sup>(2)</sup>       | 500 ns <sup>(2)</sup>       | 1.0 µs                      | 4.0 µs                      |
| Fosc/8                 | 001       | 0.5 µs <sup>(2)</sup>                              | 400 ns <sup>(2)</sup>       | 0.5 µs <sup>(2)</sup>       | 1.0 µs                      | 2.0 µs                      | 8.0 µs <sup>(3)</sup>       |
| Fosc/16                | 101       | 800 ns                                             | 800 ns                      | 1.0 µs                      | 2.0 µs                      | 4.0 µs                      | 16.0 µs <sup>(3)</sup>      |
| Fosc/32                | 010       | 1.0 µs                                             | 1.6 µs                      | 2.0 µs                      | 4.0 µs                      | 8.0 µs <sup>(3)</sup>       | 32.0 µs <sup>(3)</sup>      |
| Fosc/64                | 110       | 2.0 µs                                             | 3.2 µs                      | 4.0 µs                      | 8.0 µs <sup>(3)</sup>       | 16.0 µs <sup>(3)</sup>      | 64.0 µs <sup>(3)</sup>      |
| FRC                    | x11       | 1.0-6.0 µs <sup>(1,4)</sup>                        | 1.0-6.0 µs <sup>(1,4)</sup> | 1.0-6.0 µs <sup>(1,4)</sup> | 1.0-6.0 µs <sup>(1,4)</sup> | 1.0-6.0 µs <sup>(1,4)</sup> | 1.0-6.0 µs <sup>(1,4)</sup> |

# MCC's ADC Module

- Add ADC resource from Device Resources.



# MCC's ADC Module

The screenshot shows the 'ADC - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section contains the following controls:

- Enable ADC:** A checked checkbox, indicated by an arrow from the 'Enable ADC' label.
- ADC Clock:** A section containing:
  - Clock Source:** A dropdown menu set to 'FOSC/2', indicated by an arrow from the 'Clock Source' label.
  - Timing:** 1 TAD = 500.0 ns, Sampling Frequency = 173.913 kHz, Conversion Time = 11.5 \* TAD = 5.75 us.
- Enable ADC Interrupt:** An unchecked checkbox, indicated by an arrow from the 'Enable Interrupt' label.
- Result Alignment:** A dropdown menu set to 'left', indicated by an arrow from the 'Result Alignment' label.
- Positive Reference:** A dropdown menu set to 'VDD'.
- Negative Reference:** A dropdown menu set to 'VSS'.

The 'Selected Channels' section contains a table with the following data:

| Pin              | Channel | Custom Name  |
|------------------|---------|--------------|
| Internal Channel | FVR     | channel_FVR  |
| Internal Channel | Temp    | channel_Temp |
| Internal Channel | DAC     | channel_DAC  |
| RA0              | AN0     | channel_AN0  |

An arrow from the 'Channel Select' label points to the first row of this table.

# ADC Functions of MCC

- MCC Provide below functions for ADC:

Prototype definition : "adc.h"

- `void ADC_Initialize(void);`
- `void ADC_StartConversion(void);`
- `void ADC_SelectChannel(adc_channel_t channel);`
- `adc_result_t ADC_GetConversion(adc_channel_t channel);`
- `adc_result_t ADC_GetConversionResult(void);`
- `bool ADC_IsConversionDone(void);`

# Lab5 ADC Channel Periodically

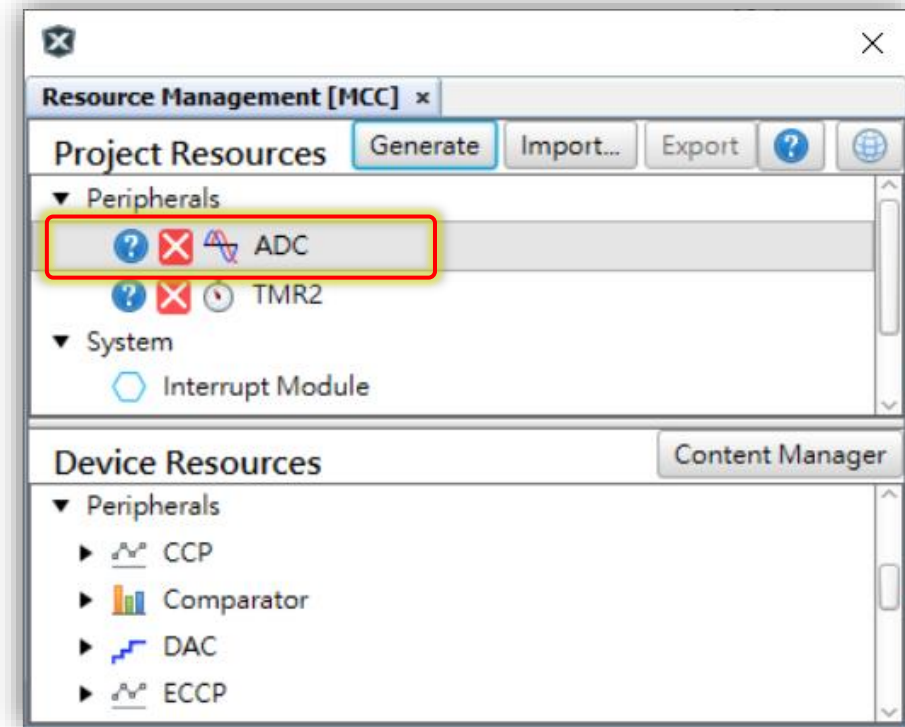
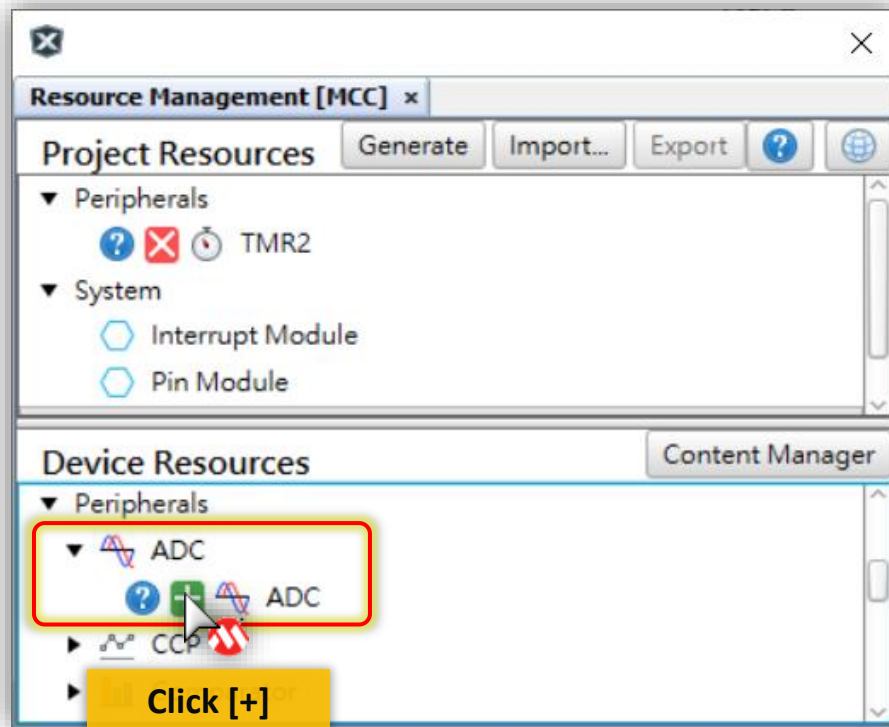
- Base on current project or Open `.\Lab5_xxx.X` to do exercises
- Try to add ADC module and conversion Potentiometer (VR2) voltage.
  - **ADC Result  $\geq 512$** , turn on **LED2**.
  - **ADC Result  $< 512$** , turn off **LED2**.
- " **Clock Source** " ► " **Fosc/4** "
- " **Result Alignment** " ► " **Right** "
- " **Channel Select** " ► " **AN8(RB2)** "

# Let's go!

# Lab5 ADC Channel Periodically

## Step 1

- Device Resources
  - **ADC ▶ ADC**



# Lab5 ADC Channel Periodically

## Step 2

- Clock Source
  - $F_{osc}/2 \rightarrow F_{osc}/4$

ADC - Editor

ADC x

Easy Setup Registers

Hardware Settings

☒ Enable ADC

ADC Clock

Clock Source FOSC/4

Result Alignment left

Positive Reference VDD

Negative Reference VSS

1 TAD 1.0 us

Sampling Frequency 86.9565 kHz

Conversion Time 11.5 \* TAD = 11.5 us

☐ Enable ADC Interrupt

Selected Channels

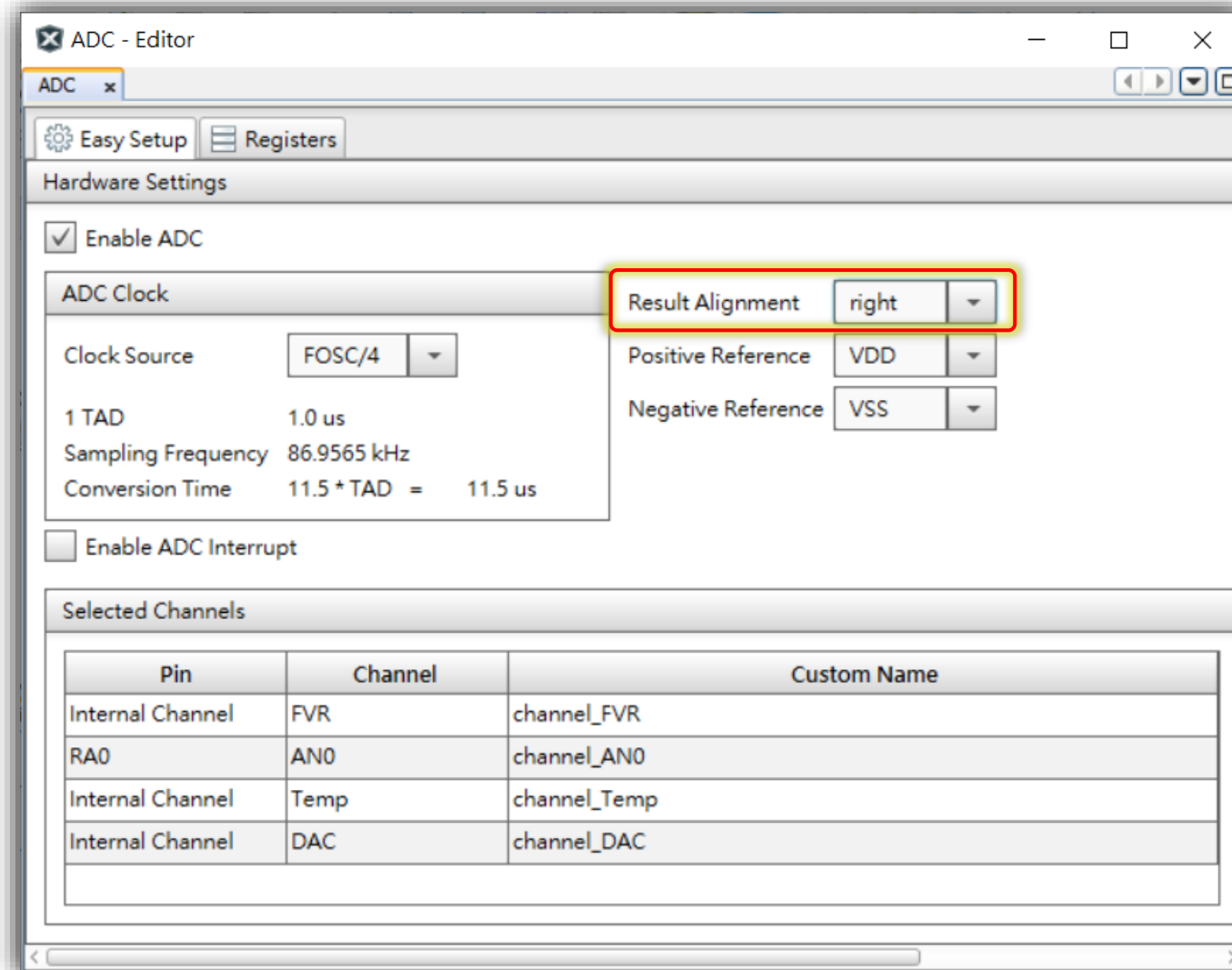
| Pin              | Channel | Custom Name  |
|------------------|---------|--------------|
| Internal Channel | FVR     | channel_FVR  |
| RA0              | AN0     | channel_AN0  |
| Internal Channel | Temp    | channel_Temp |
| Internal Channel | DAC     | channel_DAC  |

1TAD : 1.0us  
Sampling Time : 5us  
Conversion Time : 11us(11TAD)  
Sampling Frequency : 62.5kHz

# Lab5 ADC Channel Periodically

## Step 3

- Result alignment
  - Left ► Right



ADC - Editor

ADC x

Easy Setup Registers

Hardware Settings

☒ Enable ADC

ADC Clock

Clock Source FOSC/4

1 TAD 1.0 us

Sampling Frequency 86.9565 kHz

Conversion Time  $11.5 \times \text{TAD} = 11.5 \text{ us}$

Result Alignment right

Positive Reference VDD

Negative Reference VSS

☐ Enable ADC Interrupt

Selected Channels

| Pin              | Channel | Custom Name  |
|------------------|---------|--------------|
| Internal Channel | FVR     | channel_FVR  |
| RA0              | AN0     | channel_AN0  |
| Internal Channel | Temp    | channel_Temp |
| Internal Channel | DAC     | channel_DAC  |

# Lab5 ADC Channel Periodically

## Step 4

- Pin Manager : Grid View
  - AN0(RA0) ► disable

|                          |          |           |         |   |          |    |    |    |    |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
|--------------------------|----------|-----------|---------|---|----------|----|----|----|----|----|----|----------|---|---|----|----|----|----|----------|----|----|----|----|----|----|----|----|---|
| Pin Manager: Grid View x |          |           |         |   |          |    |    |    |    |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
| Package:                 | TQFP44   |           | Pin No: |   | 19       | 20 | 21 | 22 | 23 | 24 | 31 | 30       | 8 | 9 | 10 | 11 | 14 | 15 | 16       | 17 | 32 | 35 | 36 | 37 | 42 | 43 | 44 | 1 |
|                          |          |           |         |   | Port A ▼ |    |    |    |    |    |    | Port B ▼ |   |   |    |    |    |    | Port C ▼ |    |    |    |    |    |    |    |    |   |
| Module                   | Function | Direction | 0       | 1 | 2        | 3  | 4  | 5  | 6  | 7  | 0  | 1        | 2 | 3 | 4  | 5  | 6  | 7  | 0        | 1  | 2  | 3  | 4  | 5  | 6  | 7  |    |   |
| ADC ▼                    | ANx      | input     | 🔒       | 🔒 | 🔒        | 🔒  |    | 🔒  |    |    | 🔒  | 🔒        | 🔒 | 🔒 | 🔒  | 🔒  |    |    |          |    |    |    |    |    |    |    |    |   |
|                          | VREF+    | input     |         |   |          | 🔒  |    |    |    |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
|                          | VREF-    | input     |         |   | 🔒        |    |    |    |    |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
| OSC                      | CLKOUT   | output    |         |   |          |    |    |    | 🔒  |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
| Pin Module ▼             | GPIO     | input     | 🔒       | 🔒 | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒 | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  |   |
|                          | GPIO     | output    | 🔒       | 🔒 | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒 | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  |   |
| RESET ▼                  | MCLR     | input     |         |   |          |    |    |    |    |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |
|                          | VCAP     | input     | 🔒       |   |          |    |    | 🔒  | 🔒  |    |    |          |   |   |    |    |    |    |          |    |    |    |    |    |    |    |    |   |

# Lab5 ADC Channel Periodically

## Step 5

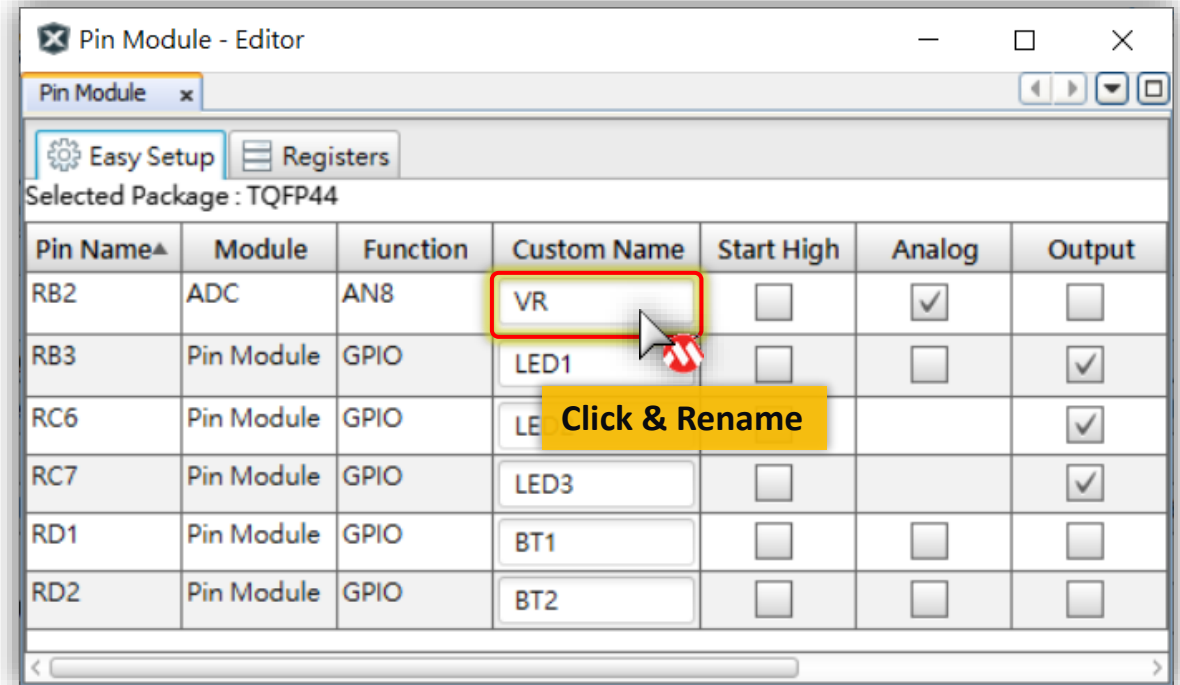
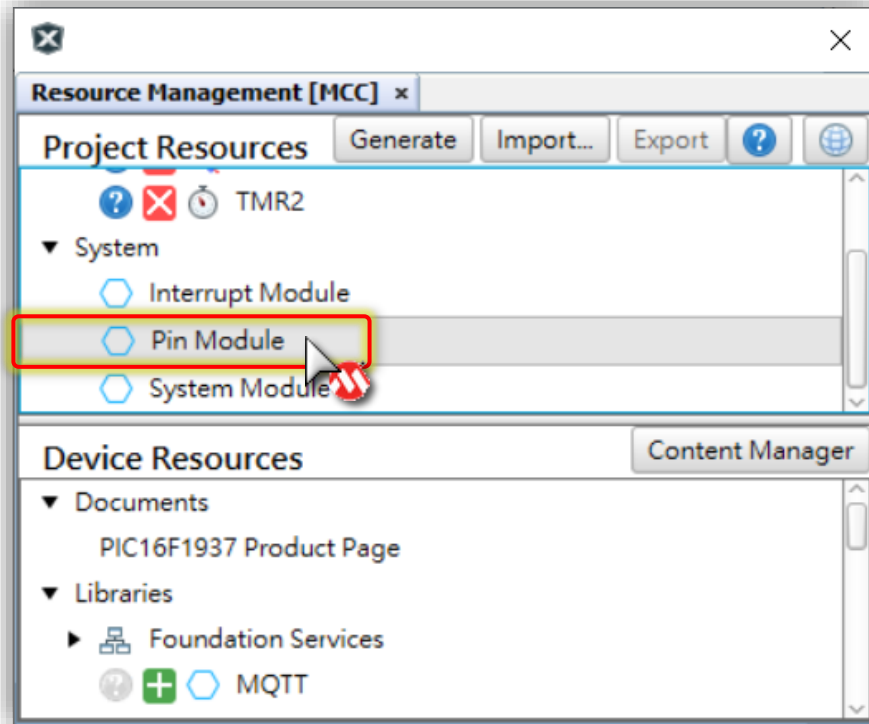
- Pin Manager : Grid View
  - AN8(RB2) ► enable

| Pin Manager: Grid View x |          |           |   |          |   |    |    |    |    |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |
|--------------------------|----------|-----------|---|----------|---|----|----|----|----|----|----------|----|----|---|---|----|----|----------|----|----|----|----|----|----|----|----|----|----|---|
| Package:                 |          | TQFP44    |   | Pin No:  |   | 19 | 20 | 21 | 22 | 23 | 24       | 31 | 30 | 8 | 9 | 10 | 11 | 14       | 15 | 16 | 17 | 32 | 35 | 36 | 37 | 42 | 43 | 44 | 1 |
|                          |          |           |   | Port A ▼ |   |    |    |    |    |    | Port B ▼ |    |    |   |   |    |    | Port C ▼ |    |    |    |    |    |    |    |    |    |    |   |
| Module                   | Function | Direction | 0 | 1        | 2 | 3  | 4  | 5  | 6  | 7  | 0        | 1  | 2  | 3 | 4 | 5  | 6  | 7        | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |    |    |   |
| ADC ▼                    | ANx      | input     | 🔒 | 🔒        | 🔒 | 🔒  |    | 🔒  |    |    | 🔒        | 🔒  | 🔒  | 🔒 | 🔒 | 🔒  |    |          |    |    |    |    |    |    |    |    |    |    |   |
|                          | VREF+    | input     |   |          |   | 🔒  |    |    |    |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |
|                          | VREF-    | input     |   |          | 🔒 |    |    |    |    |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |
| OSC                      | CLKOUT   | output    |   |          |   |    |    |    | 🔒  |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |
| Pin Module ▼             | GPIO     | input     | 🔒 | 🔒        | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒 | 🔒 | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  |   |
|                          | GPIO     | output    | 🔒 | 🔒        | 🔒 | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒 | 🔒 | 🔒  | 🔒  | 🔒        | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  | 🔒  |   |
| RESET ▼                  | MCLR     | input     |   |          |   |    |    |    |    |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |
|                          | VCAP     | input     | 🔒 |          |   |    |    | 🔒  | 🔒  |    |          |    |    |   |   |    |    |          |    |    |    |    |    |    |    |    |    |    |   |

# Lab5 ADC Channel Periodically

## Step 6

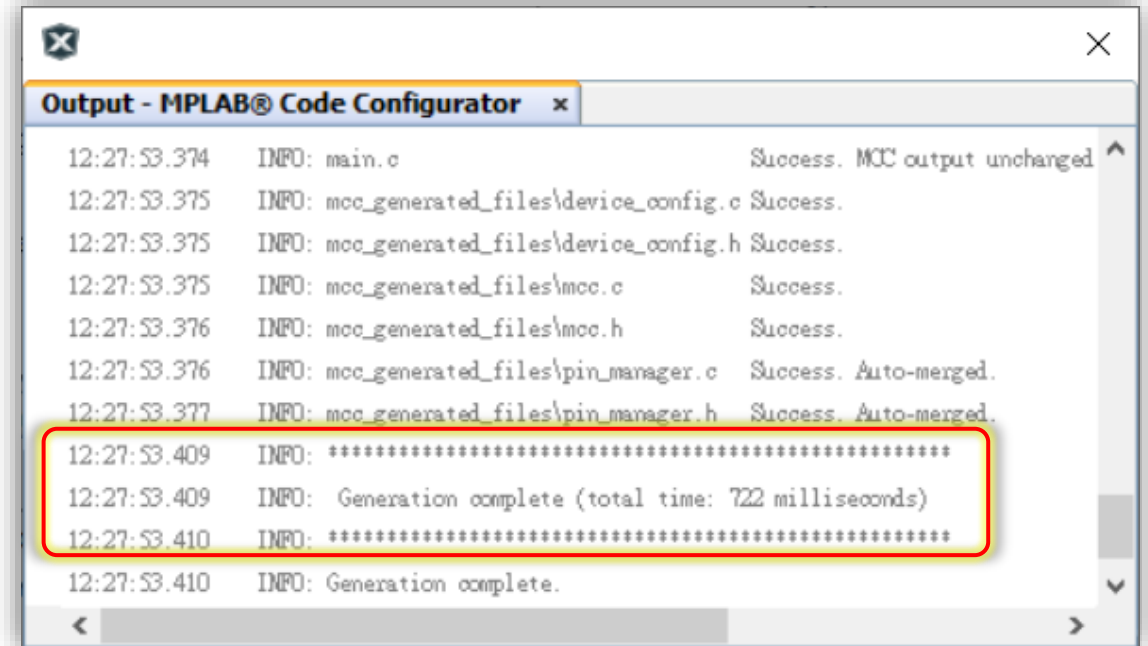
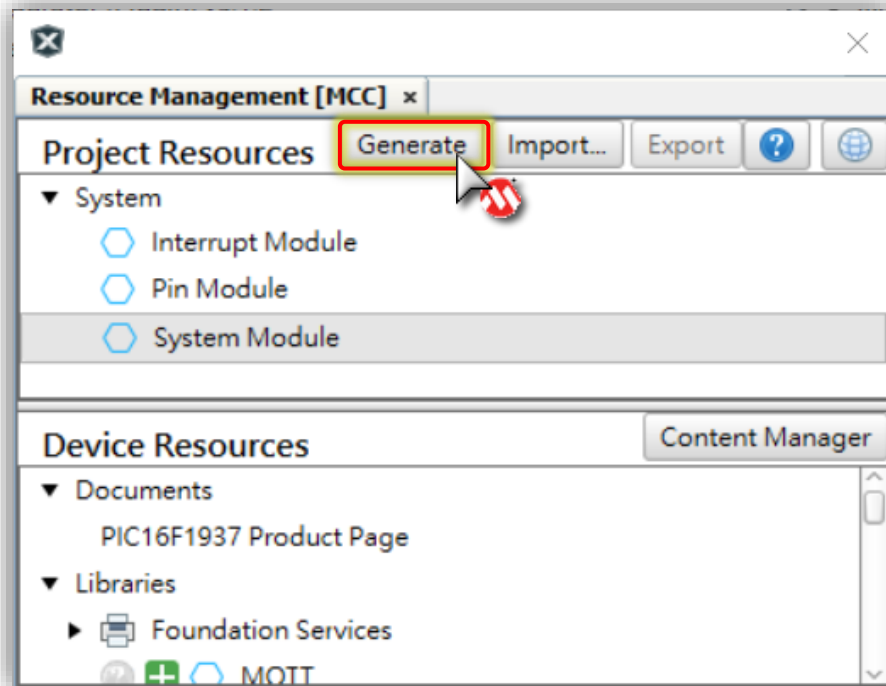
- Set Alias for RB2.
  - Pin Module : Custom Name ▶ VR.



# Lab5 ADC Channel Periodically

## Step 7

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



# Lab5 ADC Channel Periodically

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"

void TIM2_ISR(void);

uint32_t i = 0;

uint16_t ADC_Value=0;

int main(void)
{
    ...
    while (1)
    {
        ...
    }
}

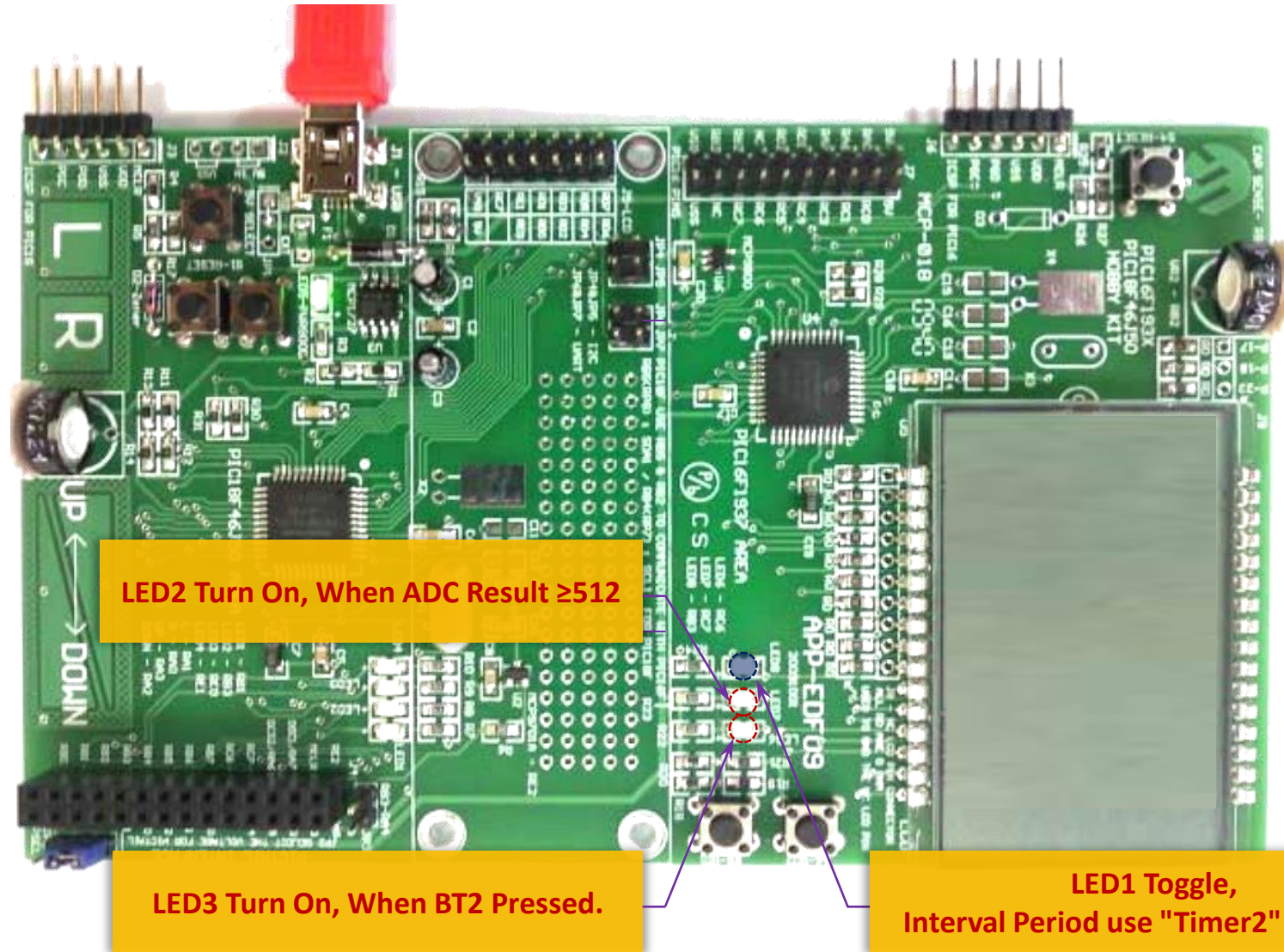
void TIM2_ISR(void)
{
    LED1_Toggle();
    ADC_Value = ADC_GetConversion(VR);
}
```

```
int main(void)
{
    ...
    TMR2_SetInterruptHandler(TIM2_ISR);
    while (1)
    {
        // Add your application code

        if(ADC_Value>=512)
        {
            LED2_SetLow();
        }
        else
        {
            LED2_SetHigh();
        }
        if(!BT2_GetValue())
        {
            LED3_SetLow();
        }
        else
        {
            LED3_SetHigh();
        }
    }
}
```

# Lab5 ADC Channel Periodically

## Result

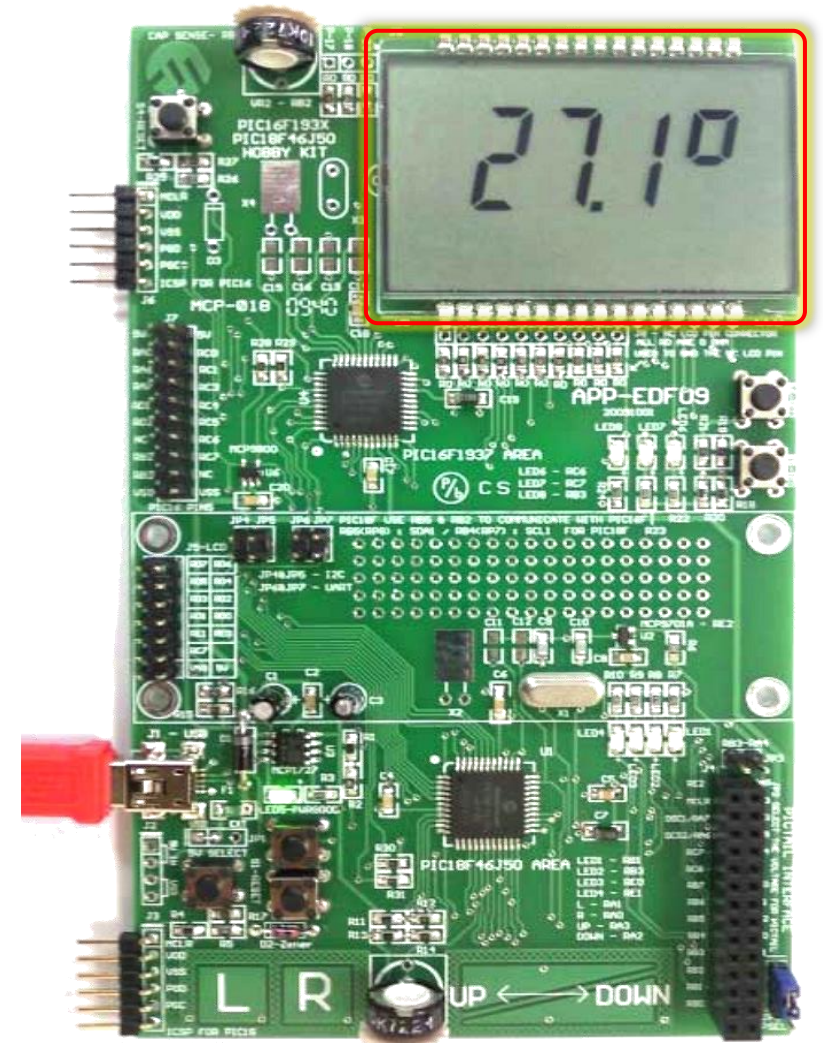


# LCD Architecture

---

# PIC16F Series LCD Features

- **Built-in LCD driver module can directly drive LCD panel.**
- **Up to four common pins.**
  - Static (1 common)
  - 1/2 multiplex (2 common)
  - 1/3 multiplex (3 common)
  - 1/4 multiplex (4 common)
- **Segment pins up to.**
  - 28 pins - up to 60 segments.
  - 44 pins - up to 96 segments.
  - 64 pins - up to 184 segments.
  - 80 pins - up to 192 segments.
- **Three LCD clock sources with selectable prescale.**
- **Built-in LCD bias generator.**
- **Support 3V and 5V glass panel.**
- **Use software adjusted contrast.**
- **Low power consumption.**



# LCD Driver Module

- LCD physical pin total= Common x Segments

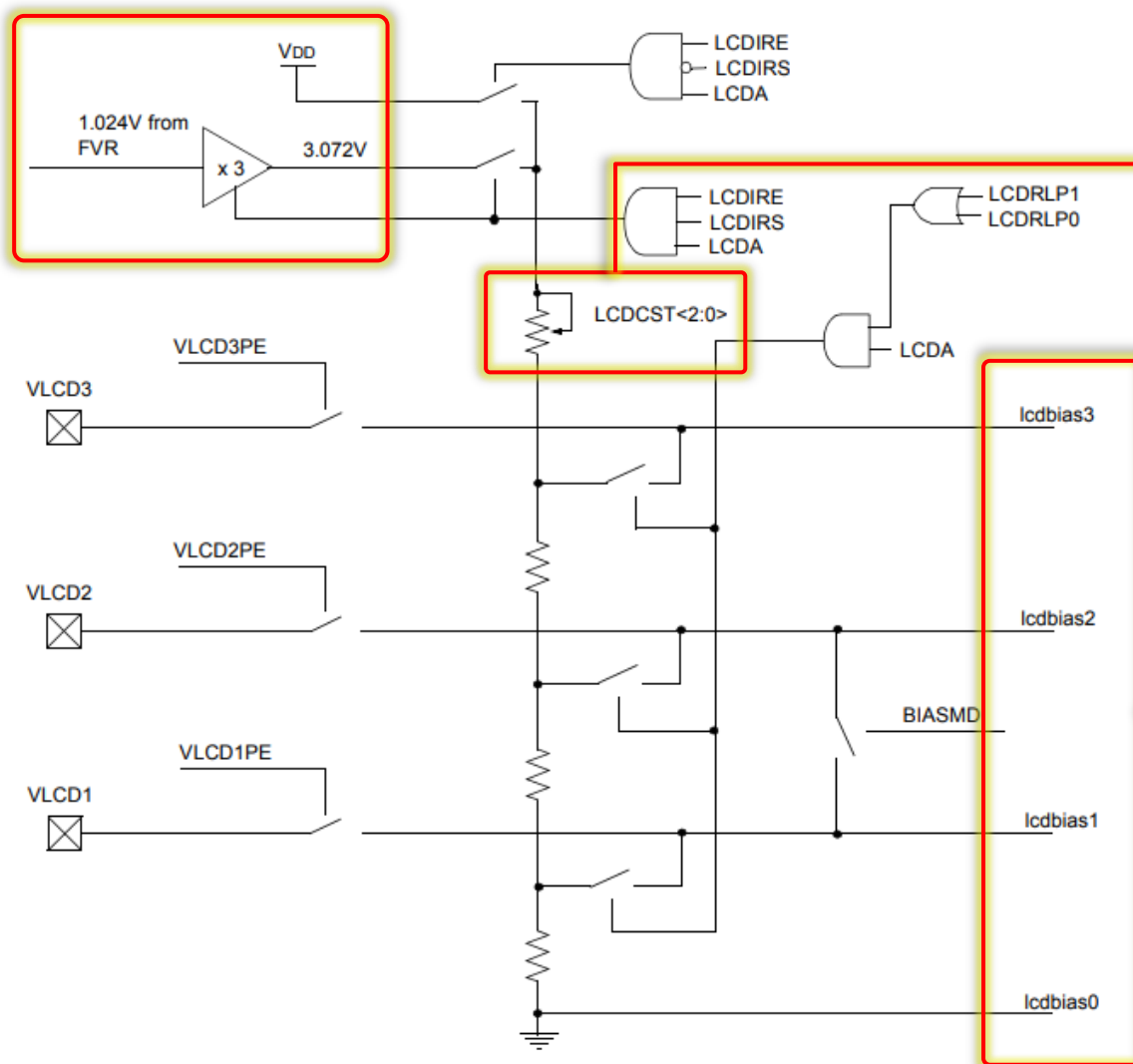
| PIC16(L)F 1933/1936/1938 | PIC16(L)F 1934/1937/1939 | PIC16(L)F 1946/1947 |
|--------------------------|--------------------------|---------------------|
| $4 \times 15 = 60$       | $4 \times 24 = 96$       | $4 \times 46 = 184$ |

- Drives a number of LCD types:

- Static (1 common)
  - Static Bias
- 1/2 multiplex (2 common)
  - 1/2 or 1/3 Bias
- 1/3 multiplex (3 common)
  - 1/2 or 1/3 Bias
- 1/4 multiplex (4 common)
  - 1/4 Bias

| Multiplex      | Maximum Number of Pixels |                 | Bias       |
|----------------|--------------------------|-----------------|------------|
|                | PIC16(L)F1936            | PIC16(L)F1934/7 |            |
| Static (COM0)  | 16                       | 24              | Static     |
| 1/2 (COM<1:0>) | 32                       | 48              | 1/2 or 1/3 |
| 1/3 (COM<2:0>) | 48                       | 72              | 1/2 or 1/3 |
| 1/4 (COM<3:0>) | 60 <sup>(1)</sup>        | 96              | 1/3        |

# LCD Bias Voltage Generation



| Power Mode | Nominal Resistance of Entire Ladder | Nominal I <sub>DD</sub> |
|------------|-------------------------------------|-------------------------|
| Low        | 3 Mohm                              | 1 $\mu$ A               |
| Medium     | 300 kohm                            | 10 $\mu$ A              |
| High       | 30 kohm                             | 100 $\mu$ A             |

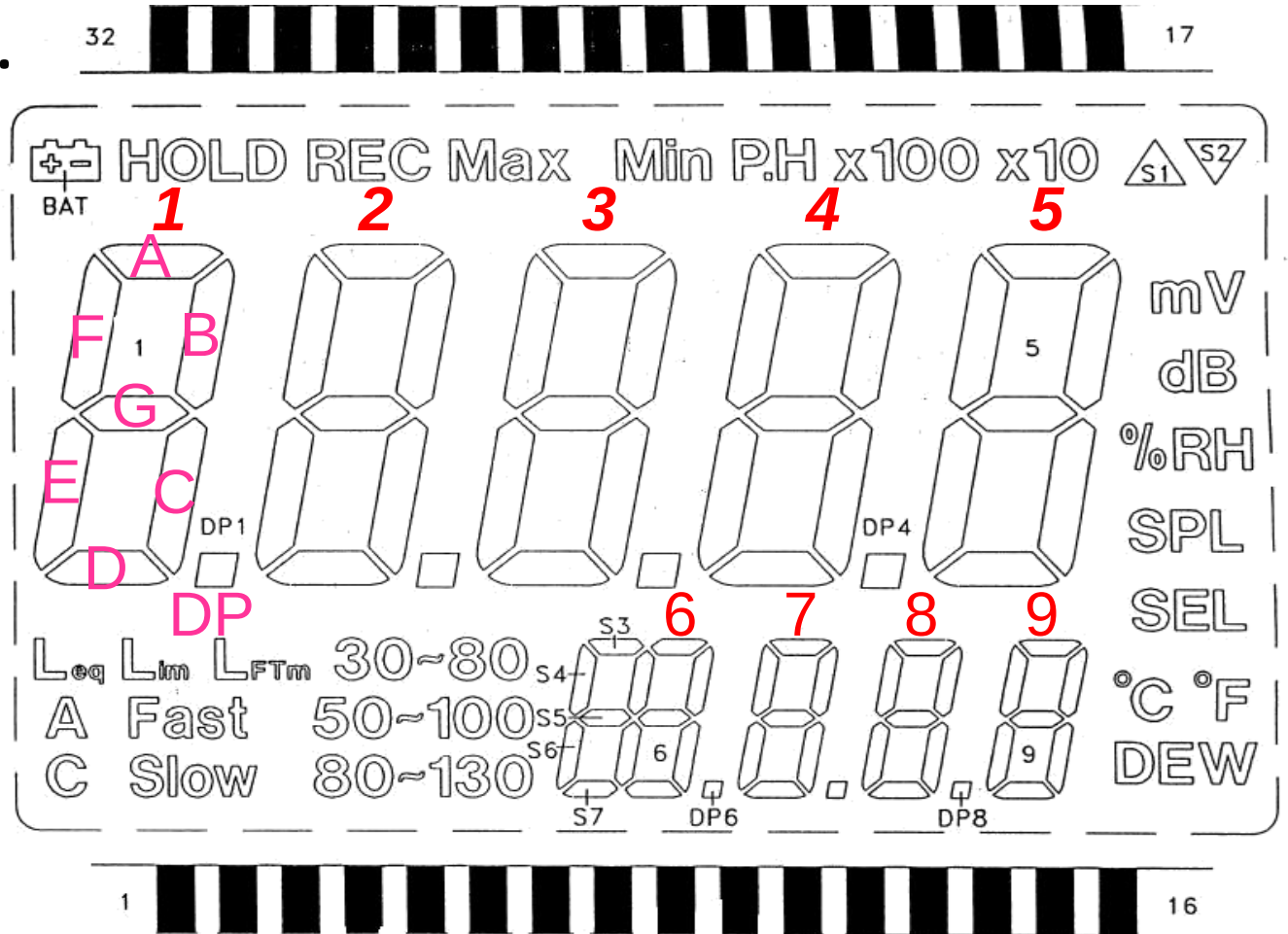
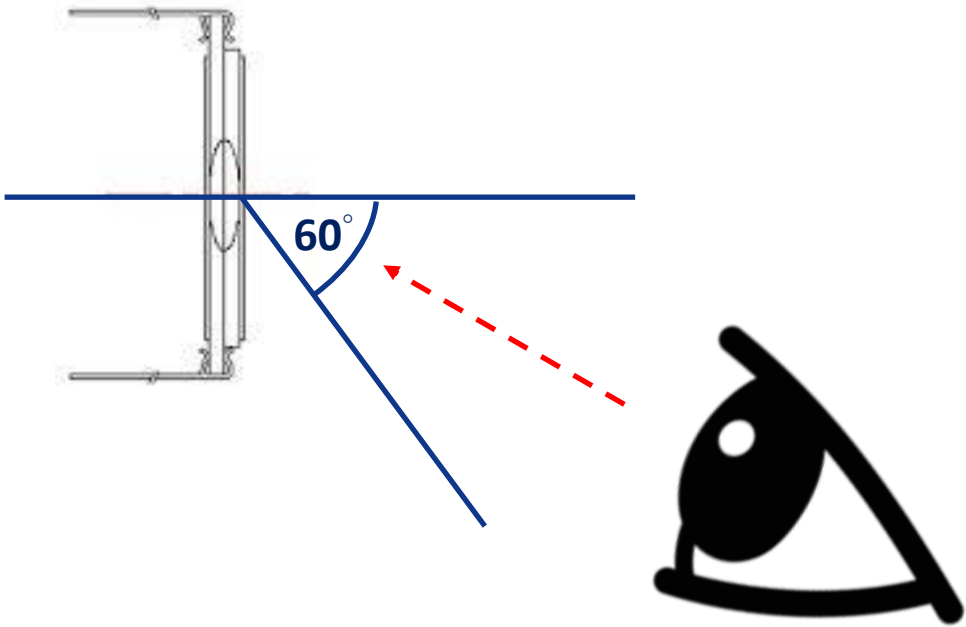
**LCD Internal Ladder  
power Modes(1/3 Bias)**

|            | Static Bias | 1/2 Bias            | 1/3 Bias            |
|------------|-------------|---------------------|---------------------|
| LCD Bias 0 | Vss         | Vss                 | Vss                 |
| LCD Bias 1 | —           | 1/2 V <sub>DD</sub> | 1/3 V <sub>DD</sub> |
| LCD Bias 2 | —           | 1/2 V <sub>DD</sub> | 2/3 V <sub>DD</sub> |
| LCD Bias 3 | VLCD3       | VLCD3               | VLCD3               |

**LCD Bias Voltages**

# LCD panel Specifications

- Operating voltage : **3.3V.**
  - Operating temperature : **-10°C ~ 70°C.**
  - Drive Scheme : **1/4 Duty**、**1/3 Bias.**
  - Viewing Direction : **6:00 O'Clock.**
  - TN(Twisted Nematic) Type.
- 



# LCD Pin Configuration

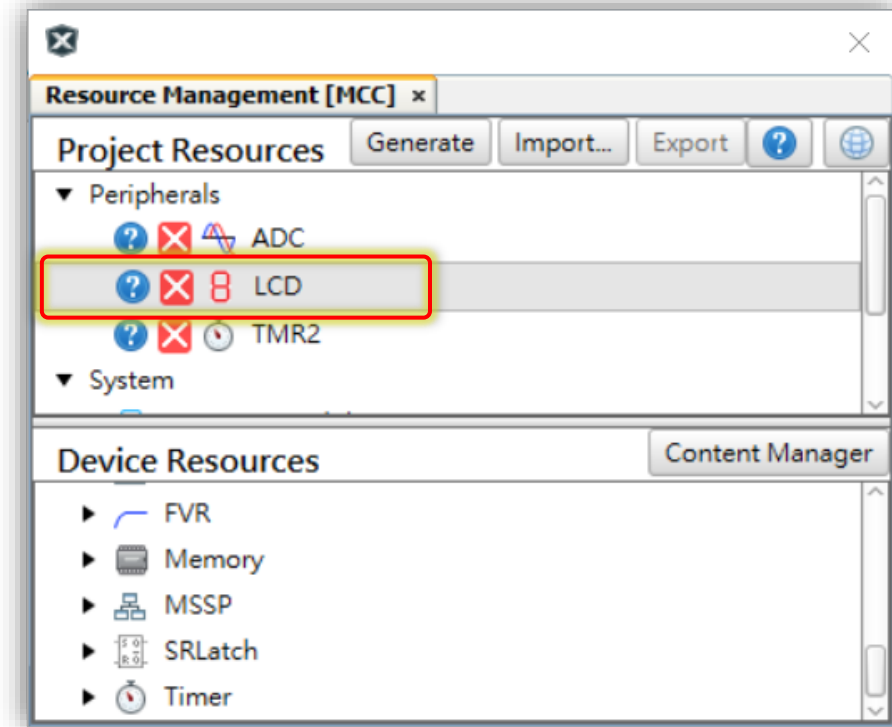
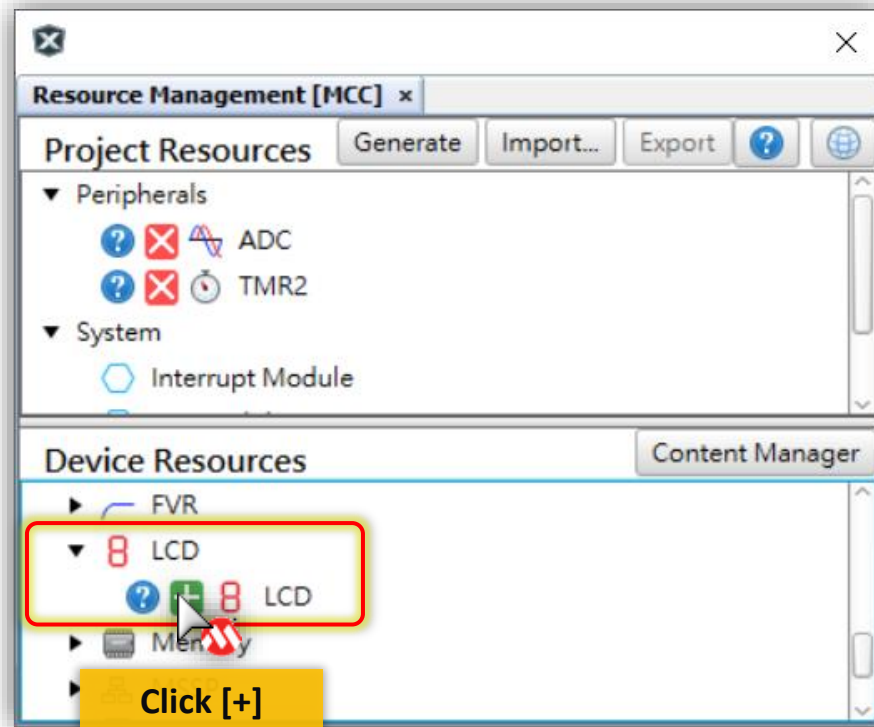
| LCD PIN | COM0 | COM1            | COM2            | COM3             | Segment |
|---------|------|-----------------|-----------------|------------------|---------|
| 1       |      | L <sub>eq</sub> | L <sub>im</sub> | L <sub>FTm</sub> | N.C.    |
| 2       | S7   | S6              | S5              | S4               | N.C.    |
| 3       | 6D   | 6E              | 6G              | 6F               | N.C.    |
| 4       | 6DP  | 6C              | 6B              | 6A               | N.C.    |
| 5       | 7D   | 7E              | 7G              | 7F               | N.C.    |
| 6       | 7DP  | 7C              | 7B              | 7A               | N.C.    |
| 7       | 8D   | 8E              | 8G              | 8F               | N.C.    |
| 8       | 8DP  | 8C              | 8B              | 8A               | N.C.    |
| 9       | 9D   | 9E              | 9G              | 9F               | N.C.    |
| 10      |      | 9C              | 9B              | 9A               | N.C.    |
| 11      |      | °F              | °C              | S3               | SEG21   |
| 12      | C    | A               | Slow            | Fast             | SEG5    |
| 13      | COM0 |                 |                 |                  | COM0    |
| 14      |      | COM1            |                 |                  | COM1    |
| 15      |      |                 | COM2            |                  | COM2    |
| 16      |      |                 |                 | COM3             | COM3    |

# LCD Pin Configuration

| LCD PIN | COM0 | COM1   | COM2   | COM3  | Segment |
|---------|------|--------|--------|-------|---------|
| 17      | DEW  | 80~130 | 50~100 | 30~80 | N.C.    |
| 18      | SEL  | SPL    | dB     | %RH   | N.C.    |
| 19      | m    | V      | S2     | S1    | SEG7    |
| 20      |      | 5C     | 5B     | 5A    | SEG15   |
| 21      | 5D   | 5E     | 5G     | 5F    | SEG4    |
| 22      |      |        |        |       | N.C.    |
| 23      | P.H  | x10    | x100   | Min   | N.C.    |
| 24      | 4DP  | 4C     | 4B     | 4A    | SEG22   |
| 25      | 4D   | 4E     | 4G     | 4F    | SEG23   |
| 26      | 3DP  | 3C     | 3B     | 3A    | SEG3    |
| 27      | 3D   | 3E     | 3G     | 3F    | SEG16   |
| 28      | Max  | REC    | HOLD   | BAT   | SEG17   |
| 29      | 2DP  | 2C     | 2B     | 2A    | SEG18   |
| 30      | 2D   | 2E     | 2G     | 2F    | SEG19   |
| 31      | 1DP  | 1C     | 1B     | 1A    | SEG20   |
| 32      | 1D   | 1E     | 1G     | 1F    | SEG0    |

# MCC's LCD Module

- Add LCD resource from Device Resources.



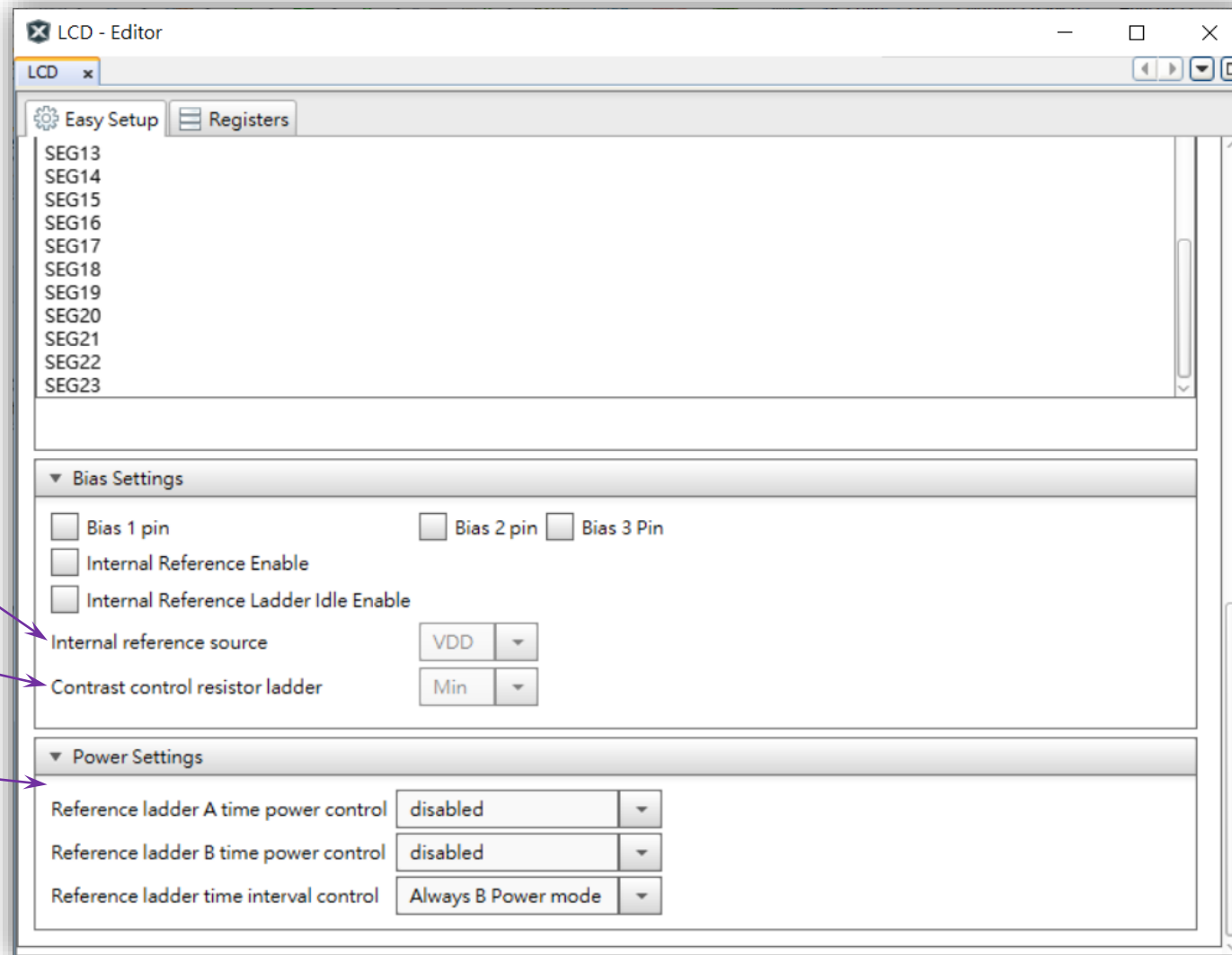
# MCC's LCD Module

The screenshot shows the 'LCD - Editor' window with the 'Easy Setup' tab selected. The interface is divided into 'Hardware Settings' and 'LCD Design' sections. Callouts point to the following elements:

- Enable LCD**: Points to the 'Enable LCD' checkbox in the Hardware Settings section.
- Waveform Type**: Points to the 'Waveform type' dropdown menu, which is set to 'Type-A waveform'.
- Clock Source**: Points to the 'Clock Source' dropdown menu, which is set to 'FOSC/256'.
- Prescaler Settings**: Points to the 'Prescaler Settings' dropdown menu, which is set to '1:1'.
- Multiplex Mode**: Points to the 'Multiplex Mode' dropdown menu, which is set to 'Static COM0'.
- LCD Design**: Points to the large yellow area in the LCD Design section.
- Enable Interrupt**: Points to the 'Enable LCD Interrupt' checkbox.
- Sleep Mode**: Points to the 'Disable Sleep Mode' checkbox.
- Base Mode**: Points to the 'Frame Frequency' label, which is set to '1.95313 kHz'.
- SEG Select**: Points to the 'SEG' dropdown menu at the bottom right.
- COM Select**: Points to the 'COM' dropdown menu at the bottom right.

Other visible settings include 'Bias Mode' set to 'Static' and 'Available Pixels' set to '24'. The bottom status bar shows 'Macro: PREFIX \_SYM'.

# MCC's LCD Module



Voltage reference

Contrast control

Power Setting

# LCD Functions of MCC

- MCC Provide below functions for LCD:

Prototype definition : "lcd.h"

- **Void** LCD\_Initialize (**void**);
- **void** LCD\_Enable (**void**);
- **void** LCD\_EnableSleepMode (**void**)
- **void** LCD\_Disable (**void**);
- **void** LCD\_DisableSleepMode (**void**);
- **bool** LCD\_IsActive (**void**);
- **void** LCD\_SetContrast (**unsigned int** **value**);
- **void** LCD\_SetIntervalAPowerMode (**unsigned int** **value**);
- **void** LCD\_SetIntervalBPowerMode (**unsigned int** **value**);
- **void** LCD\_SetPowerDistribution (**unsigned int** **value**);

# Lab7 LCD Display

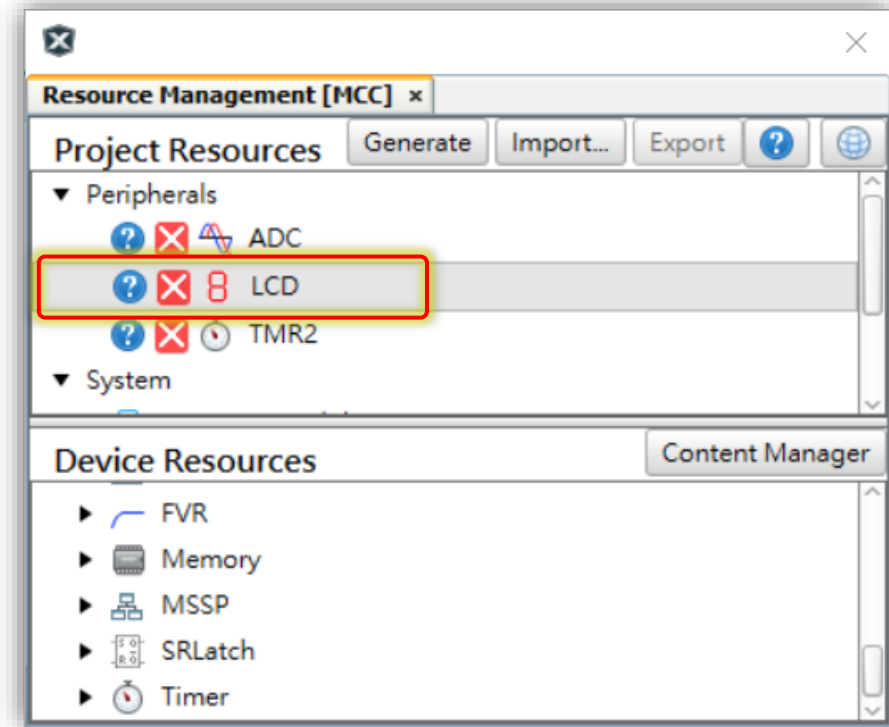
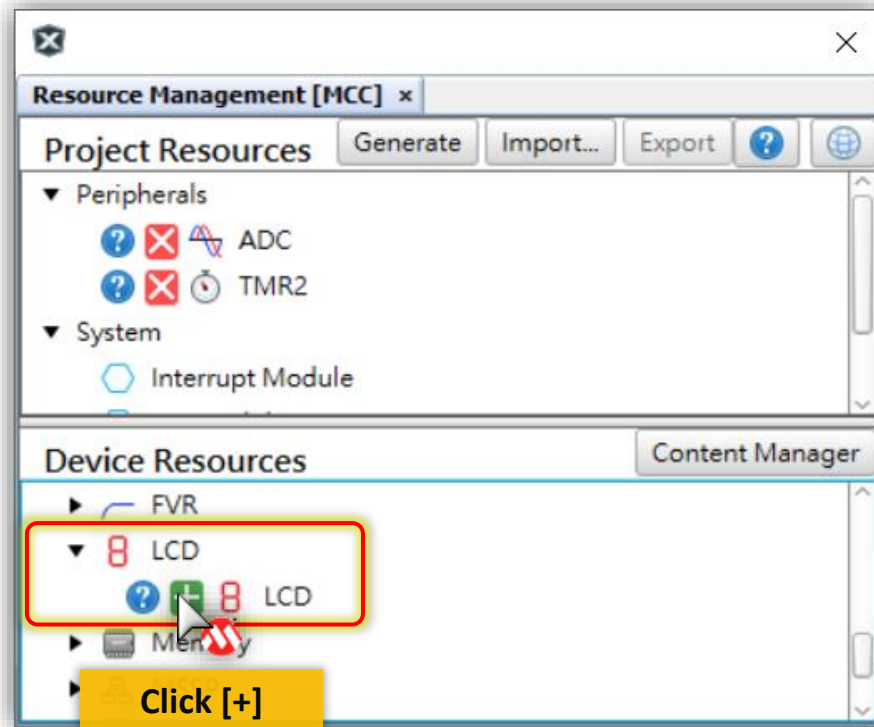
- Base on current project or Open `.\Lab7_xxx.X` to do exercises
- Try to add LCD module and conversion Potentiometer (VR) voltage to shows ADC result on LCD Display.

**Let's go!**

# Lab7 LCD Display

## Step 1

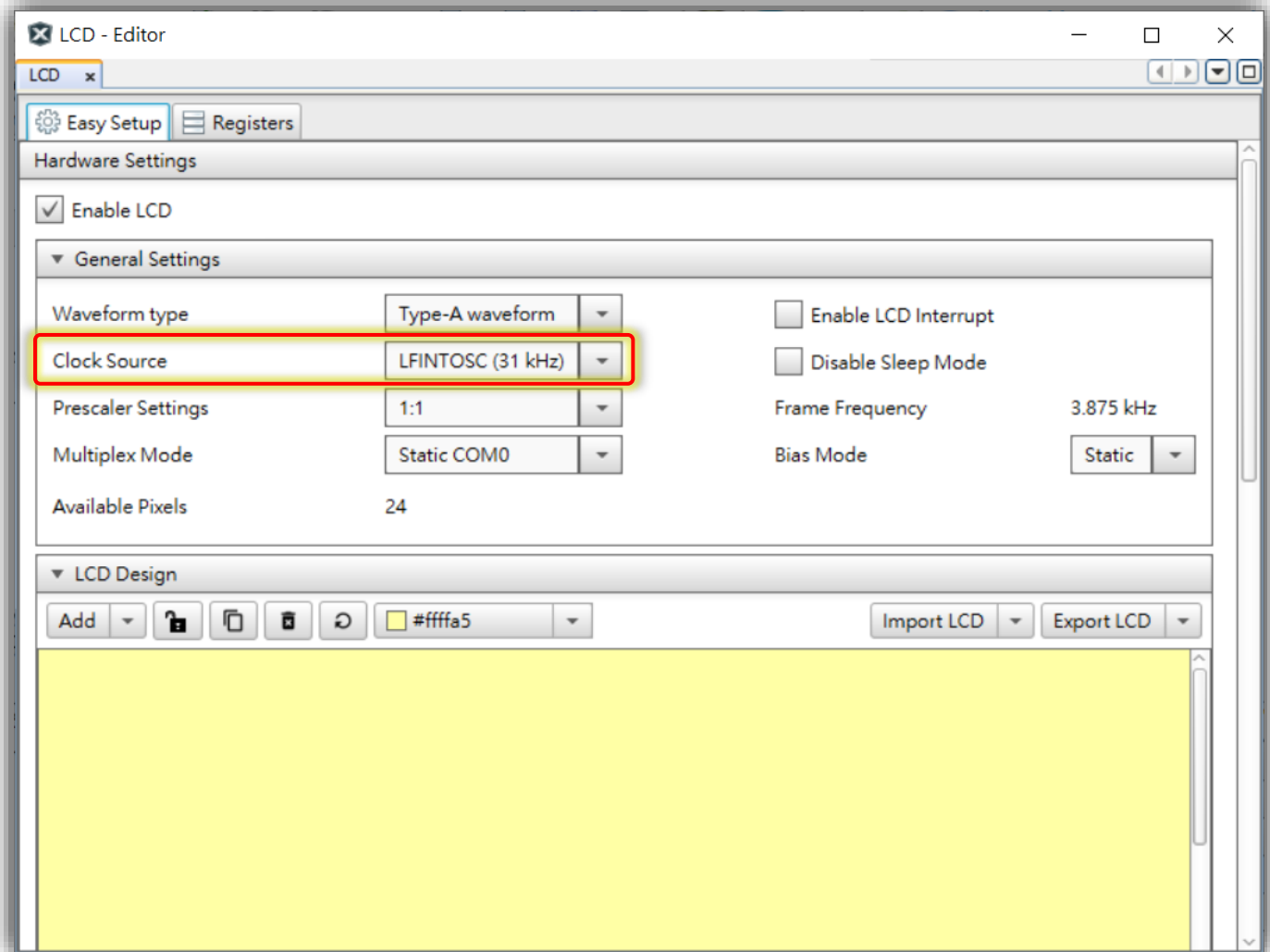
- Device Resources
  - LCD ► LCD



# Lab7 LCD Display

## Step 2

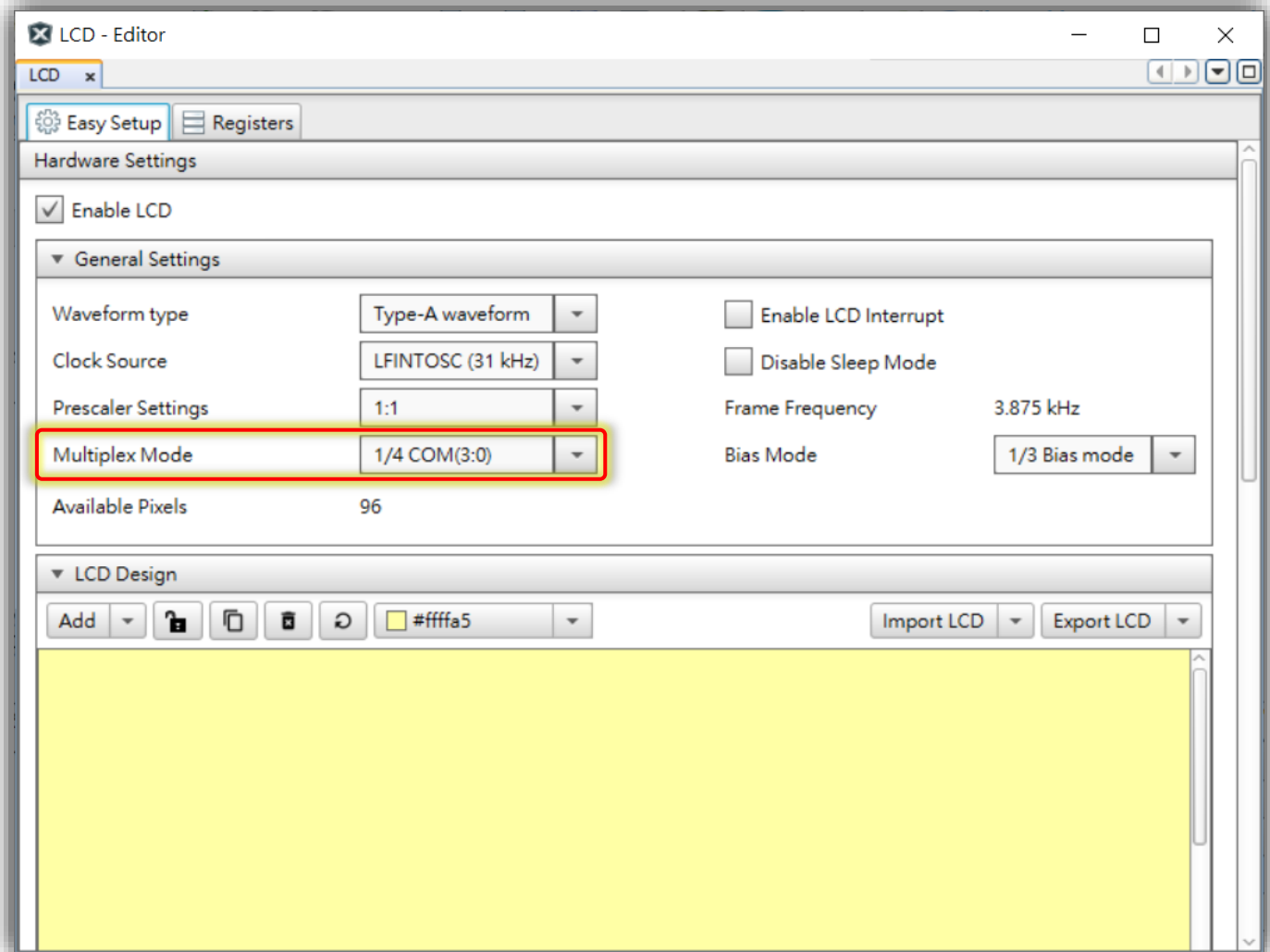
- Clock Source
  - $F_{osc}/256 \rightarrow \text{LFINTOSC}(31 \text{ kHz})$



# Lab7 LCD Display

## Step 3

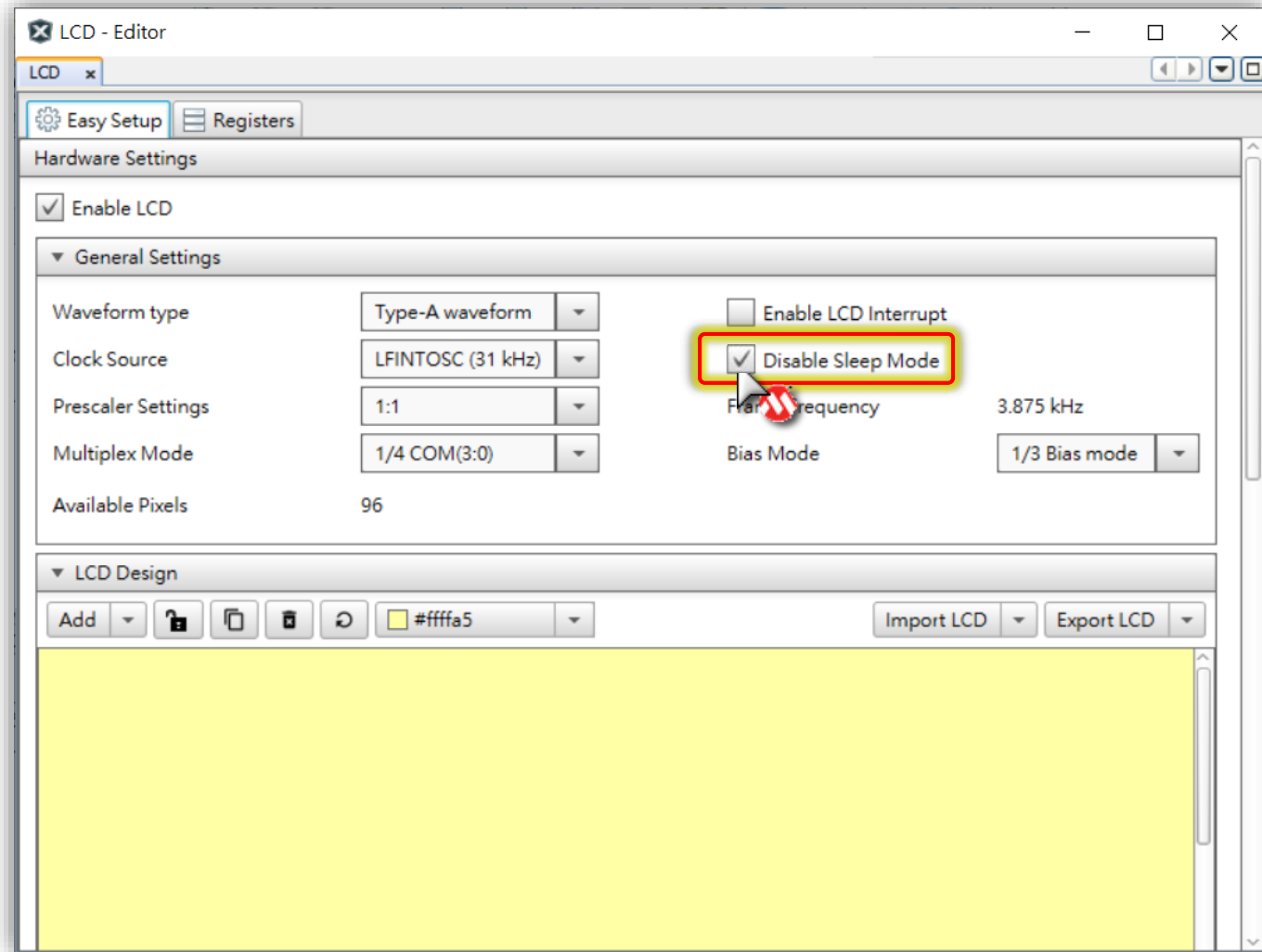
- **Multiplex Mode**
  - **Static COM0 ▶ 1/4 COM(3:0)**



# Lab7 LCD Display

## Step 4

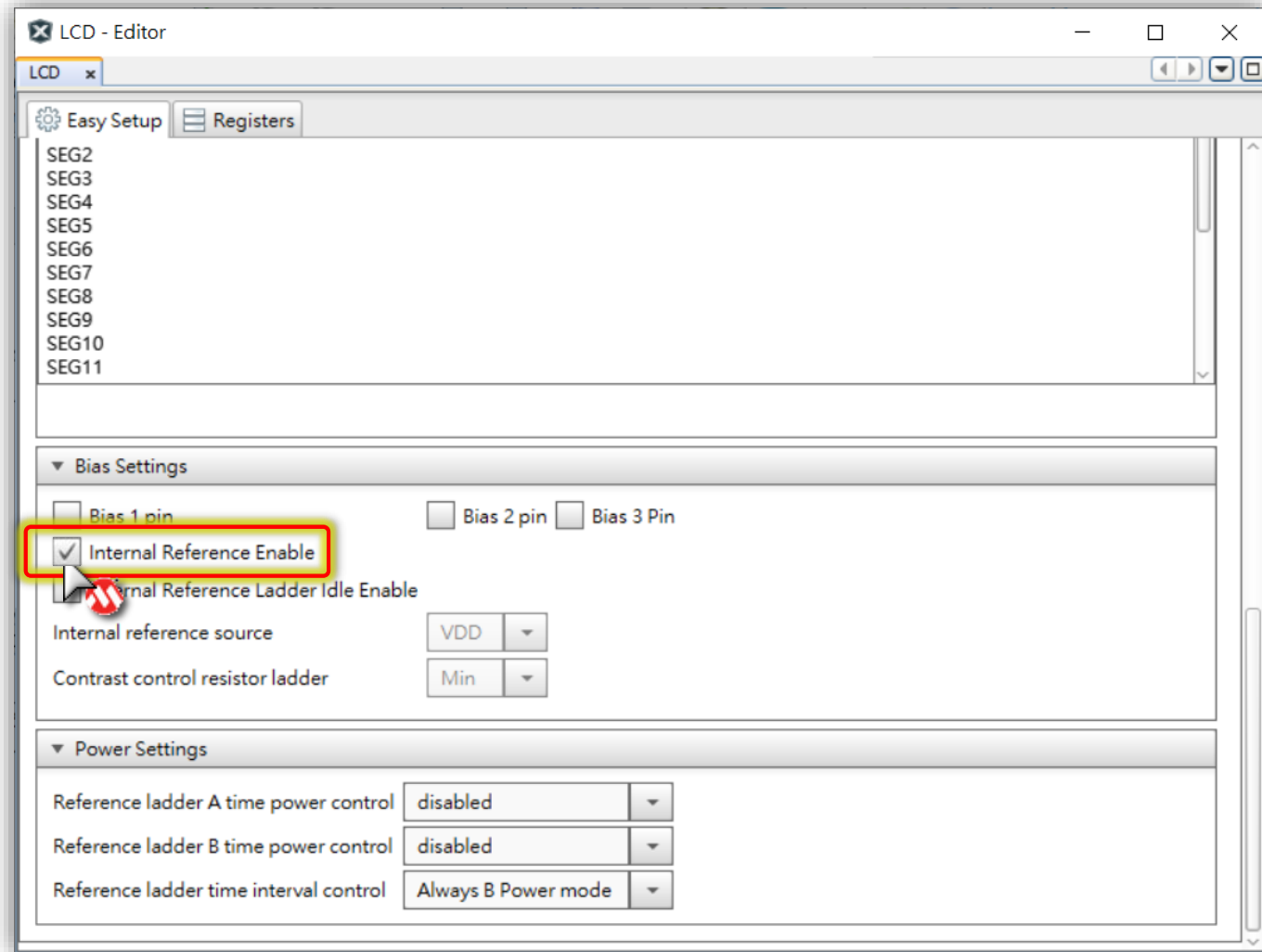
- **Disable Sleep Mode.**



# Lab7 LCD Display

## Step 5

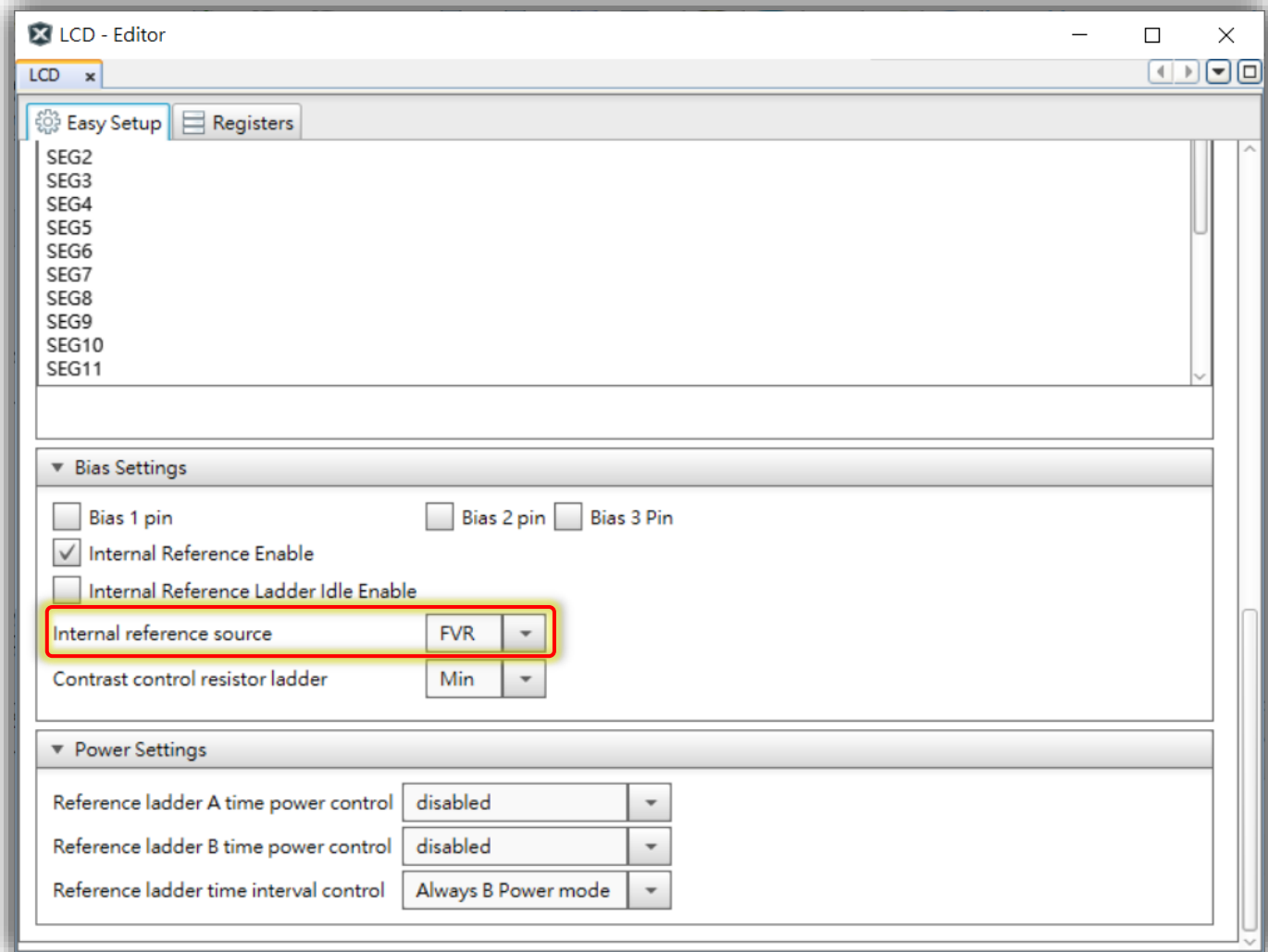
- **Enable Internal Reference.**



# Lab7 LCD Display

## Step 6

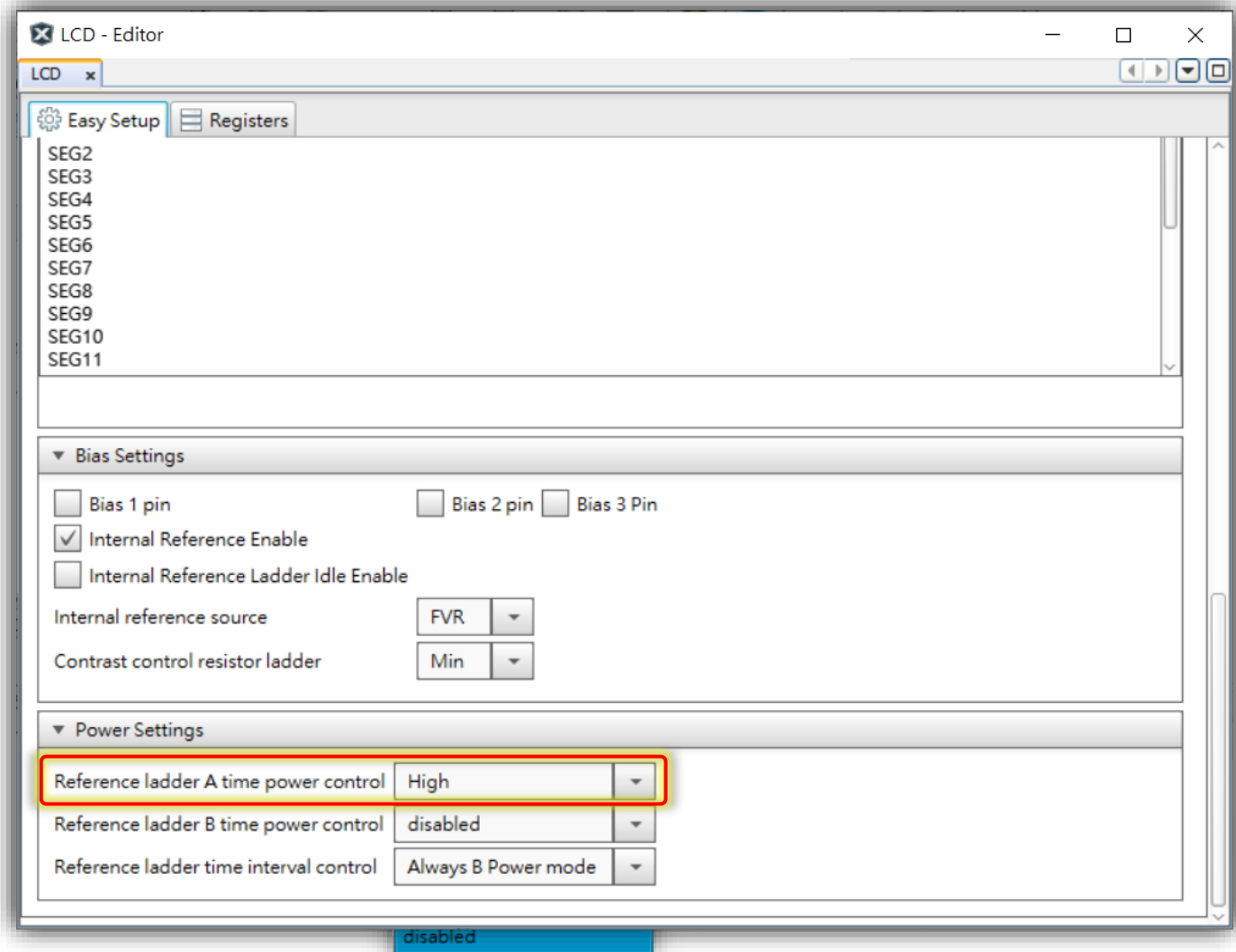
- Internal reference source
  - VDD ► FVR



# Lab7 LCD Display

## Step 7

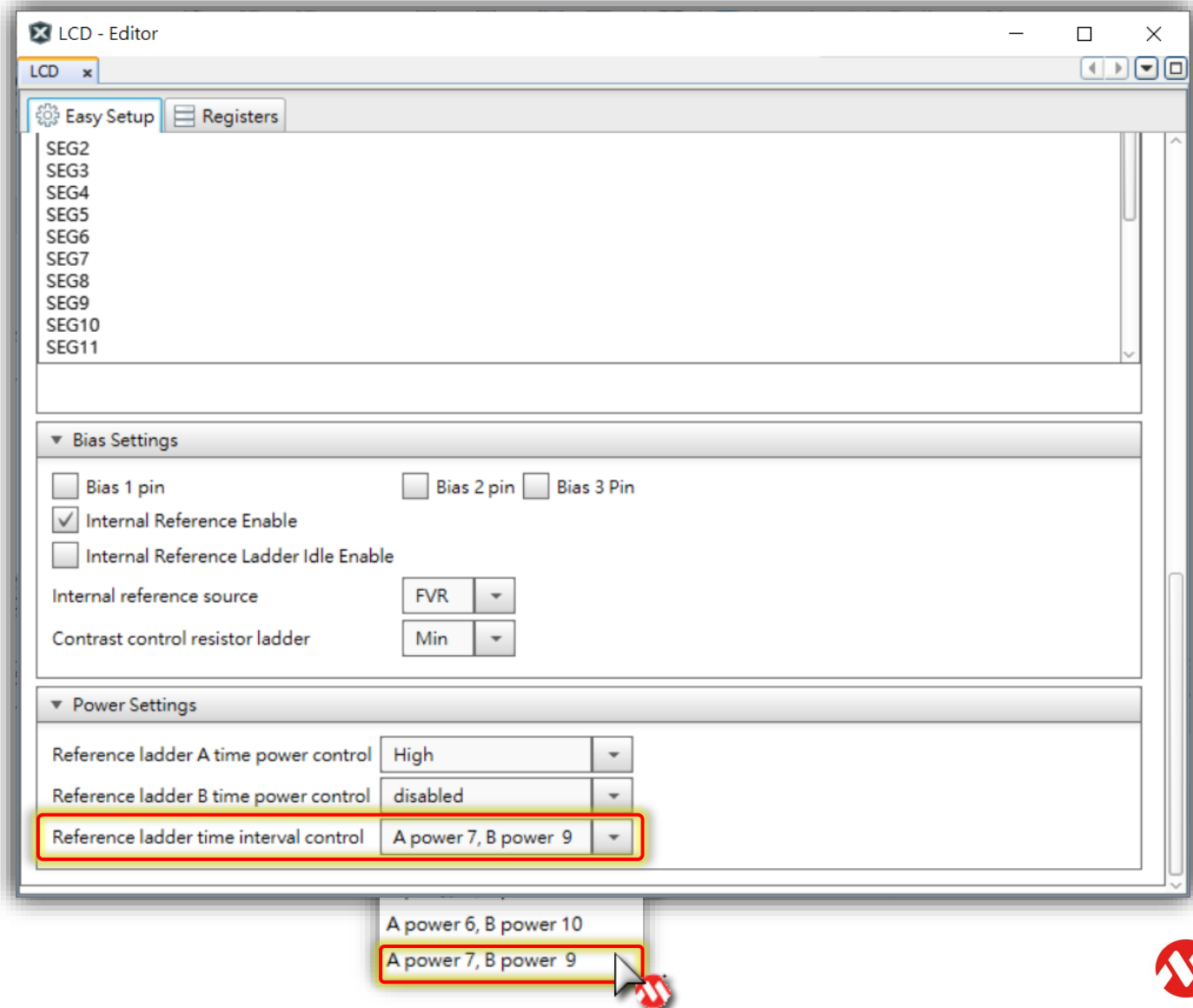
- Reference ladder A time ...
  - disabled ► High



# Lab7 LCD Display

## Step 8

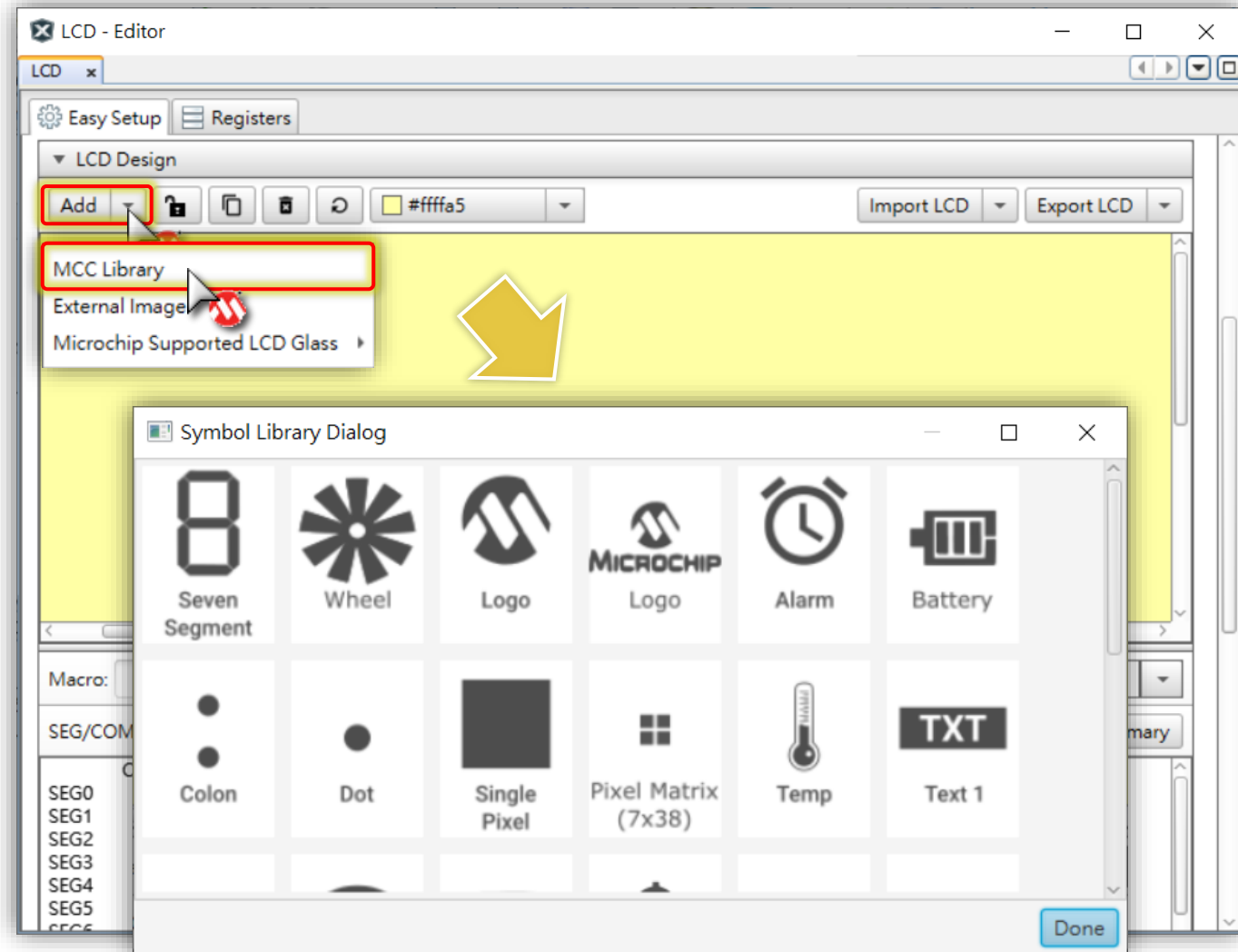
- Reference ladder time ...
  - Always B Power mode
  - ▶
  - A power 7, B power 9



# Lab7 LCD Display

## Step 9

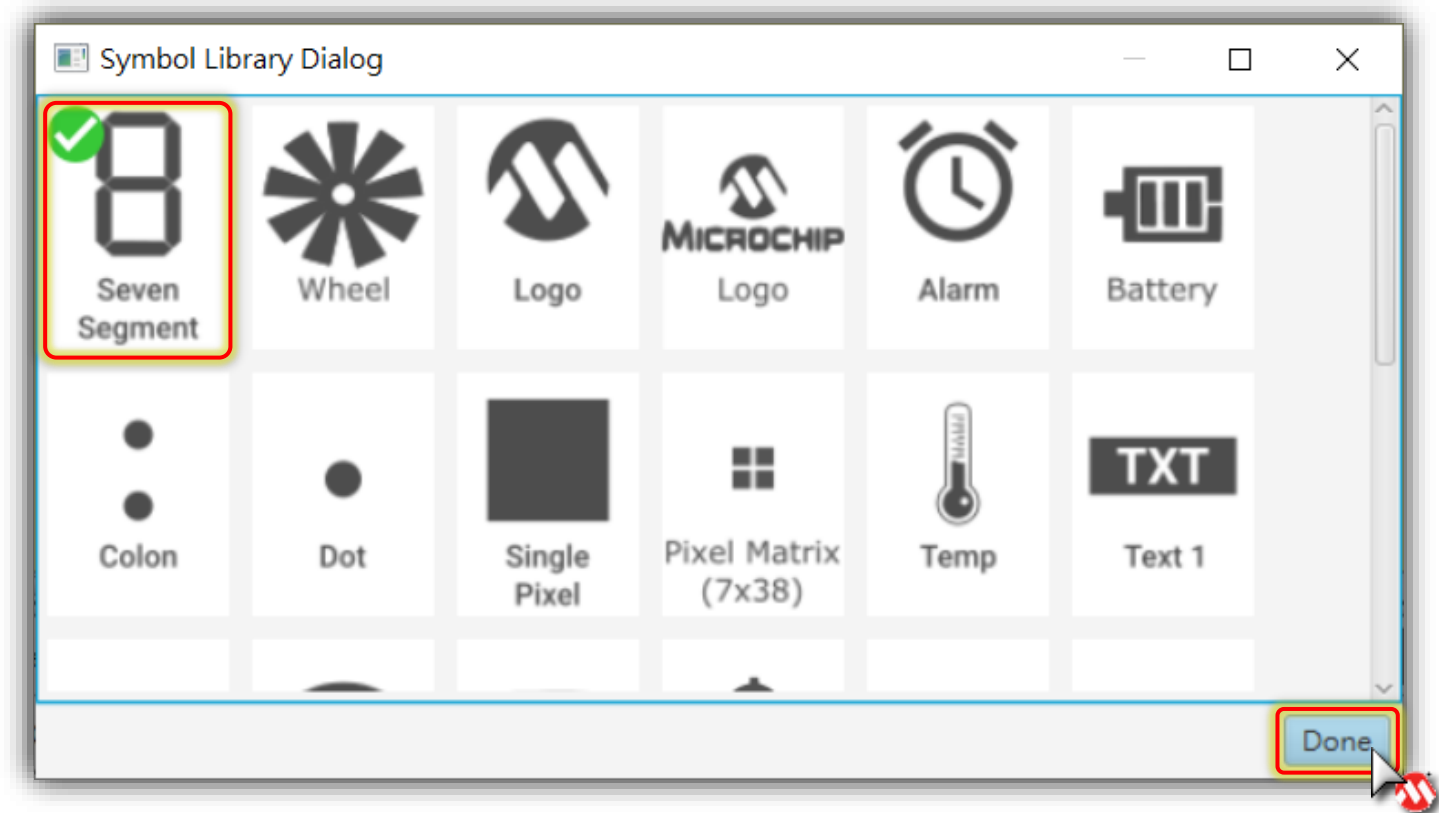
- LCD Design
  - Add ► MCC Library



# Lab7 LCD Display

## Step 10

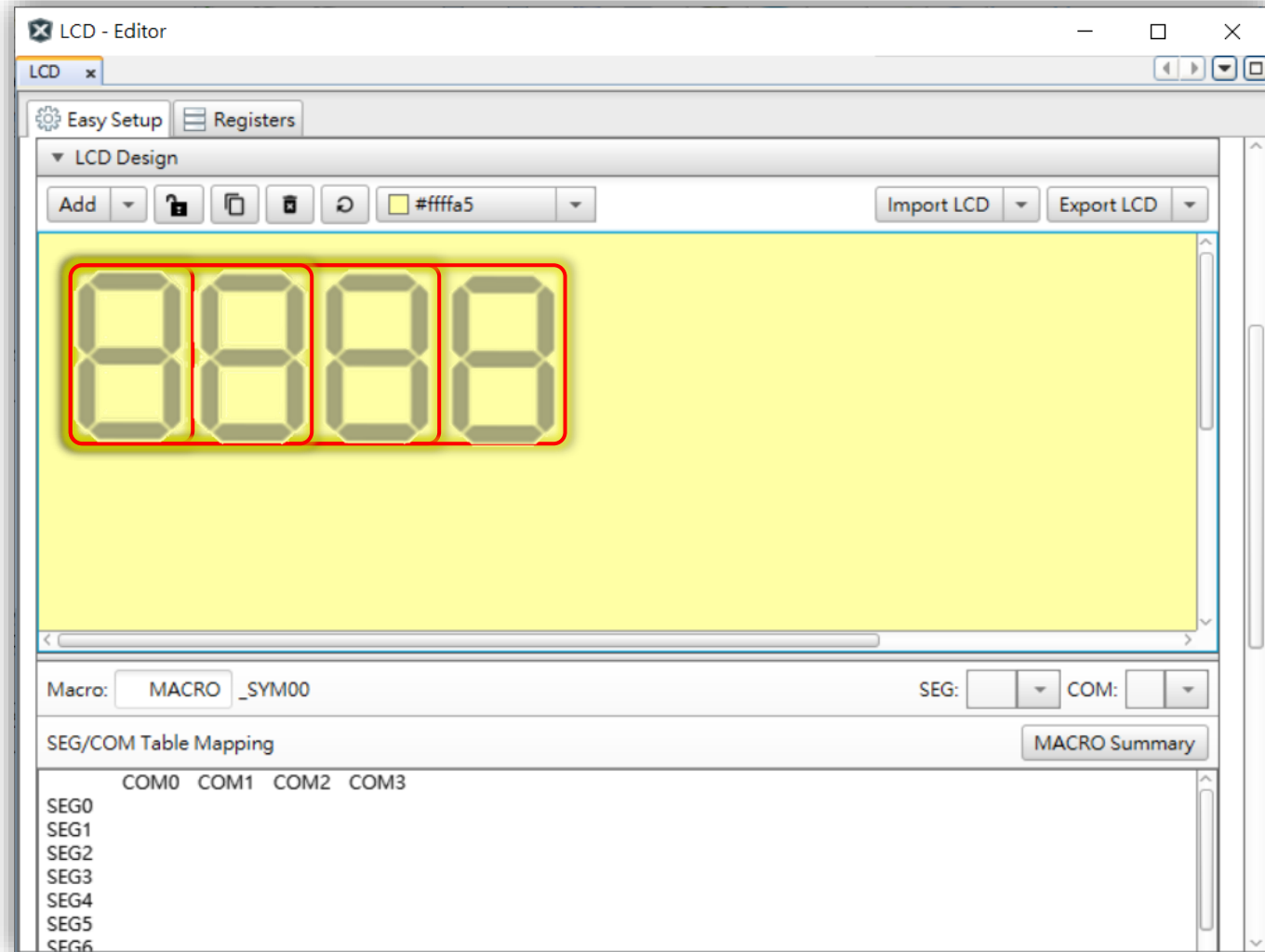
- LCD Design
  - **Select Seven Segment**



# Lab7 LCD Display

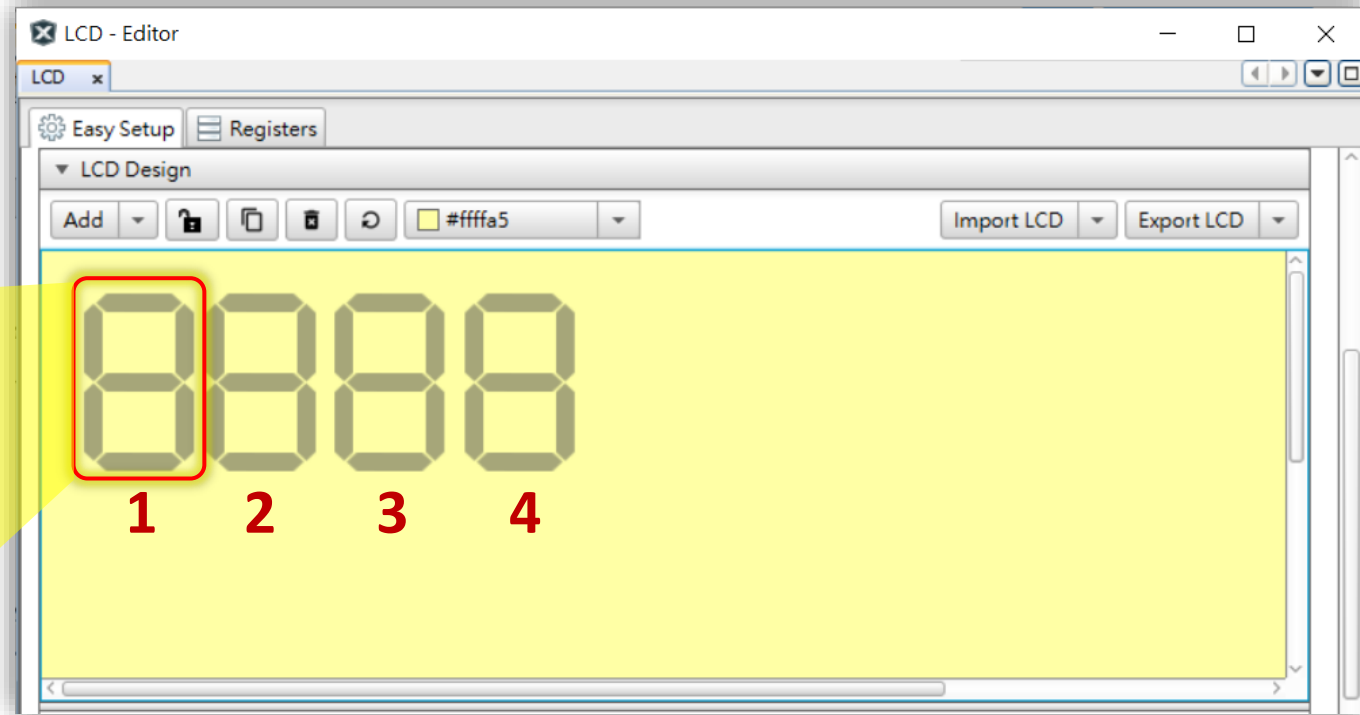
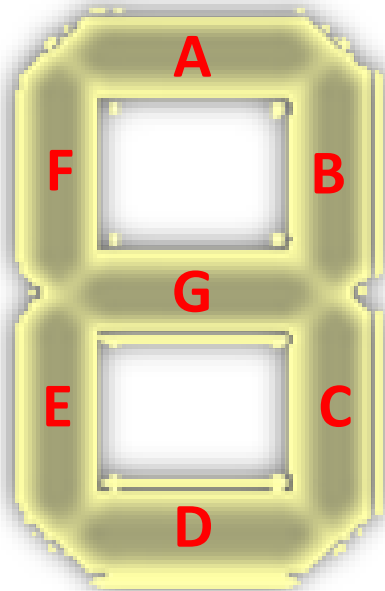
## Step 11

- **LCD Design.**



# Lab7 LCD Display

## Step 12

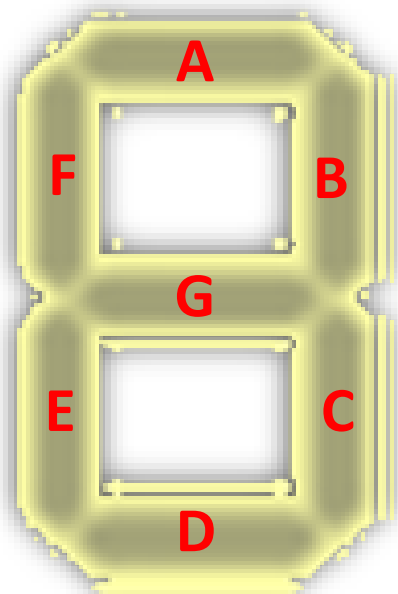


| LCD PIN | COM0 | COM1 | COM2 | COM3 | Segment |
|---------|------|------|------|------|---------|
| 31      | 1DP  | 1C   | 1B   | 1A   | SEG20   |
| 32      | 1D   | 1E   | 1G   | 1F   | SEG0    |

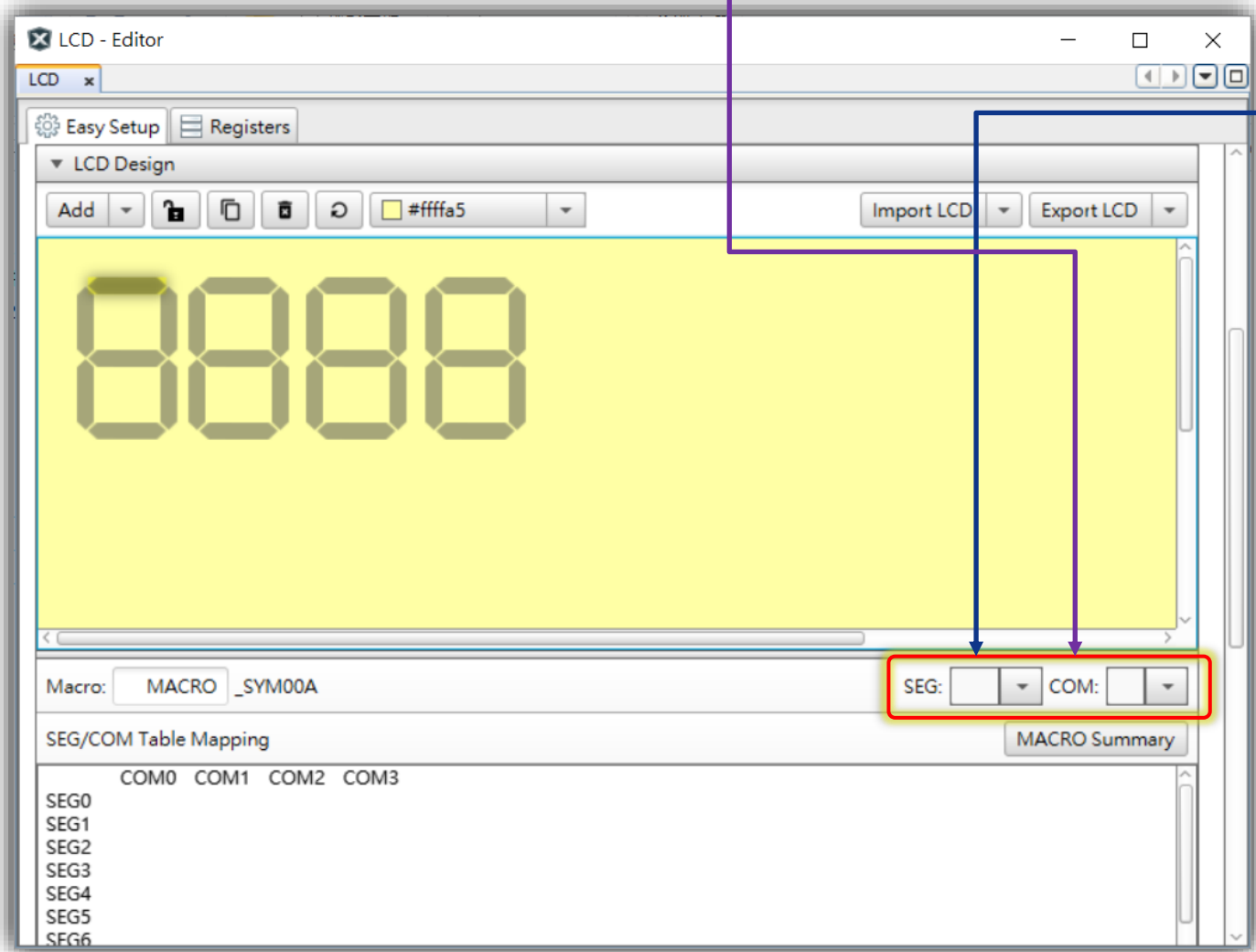
# Lab7 LCD Display

## Step 13

- Assign Segment and COM.



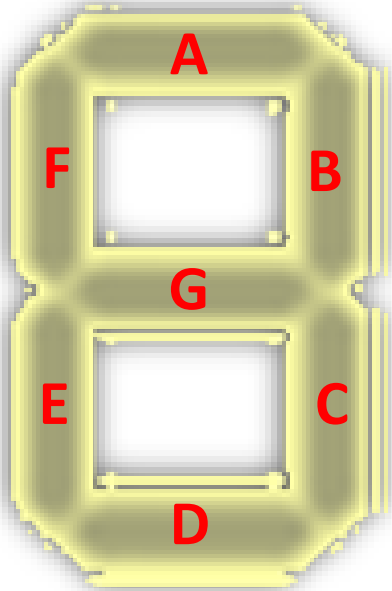
| LCD PIN | COM0 | COM1 | COM2 | COM3 | Segment |
|---------|------|------|------|------|---------|
| 31      | 1DP  | 1C   | 1B   | 1A   | SEG20   |
| 32      | 1D   | 1E   | 1G   | 1F   | SEG0    |



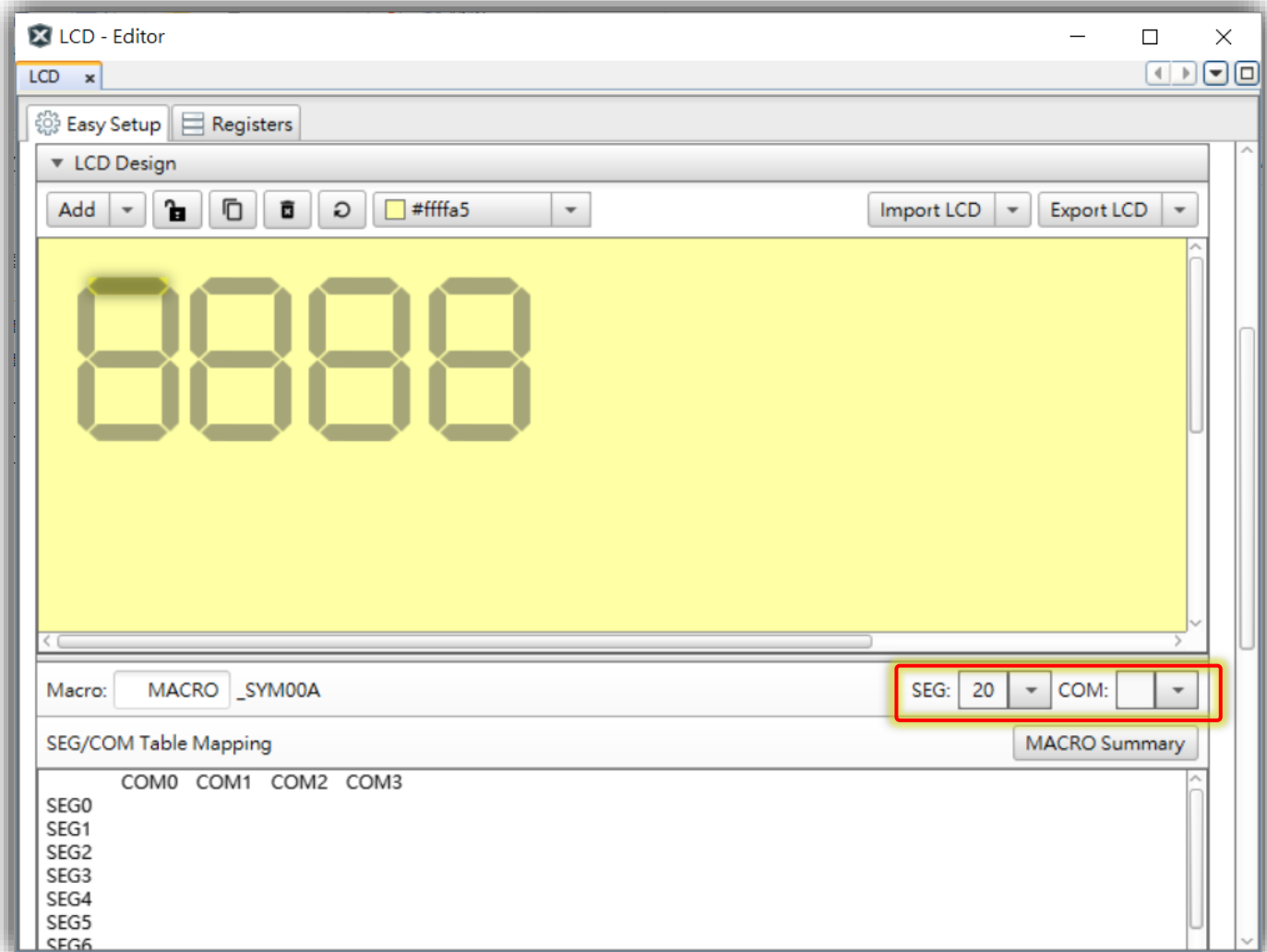
# Lab7 LCD Display

## Step 14

- Assign Segment and COM.
  - Segment A ► SEG20



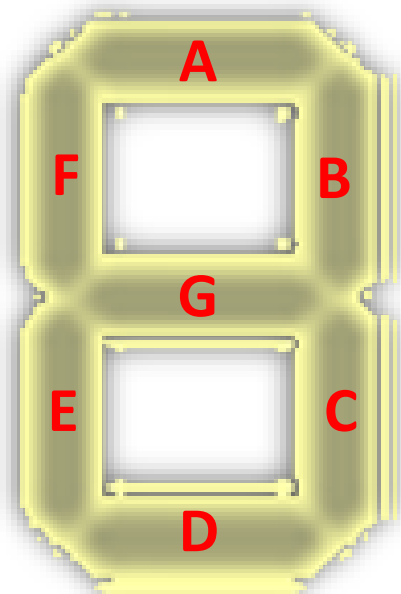
| LCD PIN | COM0 | COM1 | COM2 | COM3 | Segment |
|---------|------|------|------|------|---------|
| 31      | 1DP  | 1C   | 1B   | 1A   | SEG20   |
| 32      | 1D   | 1E   | 1G   | 1F   | SEG0    |



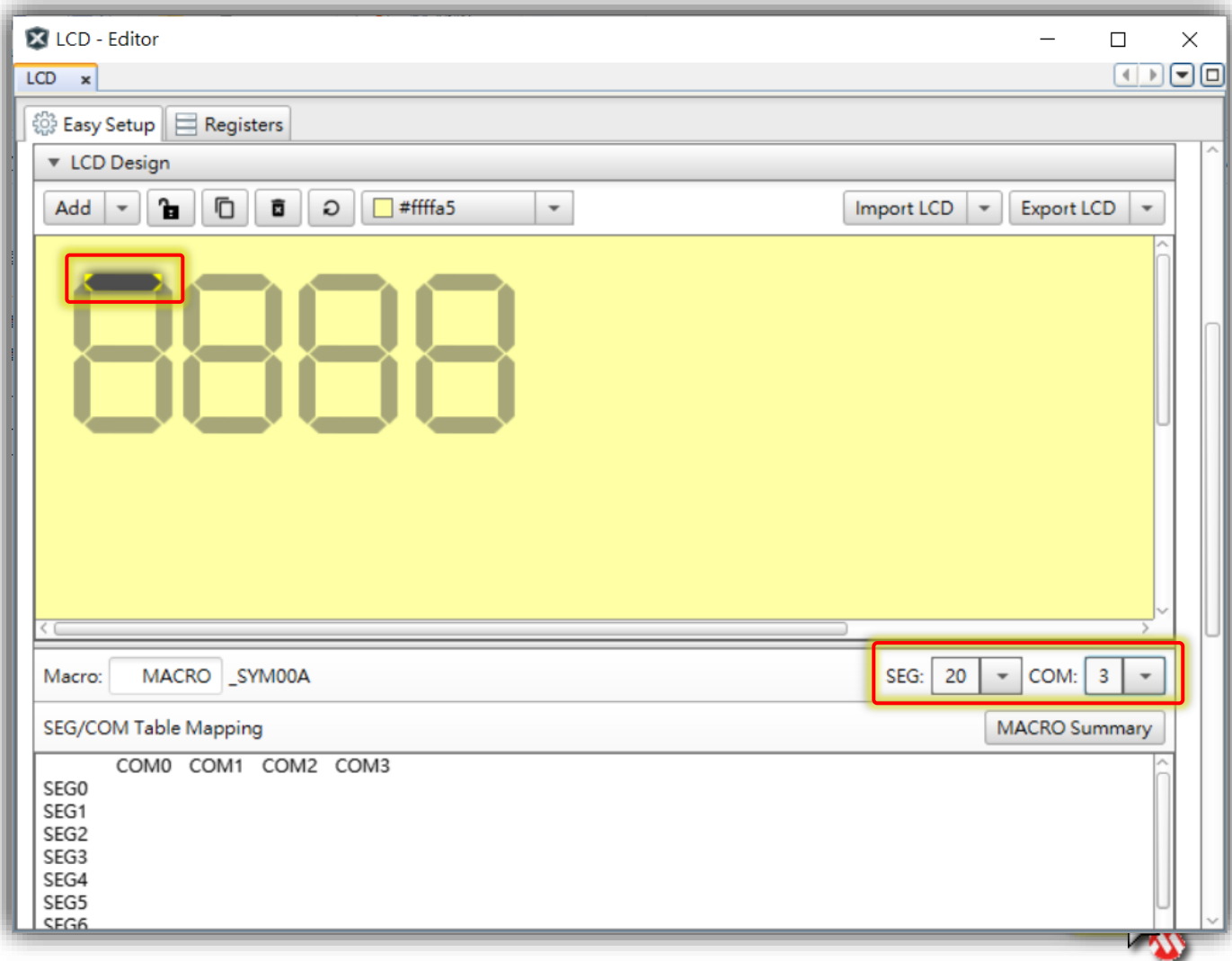
# Lab7 LCD Display

## Step 15

- Assign Segment and COM.
- Segment A ► COM3



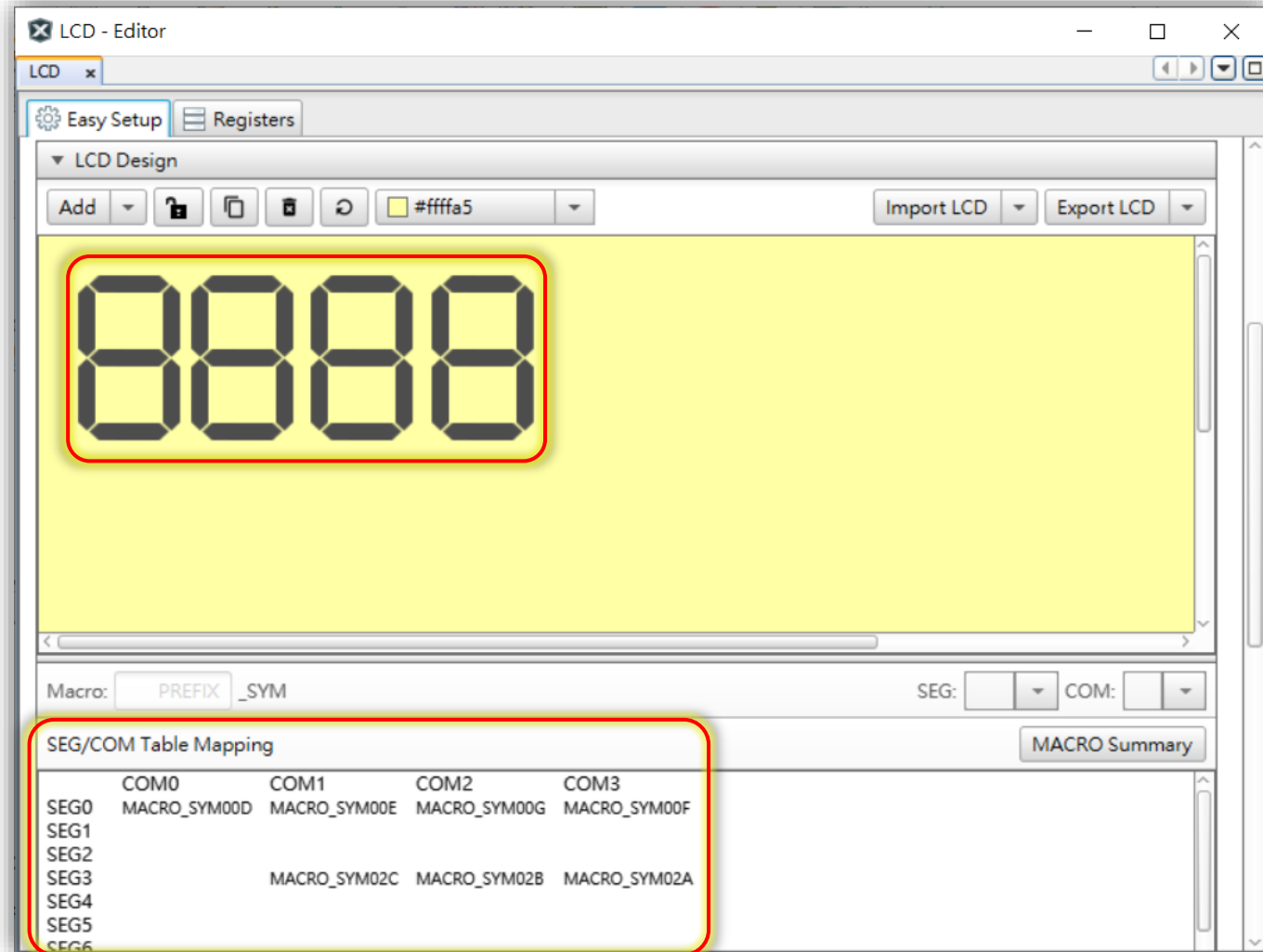
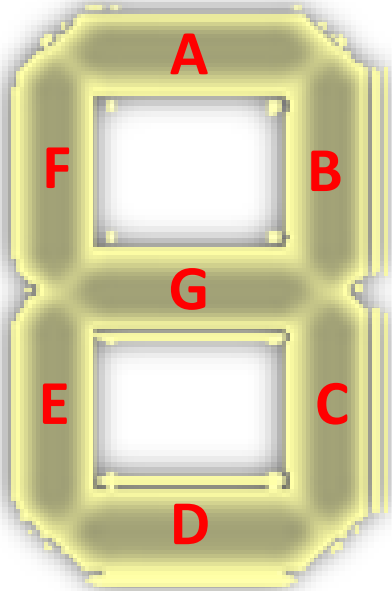
| LCD PIN | COM0 | COM1 | COM2 | COM3 | Segment |
|---------|------|------|------|------|---------|
| 31      | 1DP  | 1C   | 1B   | 1A   | SEG20   |
| 32      | 1D   | 1E   | 1G   | 1F   | SEG0    |



# Lab7 LCD Display

## Step 16

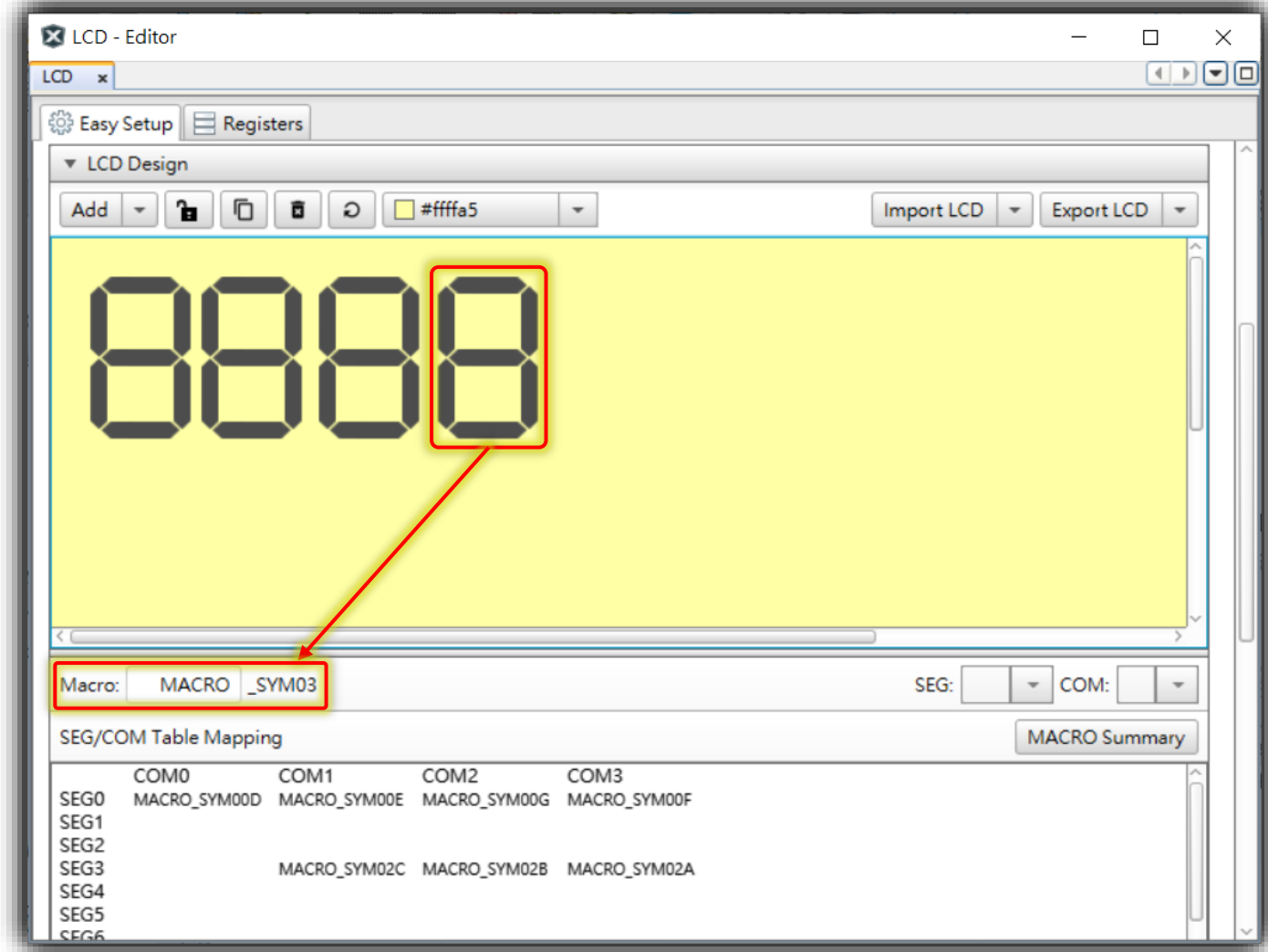
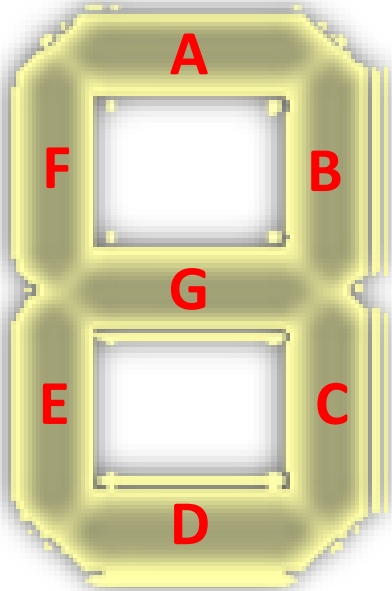
- **Confirm Segment and COM.**



# Lab7 LCD Display

## Step 17

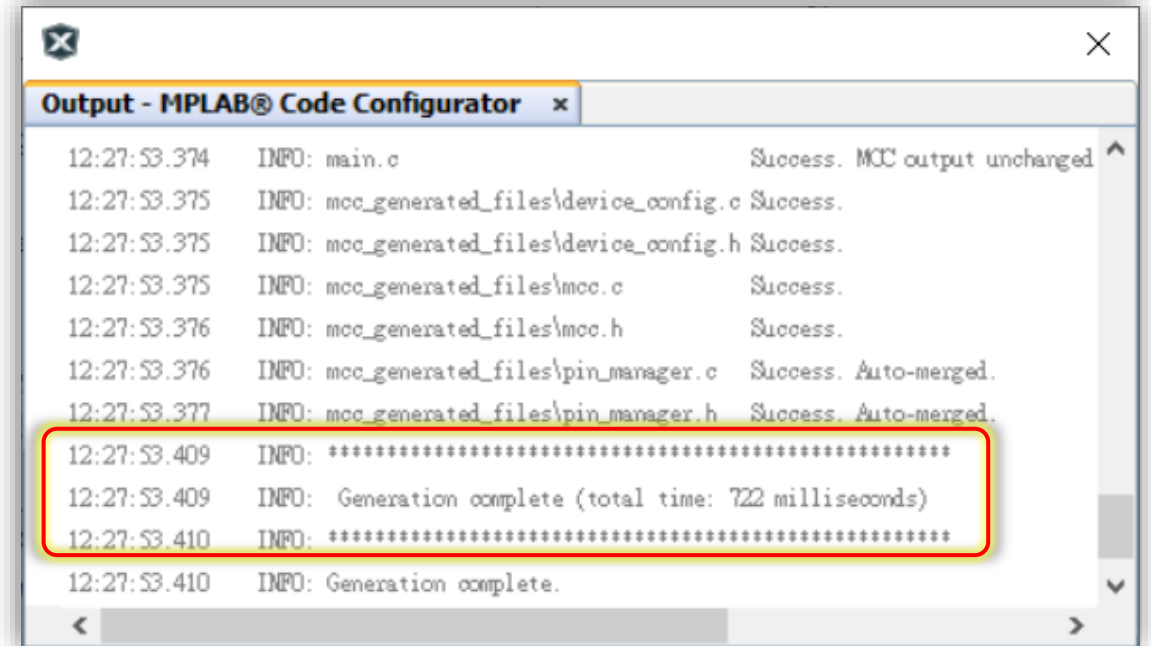
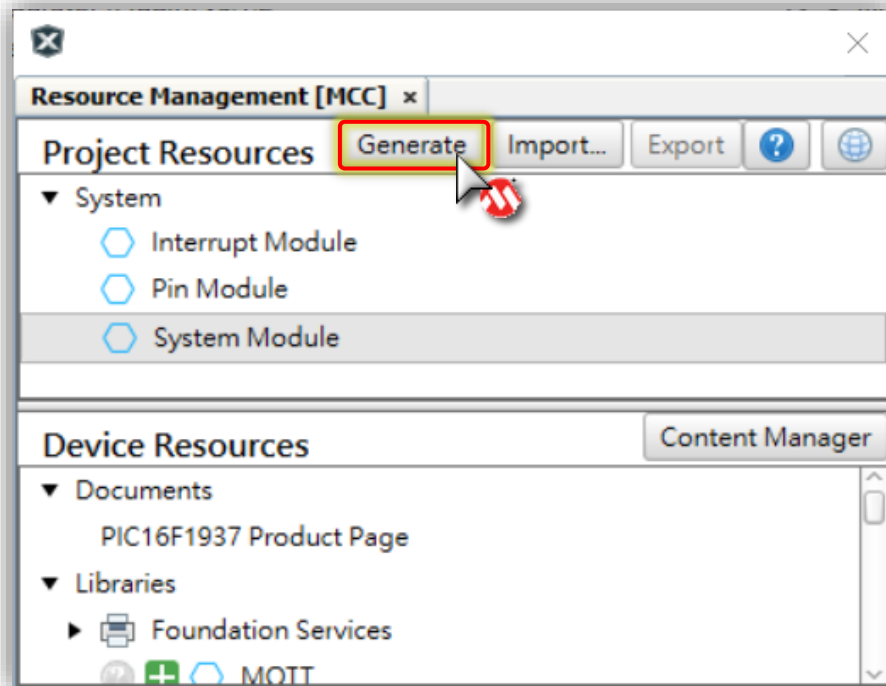
- **Confirm Macro.**



# Lab7 LCD Display

## Step 18

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



# Lab7 LCD Display

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"

void TIM2_ISR(void);
void LCD_Show(uint16_t LCD_Value);

uint16_t ADC_Value=0;
uint8_t Num[4];
volatile uint8_t TMRFlag;

int main(void)
{
    ...
    while (1)
    {
        ...
    }
}

void TIM2_ISR(void)
{
    TMRFlag=1;
}

void LCD_Show(uint16_t LCD_Value)
{
    ...
}
```

```
void LCD_Show(uint16_t LCD_Value)
{
    Num[0] = (LCD_Value / 1000) % 10;
    Num[1] = (LCD_Value / 100) % 10;
    Num[2] = (LCD_Value / 10) % 10;
    Num[3] = (LCD_Value % 10);
    if (LCD_Value >= 1000)
    {
        LCD_MACRO_SYM00Num(Num[0]);
        LCD_MACRO_SYM01Num(Num[1]);
        LCD_MACRO_SYM02Num(Num[2]);
        LCD_MACRO_SYM03Num(Num[3]);
    }
    else if (LCD_Value >= 100)
    {
        LCD_MACRO_SYM00Num(10);
        LCD_MACRO_SYM01Num(Num[1]);
        LCD_MACRO_SYM02Num(Num[2]);
        LCD_MACRO_SYM03Num(Num[3]);
    }
    else if (LCD_Value >= 10)
    {
        LCD_MACRO_SYM00Num(10);
        LCD_MACRO_SYM01Num(10);
        LCD_MACRO_SYM02Num(Num[2]);
        LCD_MACRO_SYM03Num(Num[3]);
    }
    else
    {
        LCD_MACRO_SYM00Num(10);
        LCD_MACRO_SYM01Num(10);
        LCD_MACRO_SYM02Num(10);
        LCD_MACRO_SYM03Num(Num[3]);
    }
}
```

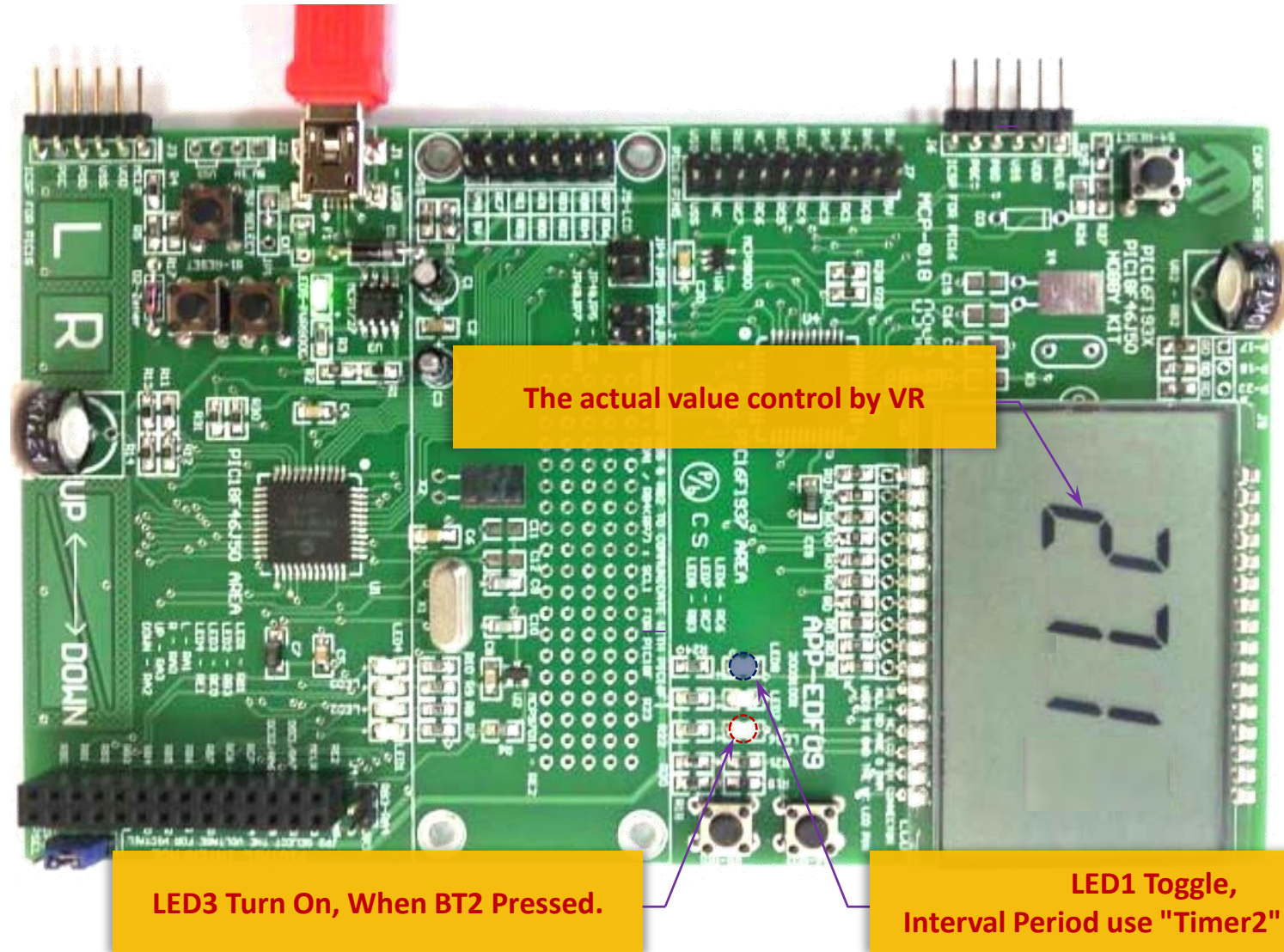
```
int main(void)
{
    ...
    TMR2_SetInterruptHandler(TIM2_ISR);
    while (1)
    {
        // Add your application code

        if(TMRFlag)
        {
            TMRFlag=0;
            LED1_Toggle();
            ADC_Value = ADC_GetConversion(VR);
            LCD_Show(ADC_Value);
        }

        if(!BT2_GetValue())
        {
            LED3_SetLow();
        }
        else
        {
            LED3_SetHigh();
        }
    }
}
```

# Lab7 LCD Display

## Result

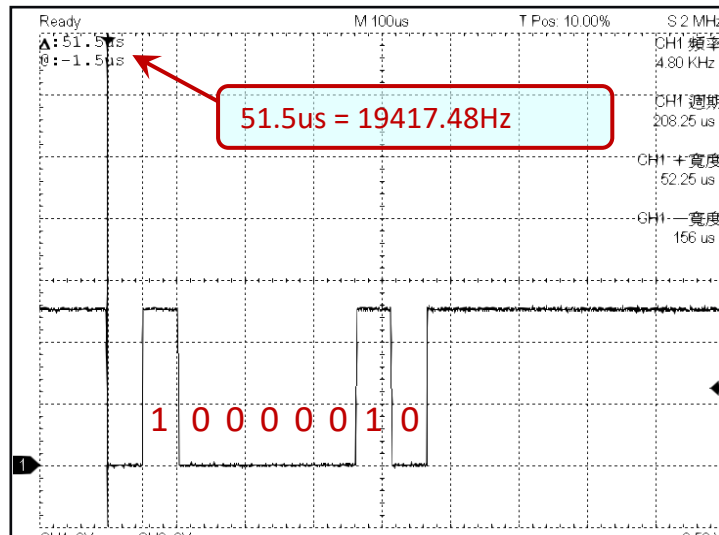


# UART Architecture

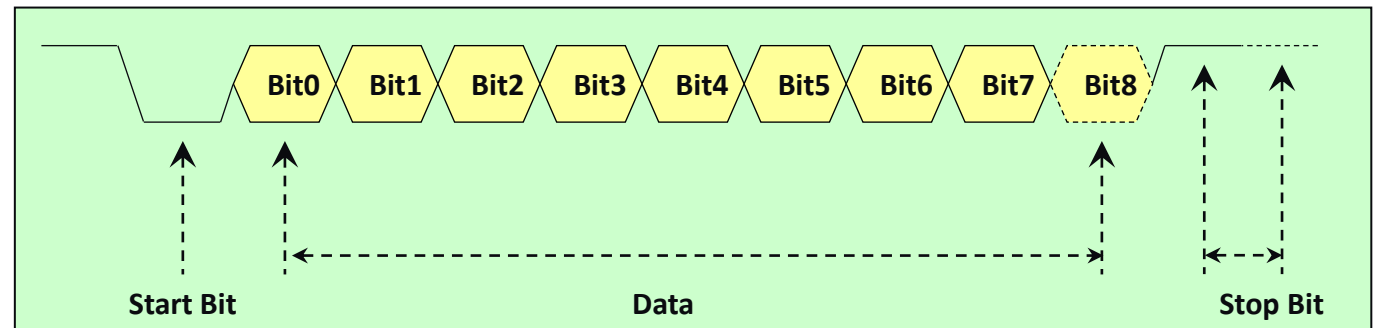
---

# 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

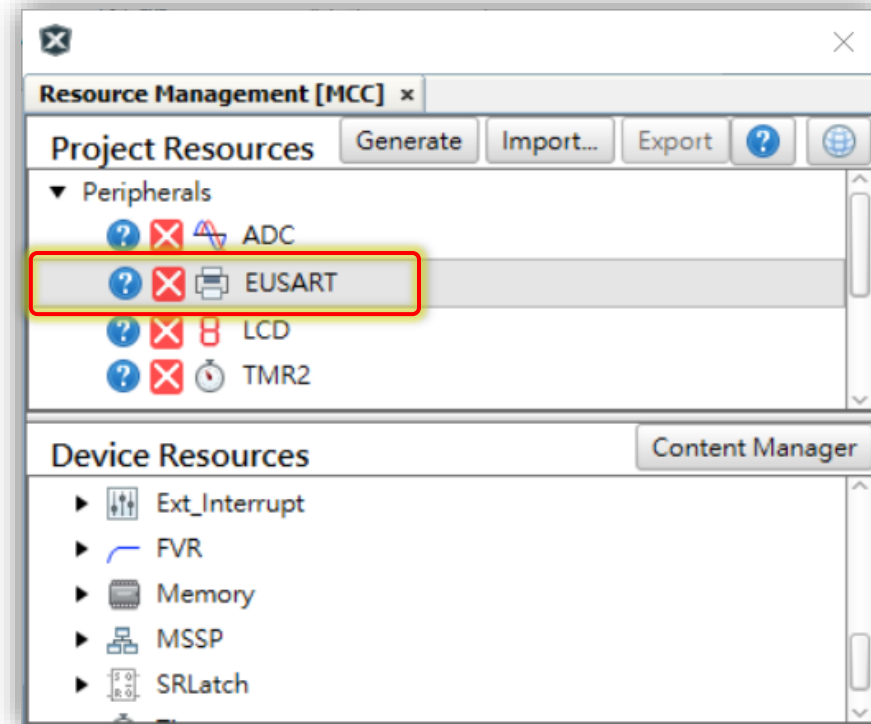
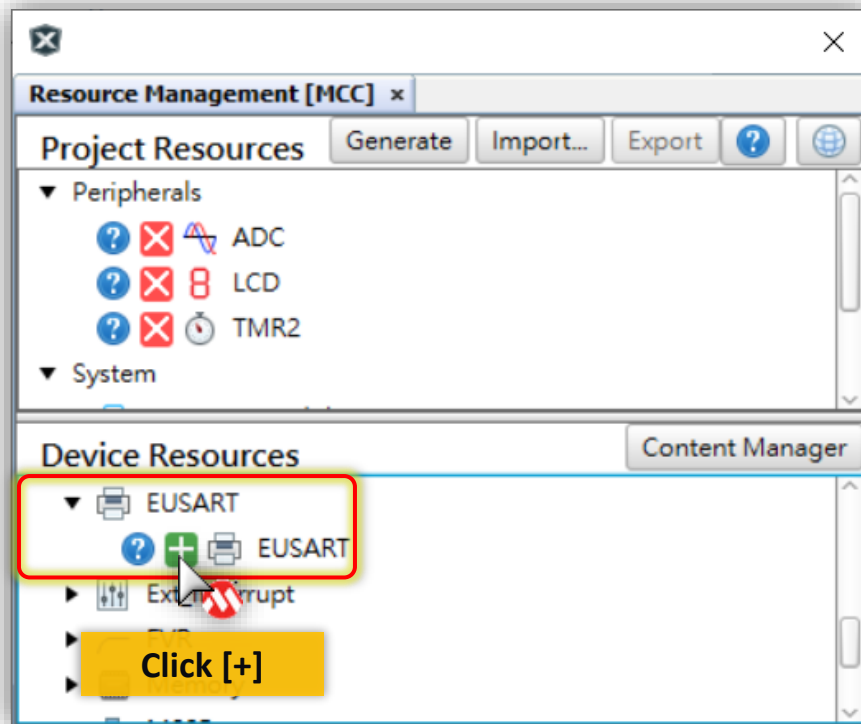


# PIC16F Series UART

- UART module is full-duplex, asynchronous communication., Support **RS-232, RS-485,** and **LIN**.
- The module also supports **8 ~ 9 Data Bits, Half-duplex synchronous master and Slave modes.**
- **2-Deep FIFO** for **RX**.
- Provide LIN bus Auto Baud Detect and calibration, 13 Data bit transmit, Wake-up on Break reception.

# MCC's UART Module

- Add UART resource from Device Resources.



# MCC's UART Module

**EUSART - Editor**

**EUSART**

**Easy Setup** **Registers**

**Hardware Settings**

Mode: asynchronous

☒ Enable EUSART      Baud Rate: 9600      Error: 0.160 %

☒ Enable Transmit      Transmission Bits: 8-bit

☐ Enable Wake-up      Reception Bits: 8-bit

☐ Auto-Baud Detection      Data Polarity: Non-Inverted

☐ Enable Address Detect      ☒ Enable Receive

☐ Enable EUSART Interrupts

**Software Settings**

☐ Redirect STDIO to USART

Software Transmit Buffer Size: 8

Software Receive Buffer Size: 8

Enable UART

Enable Transmit

Enable Receive

Enable Interrupts

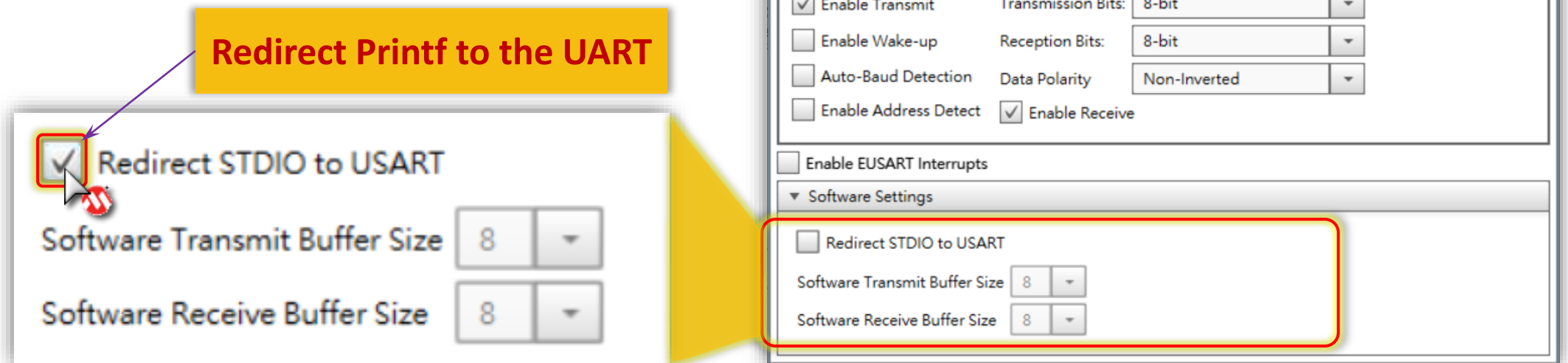
Redirect UART

Baud Rate

Data bits

# 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!! ");**



# Lab Preparation

---

PC Terminal Software Setting

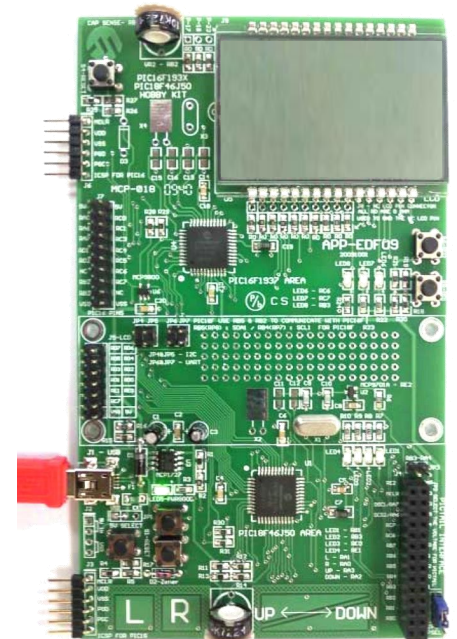
# Lab Preparation

## Step 1

- PL2303TA provides connecting an RS232-like full-duplex asynchronous serial device to any Universal Serial Bus (USB) capable host, include highly compatible drivers could simulate the traditional COM port on most operating systems.



 +5V  
 GND  
 RXD  
 TXD



# Lab Preparation

## Step 2

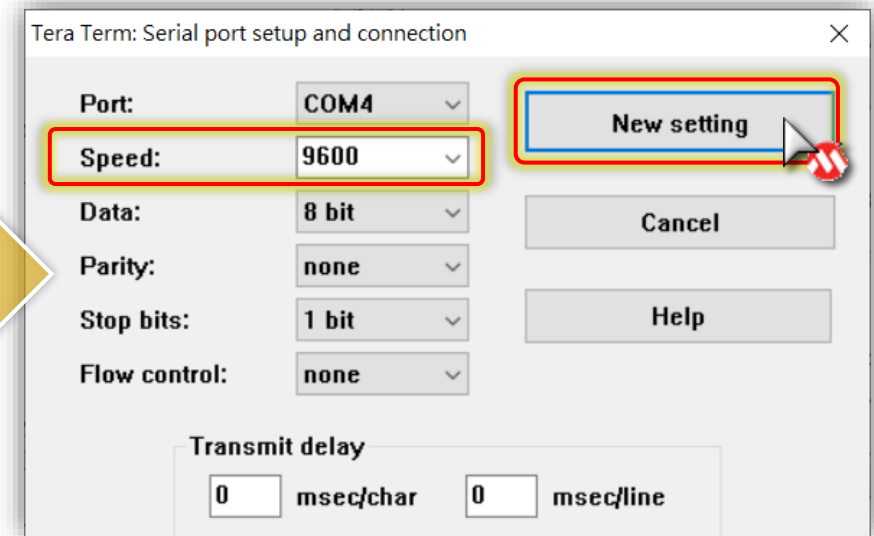
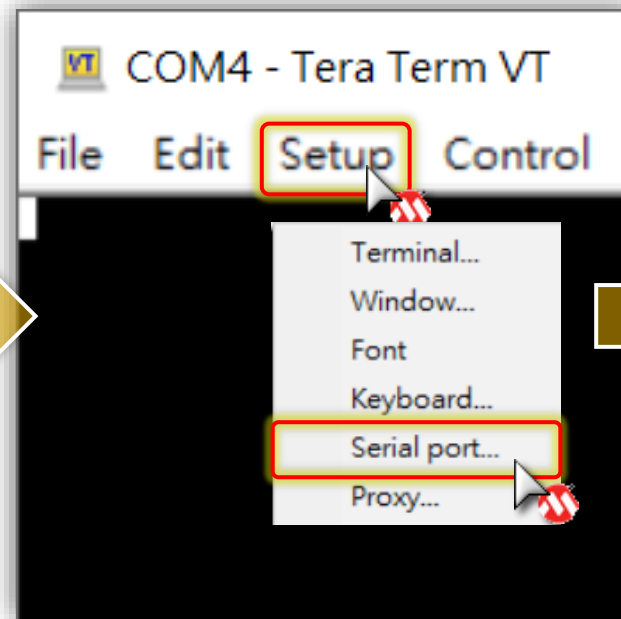
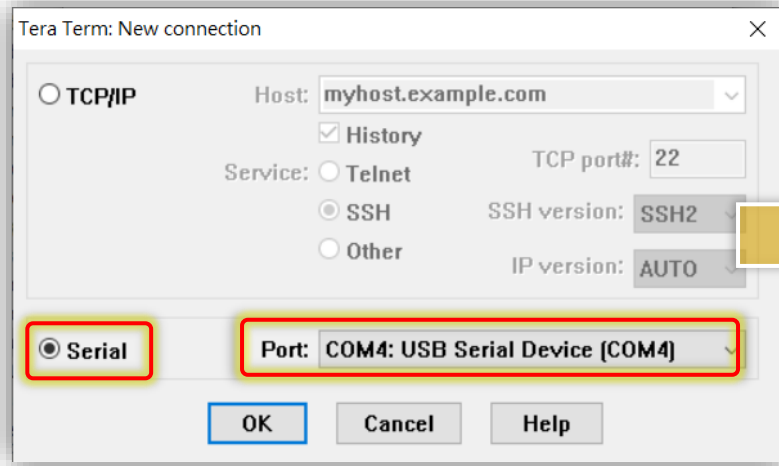
- Please confirm Windows Device Manager on your PC.
- For Window 7, you might need install driver before Labs.  
**Driver at : \Appendix\Serial Emulator Driver\mchpcdc.inf**
- You will see a **USB Serial Port (COMx)** at Device Manager.



# Lab Preparation

## Step 3

- 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**, **9600**, **8 bit**, **none**, **1 bit**, **none**



# Lab8 UART printf

- Base on current project or Open `.\Lab8_xxx.X` to do exercises
- Try to add UART1 module then output VR ADC value .
- UART set to **9600**, **N**, **8**, **1**, **No flow control**.
- **Disable LED2(RC6) and LED3(RC7) GPIO output function.**
- UART has connected by PCB layout.
  - PIC16F1937 (**UART-TX**, **RC6**) <-> PL2303TA (**RX**)
  - PIC16F1937 (**UART-RX**, **RC7**) <-> PL2303TA (**TX**)

# Let's go!

# Lab8 UART printf

## Step 1

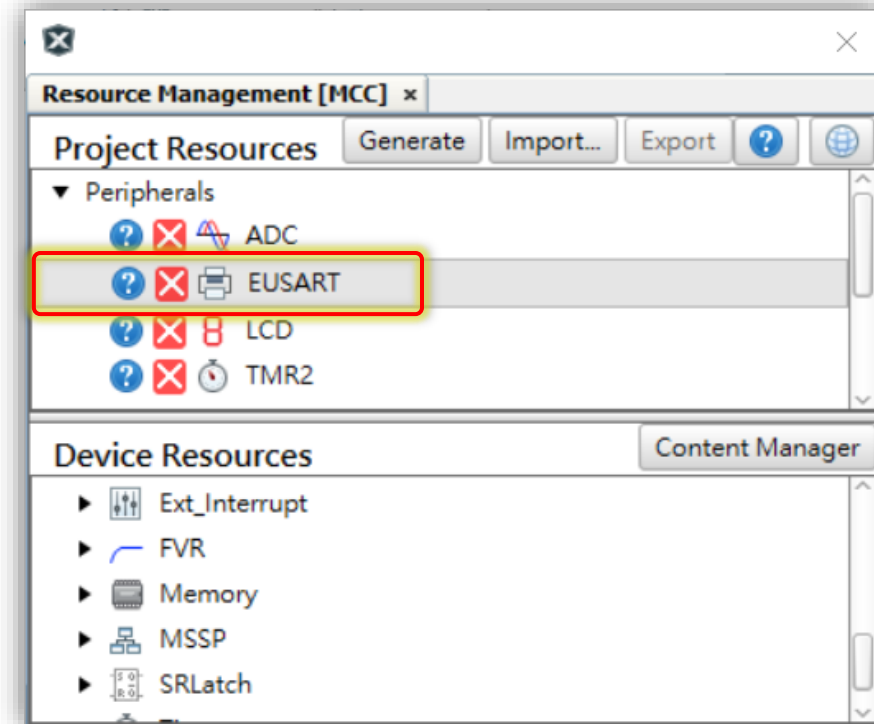
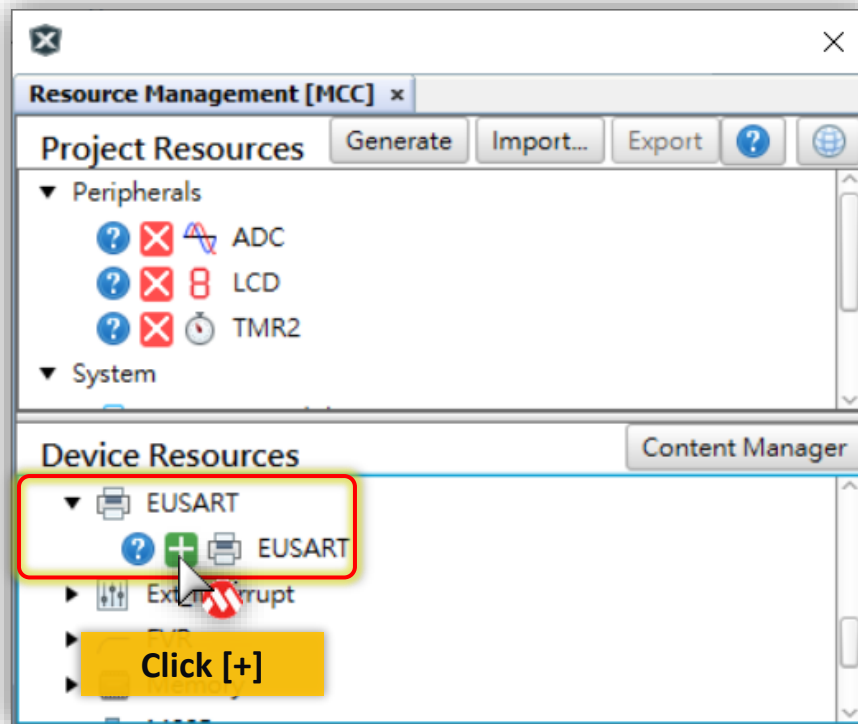
- **Disable RC6, RC7 GPIO output function.**

| Pin Manager: Grid View |          |           | Pin No: 8 9 10 11 14 15 16 17 32 35 36 37 42 43 44 1 38 39 40 41 2 3 4 5 |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
|------------------------|----------|-----------|--------------------------------------------------------------------------|---|---|---|---|---|---|--------|---|---|---|---|---|---|--------|---|---|---|---|---|---|---|---|---|--|--|--|--|
|                        |          |           | Port B                                                                   |   |   |   |   |   |   | Port C |   |   |   |   |   |   | Port D |   |   |   |   |   |   |   |   |   |  |  |  |  |
| Module                 | Function | Direction | 0                                                                        | 1 | 2 | 3 | 4 | 5 | 6 | 7      | 0 | 1 | 2 | 3 | 4 | 5 | 6      | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |  |  |  |
| ADC                    | ANx      | input     | 🔒                                                                        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
|                        | VREF+    | input     |                                                                          |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
|                        | VREF-    | input     |                                                                          |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
| LCD                    | COMx     | output    |                                                                          |   |   |   | 🔒 | 🔒 |   |        |   |   |   |   |   |   |        |   | 🔒 |   |   |   |   |   |   |   |  |  |  |  |
|                        | SEGx     | output    | 🔒                                                                        |   |   |   |   |   | 🔒 | 🔒      |   |   | 🔒 | 🔒 | 🔒 | 🔒 | 🔒      | 🔒 |   |   |   | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |  |  |  |  |
|                        | VLCDx    | input     |                                                                          | 🔒 | 🔒 | 🔒 |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
| OSC                    | CLKOUT   | output    |                                                                          |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
| Pin Module             | GPIO     | input     | 🔒                                                                        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒      | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒      | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |  |  |  |  |
|                        | GPIO     | output    | 🔒                                                                        | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒      | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒      | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 | 🔒 |  |  |  |  |
| RESET                  | MCLR     | input     |                                                                          |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |
|                        | VCAP     | input     |                                                                          |   |   |   |   |   |   |        |   |   |   |   |   |   |        |   |   |   |   |   |   |   |   |   |  |  |  |  |

# Lab8 UART printf

## Step 2

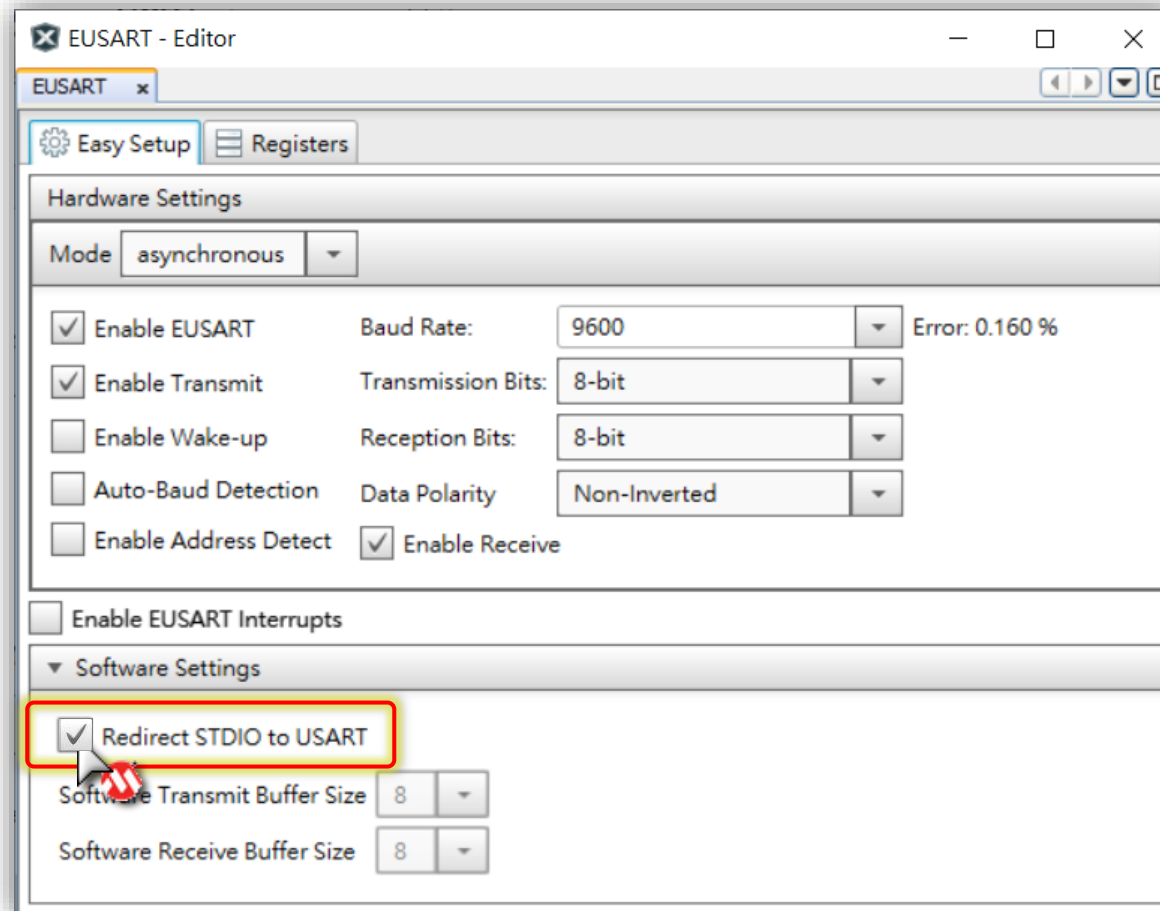
- Device Resources
  - EUSART ► EUSART



# Lab8 UART printf

## Step 3

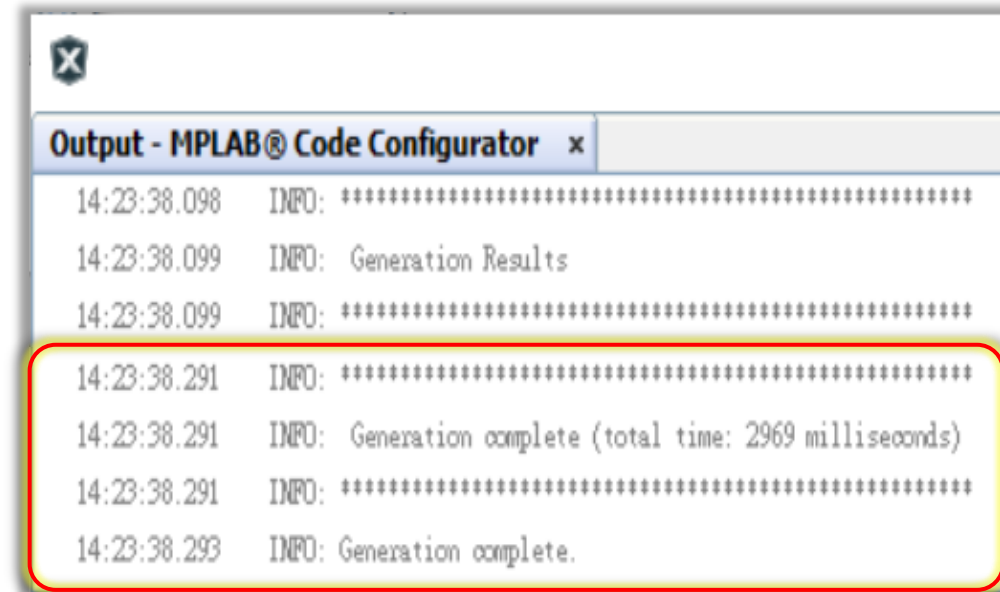
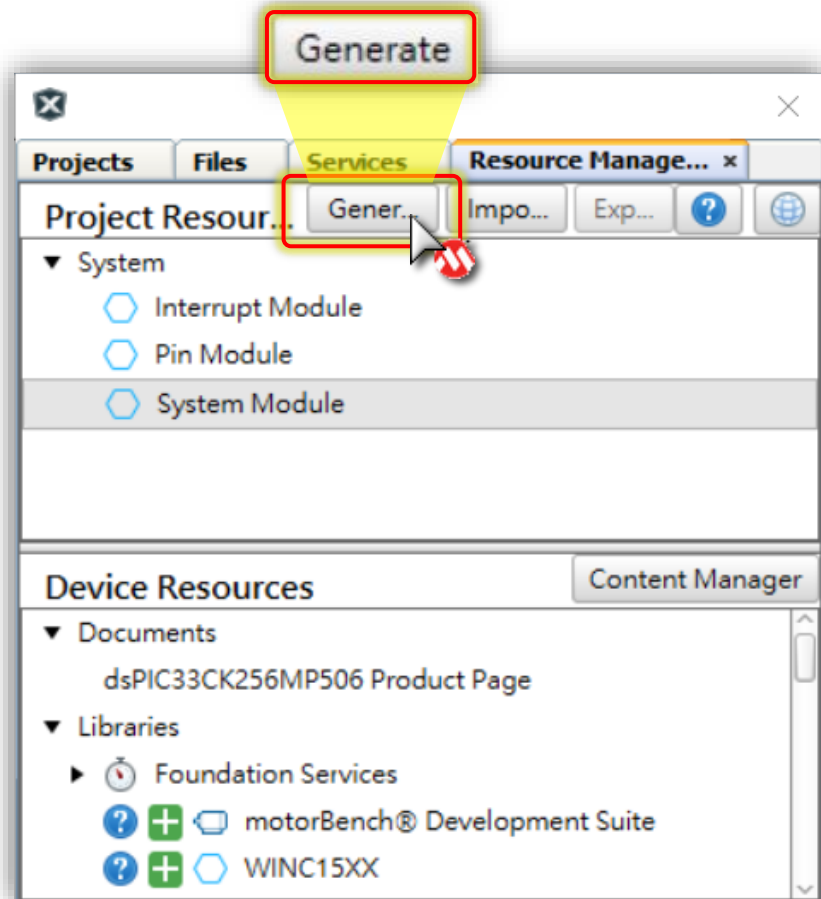
- **Enable Redirect STDIO to USART.**



# Lab8 UART printf

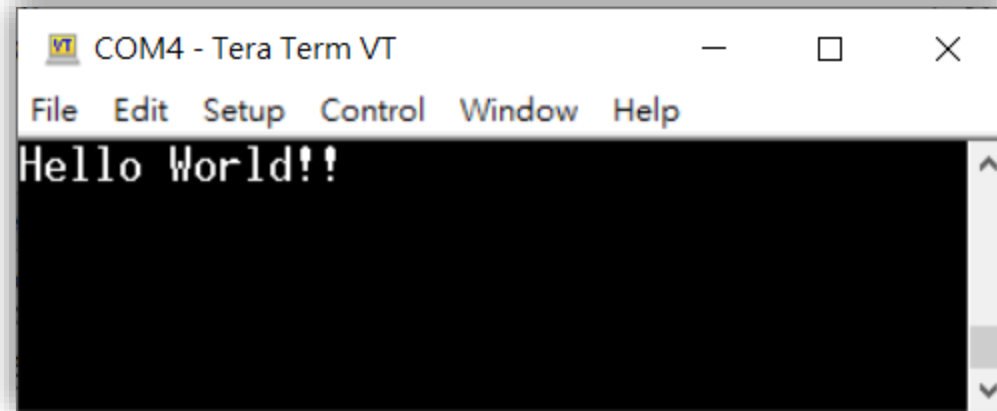
## Step 4

- Generate 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)

# Lab8 UART printf

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"
#include <stdio.h>

void TIM2_ISR(void);

uint16_t ADC_Value=0;
volatile uint8_t TMRFlag;

int main(void)
{
    ...
    while (1)
    {
        ...
    }
}

void TIM2_ISR(void)
{
    TMRFlag = 1;
}

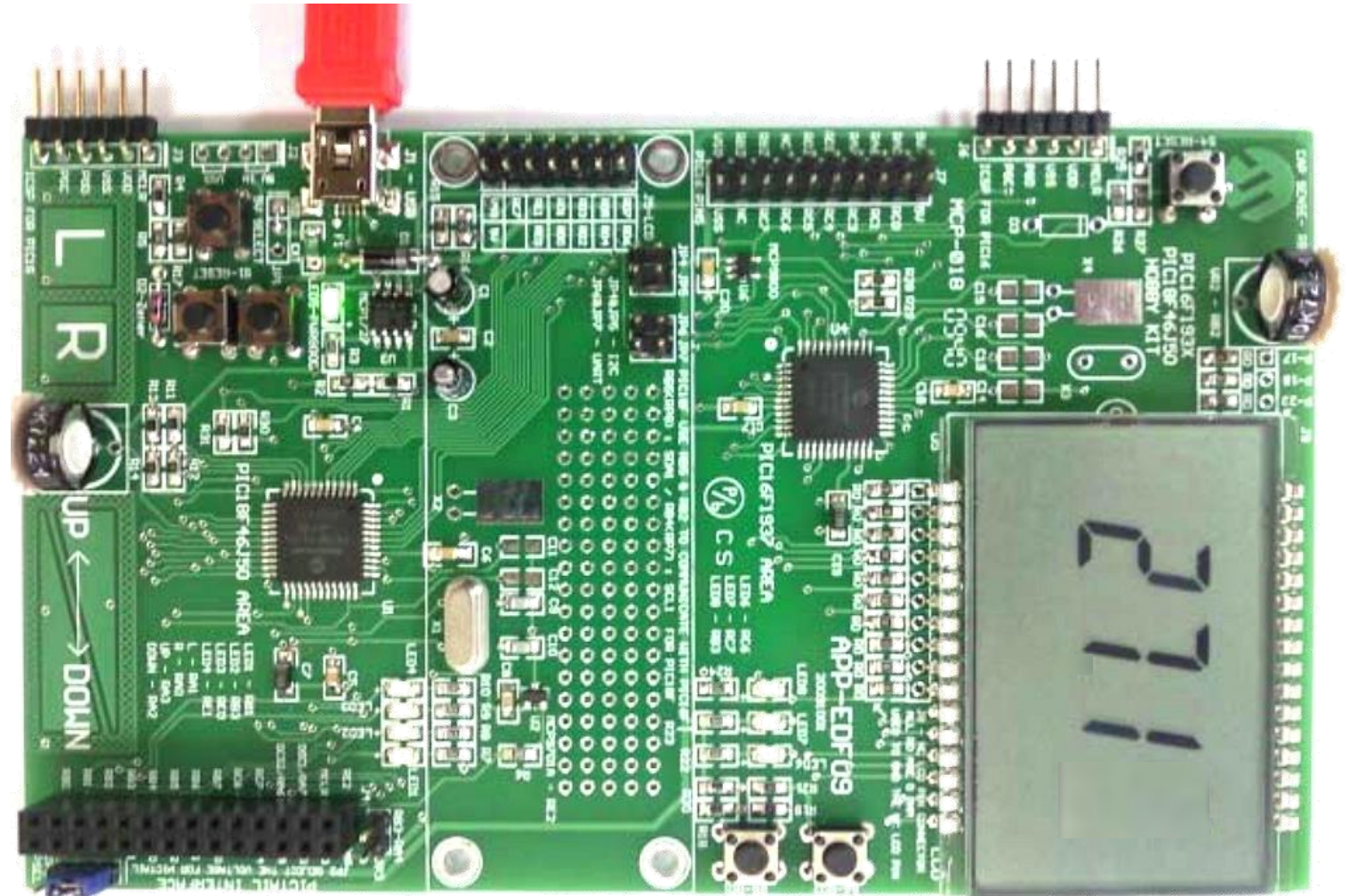
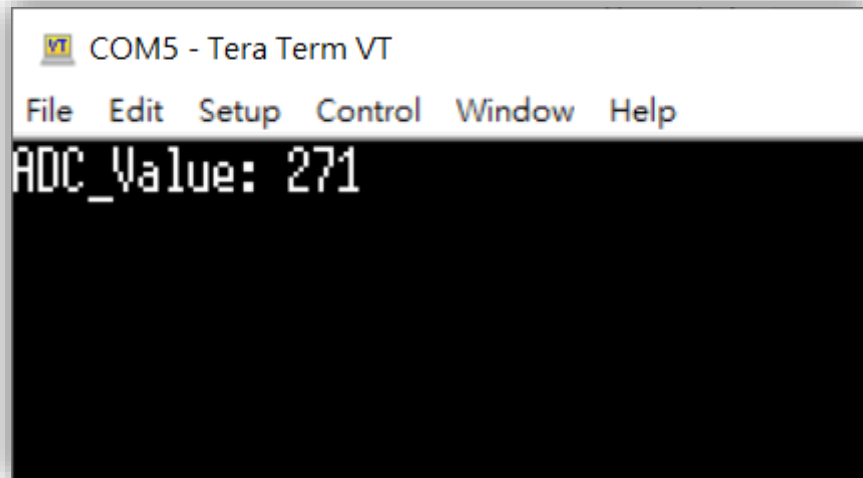
void LCD_Show(uint16_t LCD_Value)
{
    ...
}
```

```
int main(void)
{
    ...
    TMR2_SetInterruptHandler(TIM2_ISR);
    printf("\033[2J");
    printf("\033[?25l");
    while (1)
    {
        // Add your application code

        if (TMRFlag)
        {
            TMRFlag = 0;
            LED1_Toggle();
            ADC_Value = ADC_GetConversion(VR);
            LCD_Show(ADC_Value);
            printf("\033[1;1HADC_Value:%4d", ADC_Value);
        }
    }
}
```

# Lab8 UART printf

## Result



# MCC's UART Interrupt

- Polling mode not easy to catch that the data from PC to UART.
- This is very easy if enable UART interrupt to be received it.
- Receive and transmit buffer(queue) are implemented ready, so ease to use and don't worrying data lost.

# UART Functions of MCC

- MCC Provide below functions for UART:

*Prototype definition : "eusart.h"*

- void EUSART\_Initialize(void);
- bool EUSART\_is\_tx\_ready(void);
- **bool EUSART\_is\_rx\_ready(void);**
- bool EUSART\_is\_tx\_done(void);
- **uint8\_t EUSART\_Read(void);**
- void EUSART\_Write(uint8\_t txData);
- void EUSART\_Transmit\_ISR(void);
- void EUSART\_Receive\_ISR(void);
- void EUSART\_RxDataHandler(void);
- void EUSART\_SetFramingErrorHandler(void (\* interruptHandler)(void));
- void EUSART\_SetOverrunErrorHandler(void (\* interruptHandler)(void));
- void EUSART\_SetErrorHandler(void (\* interruptHandler)(void));

# Lab9 UART Communication

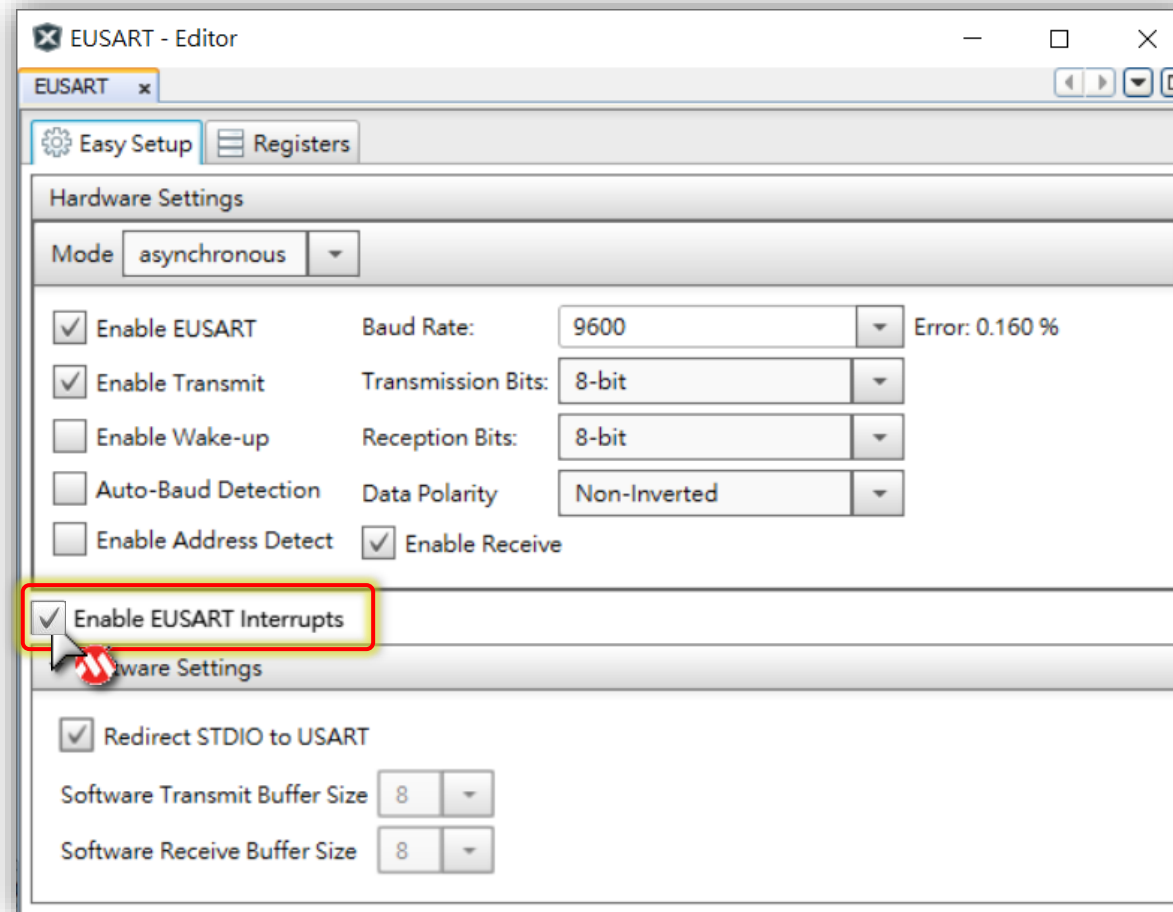
- Base on current project or Open `.\Lab9_xxx.X` to do exercises
- Try to enable UART interrupt to be received UART data come from PC.
- To set a suitable received buffer size.

**Let's go!**

# Lab9 UART Communication

## Step 1

- **Enable EUSART Interrupts.**

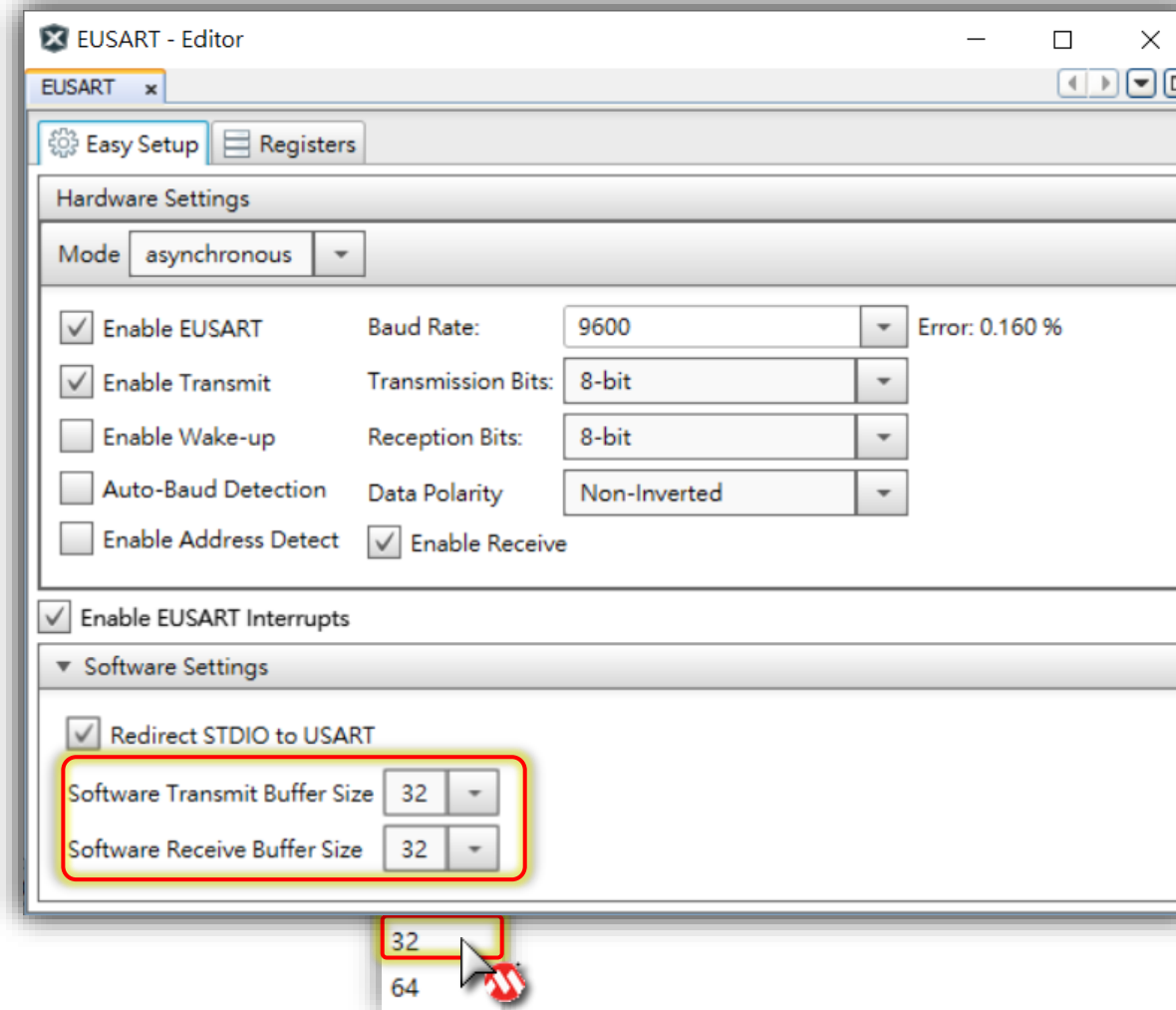


# Lab9 UART Communication

## Step 2

- **Buffer Size.**

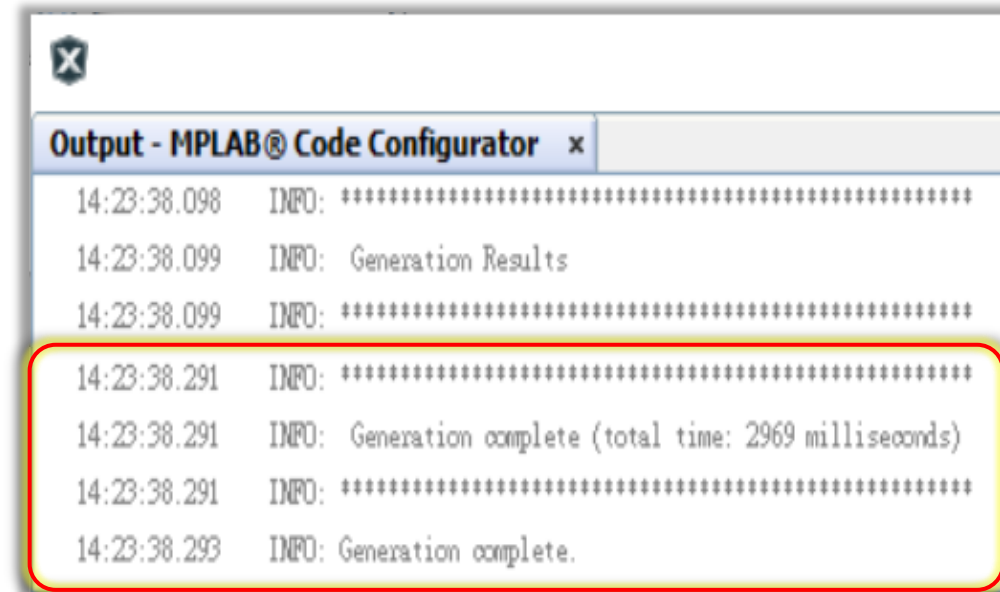
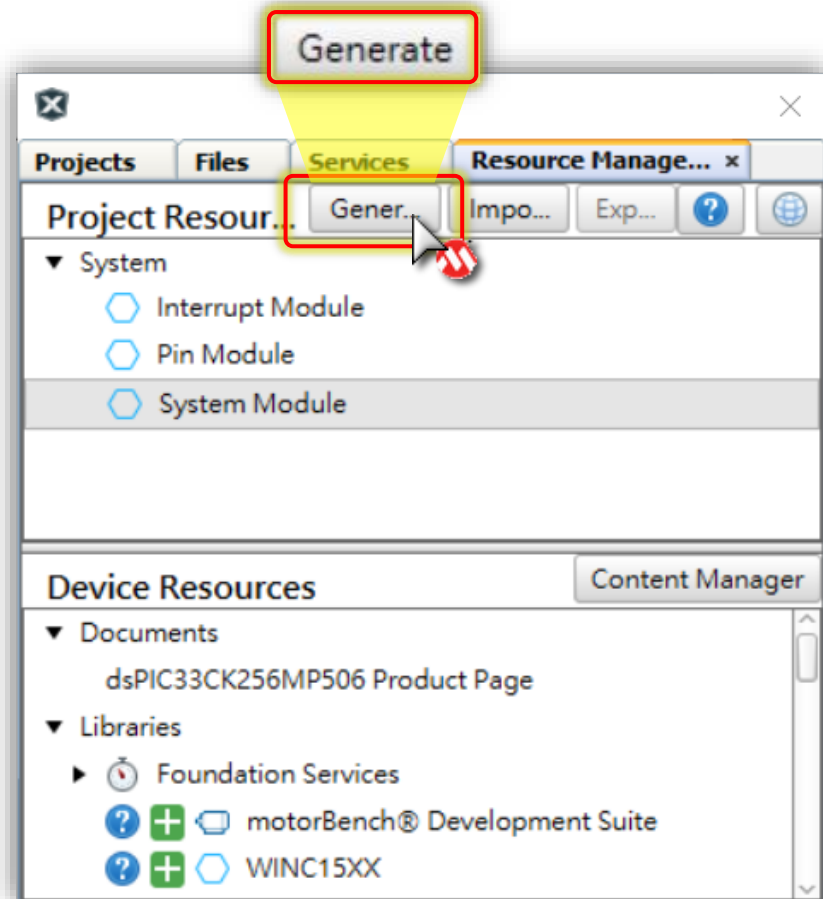
- **8 ► 32**



# Lab9 UART Communication

## Step 3

- Generate Code.



# Lab9 UART Communication

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"
#include <stdio.h>

void TIM2_ISR(void);

uint16_t ADC_Value=0;
volatile uint8_t TMRFlag;
uint8_t Rx_Buffer[1];

int main(void)
{
    ...
    while (1)
    {
        ...
    }
}

void TIM2_ISR(void)
{
    TMRFlag = 1;
}

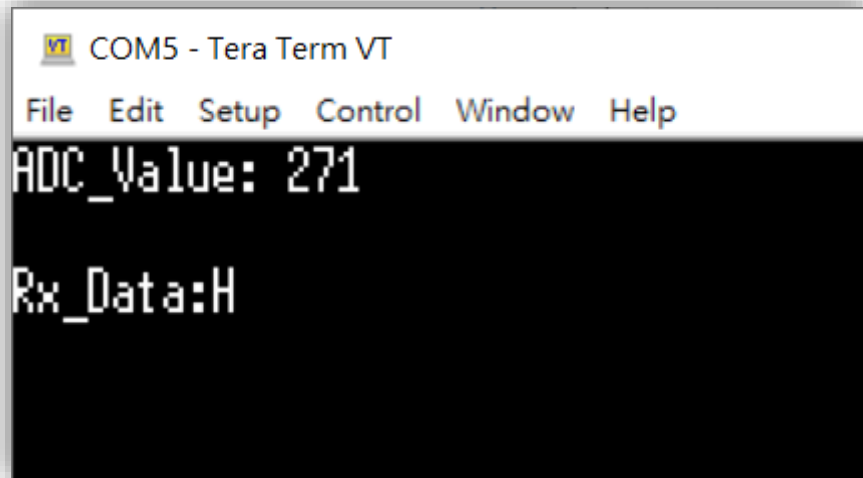
void LCD_Show(uint16_t LCD_Value)
{
    ...
}
```

```
int main(void)
{
    ...
    TMR2_SetInterruptHandler(TIM2_ISR);
    printf("\033[2J");
    printf("\033[?25l");
    while (1)
    {
        // Add your application code

        if (TMRFlag)
        {
            TMRFlag = 0;
            LED1_Toggle();
            ADC_Value = ADC_GetConversion(VR);
            LCD_Show(ADC_Value);
            printf(" \033[1;1HADC_Value:%4d", ADC_Value);
        }
        if (EUSART_is_rx_ready())
        {
            Rx_Buffer[0] = EUSART_Read();
            printf(" \033[3;1HRx_Data:%1c", Rx_Buffer[0]);
        }
    }
}
```

# Lab9 UART Communication

## Result

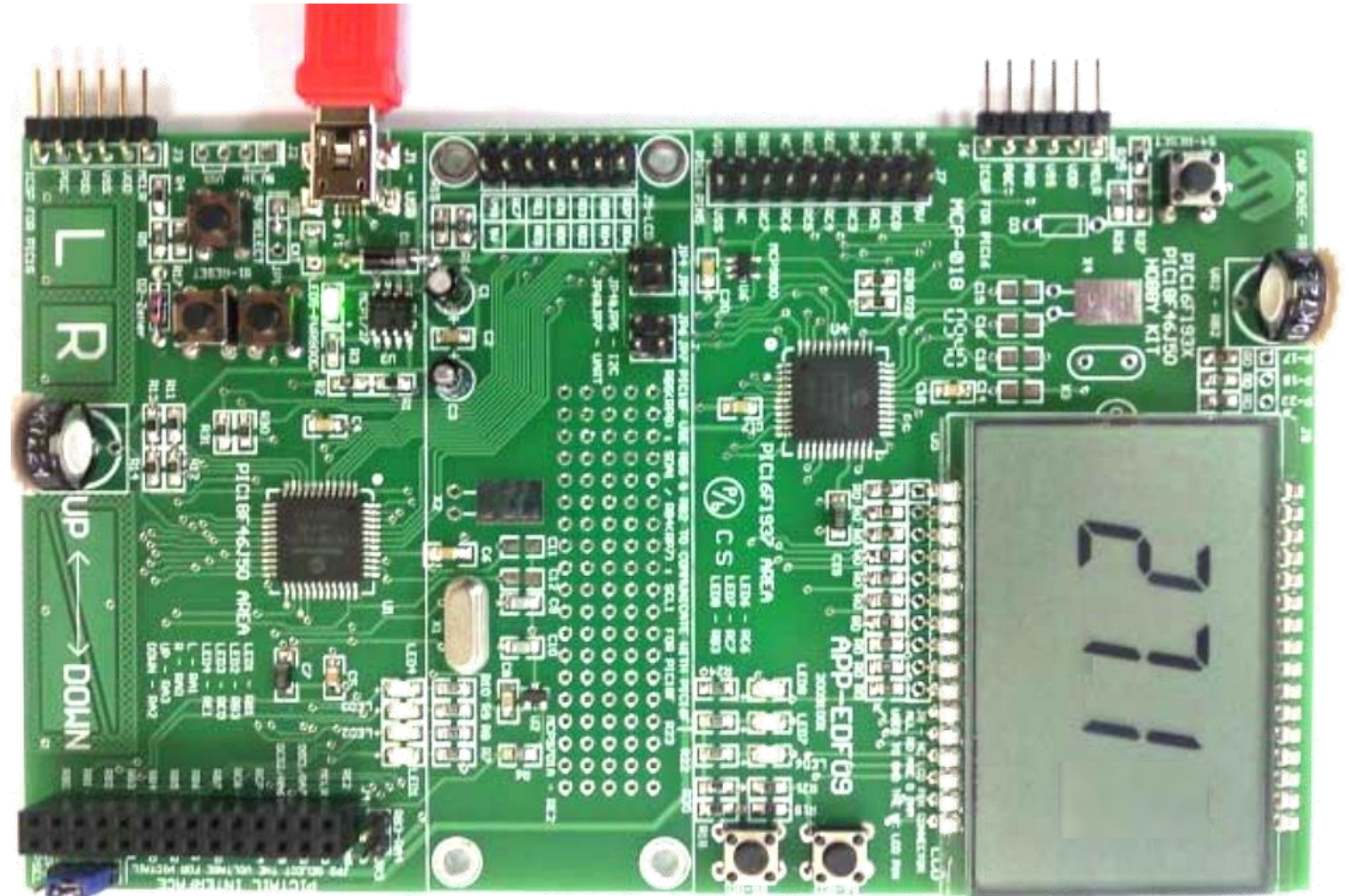


COM5 - Tera Term VT

File Edit Setup Control Window Help

ADC\_Value: 271

Rx\_Data:H

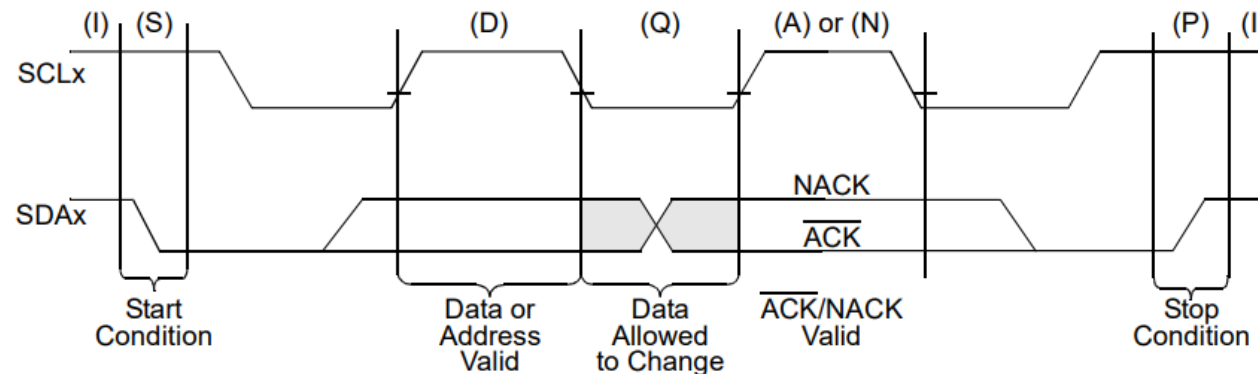


# I2C Architecture

---

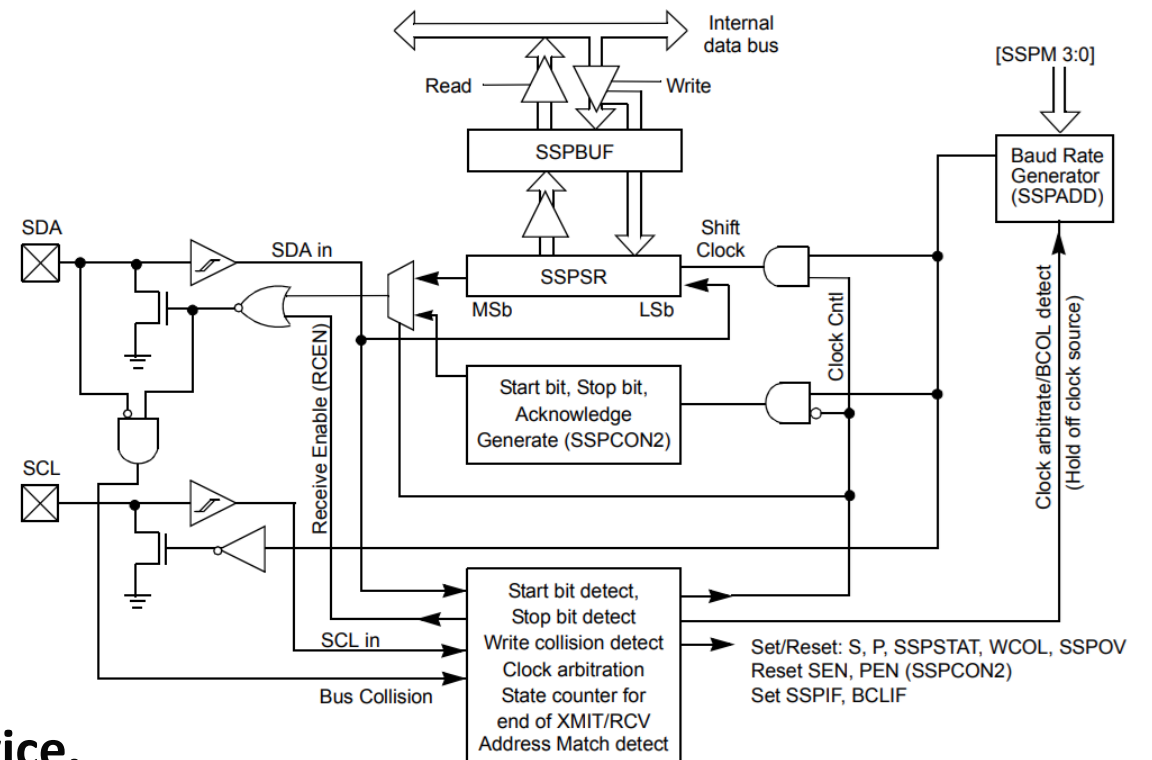
# What's I2C

- I2C, Inter-Integrated Circuit
- It's a serial communication interface.  
Synchronous, inside-board level, short distance communication.
- The data is clocked along with a clock signal(SCL).
- The clock signal controls when data is changed and when it should be read.
- Since I2C is synchronous, the clock rate can vary, unlike asynchronous (RS-232 style) communications.



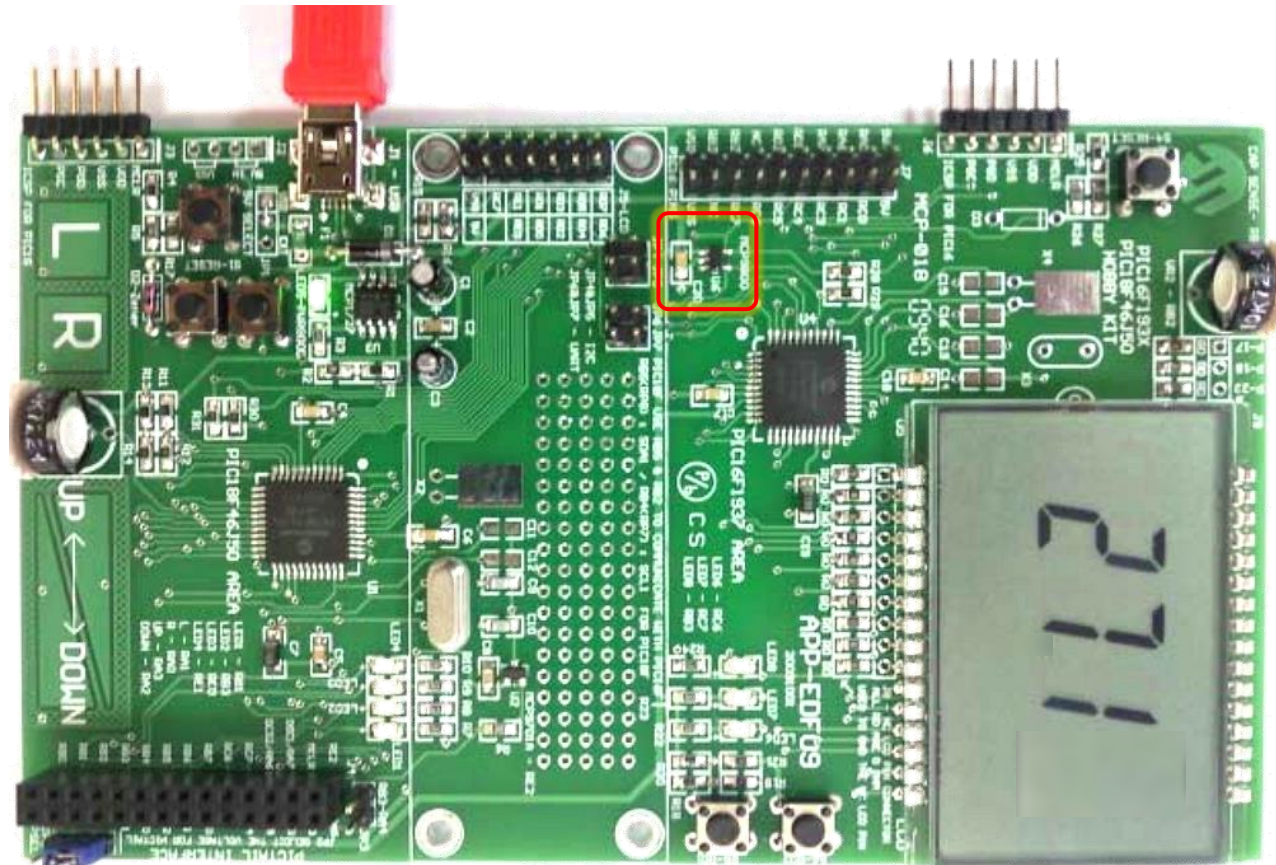
# PIC16F Series MSSP Module

- The Master Synchronous Serial Port (MSSP) module can operate in one of two modes:
  - Serial Peripheral Interface (SPI).
  - **Inter-Integrated Circuit (I2C).**
- **Serial Peripheral Interface (SPI)**
  - Master and Slave mode.
  - Clock Parity.
  - Slave Select Synchronization (Slave mode only).
- **Inter-Integrated Circuit (I2)**
  - Master and Slave mode.
  - Support limited Multi-Master.
  - Support 7-bit and 10-bit addressing.
- **Pin Definition :**
  - **SCL (Serial Clock)** : Clock, PIC16F -> Device.
  - **SDA (Serial Data)** : Data In/Out, PIC16F <-> Device.



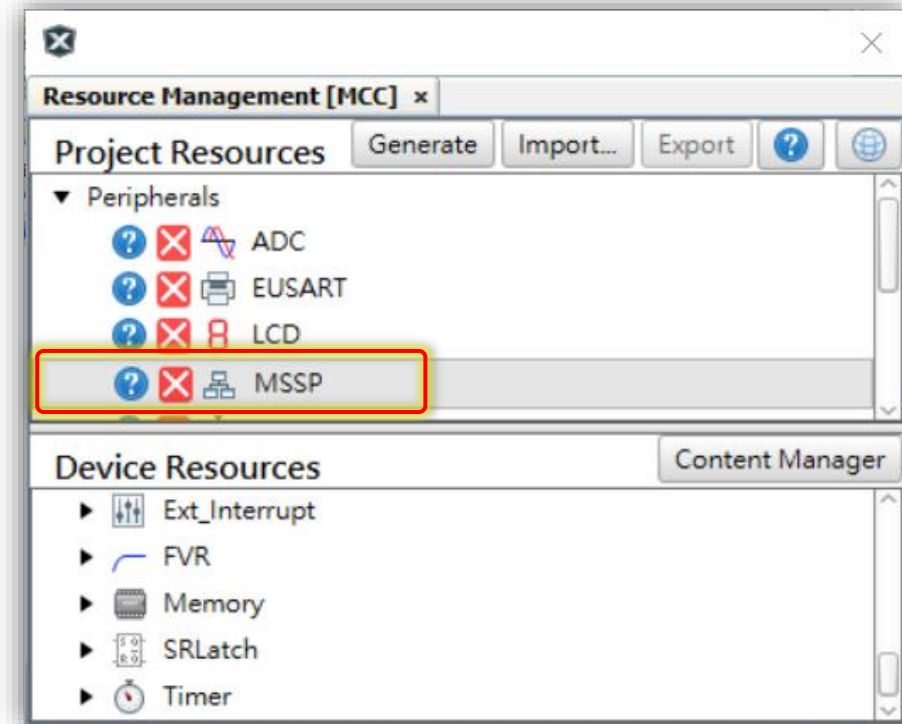
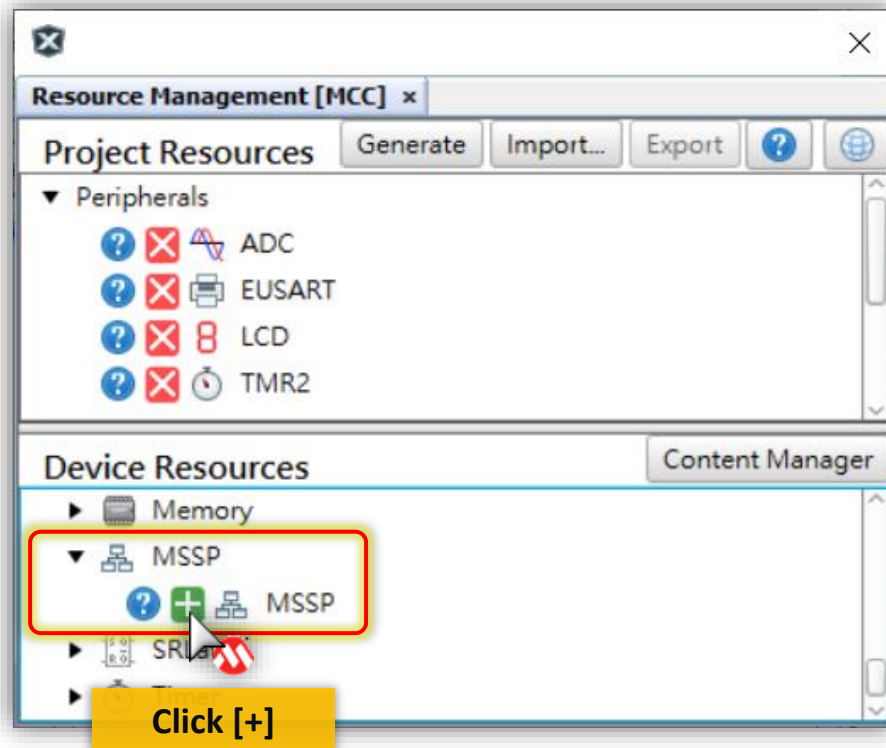
# I2C Device

- Temperature Sensor (MCP9800) :
  - MCP9800 is a digital temperature sensor converted to digital word with a user selectable 9 to 12-bit Sigma-Delta Analog to Digital Converter.  $\pm 0.5^{\circ}\text{C}$  (typical) at  $+25^{\circ}\text{C}$



# MCC's MSSP Module

- Add MSSP resource from Device Resources.



# MCC's MSSP Module

Enable Interrupt

Select Protocol

Select Mode

Clock Frequency

Enable Interrupt

MSSP - Editor

MSSP x

Easy Setup Registers

▼ Software Settings

Interrupt Driven: ☐

▼ Hardware Settings

Serial Protocol: I2C

Mode: Master

I2C Clock Frequency(Hz): 3907 ≤ 100000 ≤ 250000

Actual Clock Frequency(Hz): 100000.00

▶ Advanced Settings

▼ Interrupt Settings

Enable I2C Interrupt: ☐

# I2C Functions of MCC

- MCC Provide below functions for I2C:

Prototype definition : "i2c\_master\_example.h"

- `uint8_t I2C_Read1ByteRegister(i2c_address_t address, uint8_t reg);`
- `uint16_t I2C_Read2ByteRegister(i2c_address_t address, uint8_t reg);`
- `void I2C_Write1ByteRegister(i2c_address_t address, uint8_t reg, uint8_t data);`
- `void I2C_Write2ByteRegister(i2c_address_t address, uint8_t reg, uint16_t data);`
- `void I2C_WriteNBytes(i2c_address_t address, uint8_t *data, size_t len);`
- `void I2C_ReadNBytes(i2c_address_t address, uint8_t *data, size_t len);`
- `void I2C_ReadDataBlock(i2c_address_t address, uint8_t reg, uint8_t *data, size_t len);`

# Lab10 Temperature Sensor

- Base on current project or Open `.\Lab10_xxx.X` to do exercises
- Try to set MCP9800 ADC resolution to 12 bits.
- Read the value from Temperature Sensor and use UART to output to PC.

Let's go!

# MCP9800 Read & Configuration

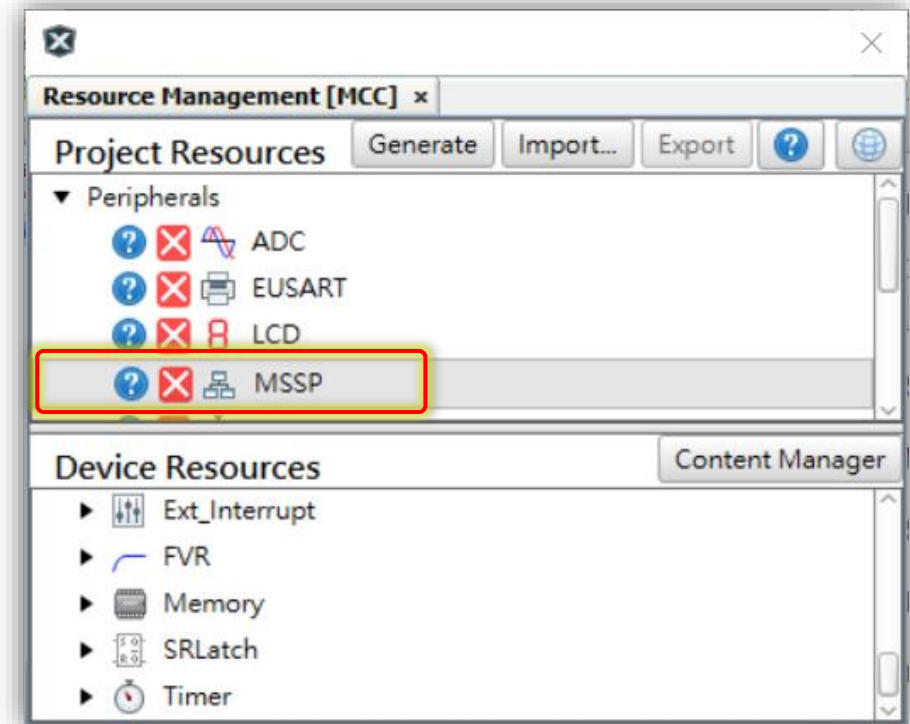
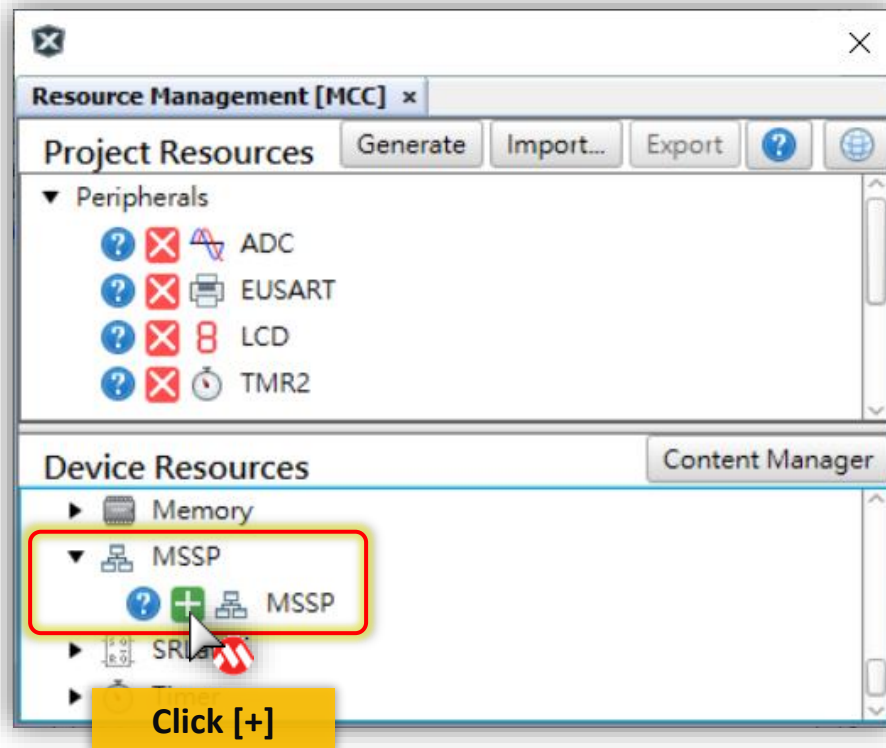
- First, Set ADC resolution & resolution & other config you want.
- Secondly, Read ADC value maximum valid 12 bits. (2 Bytes).

| TABLE 5-1: BIT ASSIGNMENT SUMMARY FOR ALL REGISTERS  |             |                    |                    |                    |                    |                   |                   |                   |                   |
|------------------------------------------------------|-------------|--------------------|--------------------|--------------------|--------------------|-------------------|-------------------|-------------------|-------------------|
| Register Pointer<br>P1 P0                            | MSB/<br>LSB | Bit Assignment     |                    |                    |                    |                   |                   |                   |                   |
|                                                      |             | 7                  | 6                  | 5                  | 4                  | 3                 | 2                 | 1                 | 0                 |
| Ambient Temperature Register (T <sub>A</sub> )       |             |                    |                    |                    |                    |                   |                   |                   |                   |
| 0 0                                                  | MSB         | Sign               | 2 <sup>6</sup> °C  | 2 <sup>5</sup> °C  | 2 <sup>4</sup> °C  | 2 <sup>3</sup> °C | 2 <sup>2</sup> °C | 2 <sup>1</sup> °C | 2 <sup>0</sup> °C |
|                                                      | LSB         | 2 <sup>-1</sup> °C | 2 <sup>-2</sup> °C | 2 <sup>-3</sup> °C | 2 <sup>-4</sup> °C | 0                 | 0                 | 0                 | 0                 |
| Sensor Configuration Register (CONFIG)               |             |                    |                    |                    |                    |                   |                   |                   |                   |
| 0 1                                                  | LSB         | One-Shot           | Resolution         |                    | Fault Queue        |                   | ALERT<br>Polarity | COMP/INT          | Shutdown          |
| Temperature Hysteresis Register (T <sub>HYST</sub> ) |             |                    |                    |                    |                    |                   |                   |                   |                   |
| 1 0                                                  | MSB         | Sign               | 2 <sup>6</sup> °C  | 2 <sup>5</sup> °C  | 2 <sup>4</sup> °C  | 2 <sup>3</sup> °C | 2 <sup>2</sup> °C | 2 <sup>1</sup> °C | 2 <sup>0</sup> °C |
|                                                      | LSB         | 2 <sup>-1</sup> °C | 0                  | 0                  | 0                  | 0                 | 0                 | 0                 | 0                 |
| Temperature Limit-Set Register (T <sub>SET</sub> )   |             |                    |                    |                    |                    |                   |                   |                   |                   |
| 1 1                                                  | MSB         | Sign               | 2 <sup>6</sup> °C  | 2 <sup>5</sup> °C  | 2 <sup>4</sup> °C  | 2 <sup>3</sup> °C | 2 <sup>2</sup> °C | 2 <sup>1</sup> °C | 2 <sup>0</sup> °C |
|                                                      | LSB         | 2 <sup>-1</sup> °C | 0                  | 0                  | 0                  | 0                 | 0                 | 0                 | 0                 |

# Lab10 Temperature Sensor

## Step 1

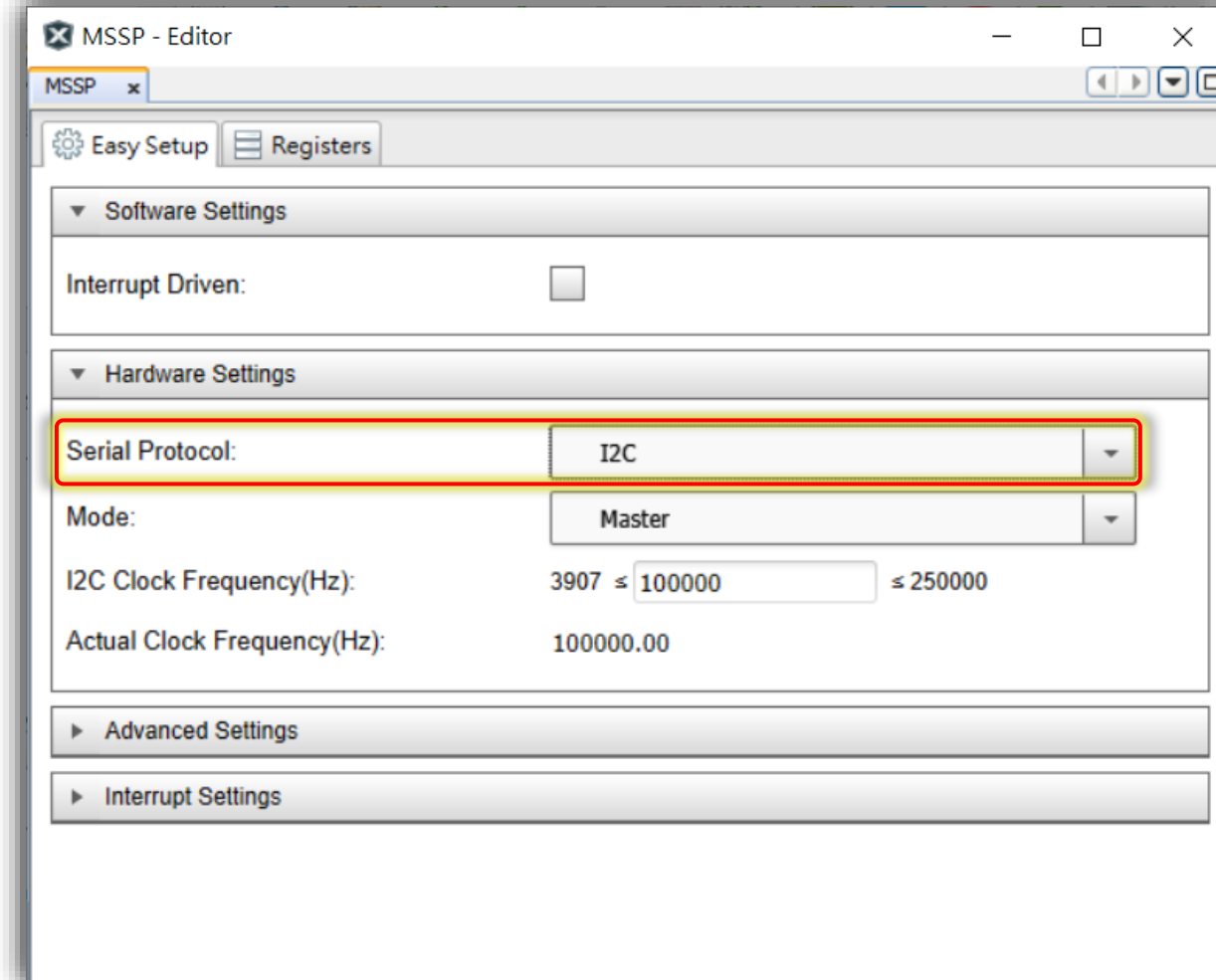
- Device Resources
  - **MSSP** ▶ **MSSP**



# Lab10 Temperature Sensor

## Step 2

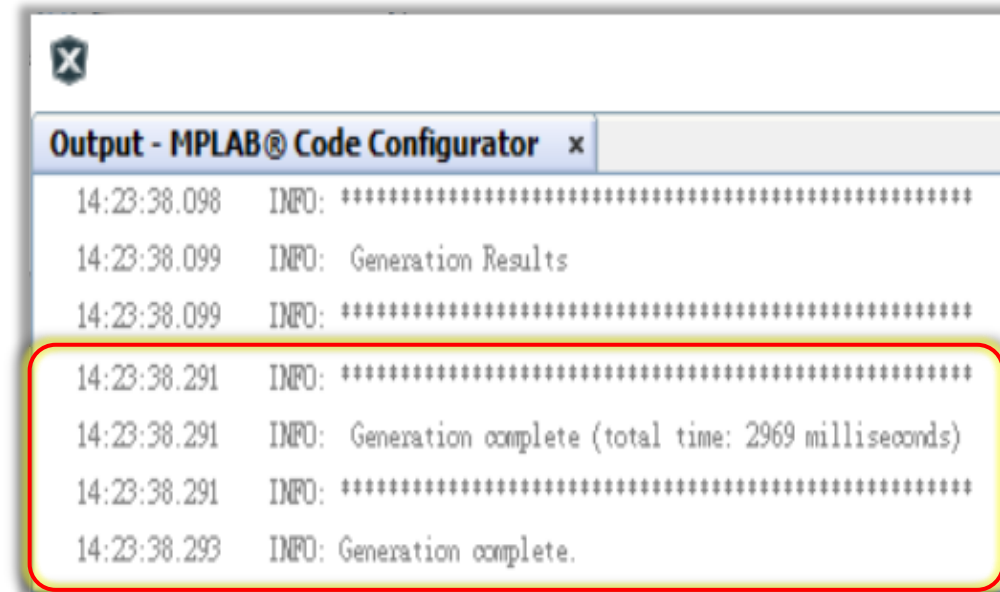
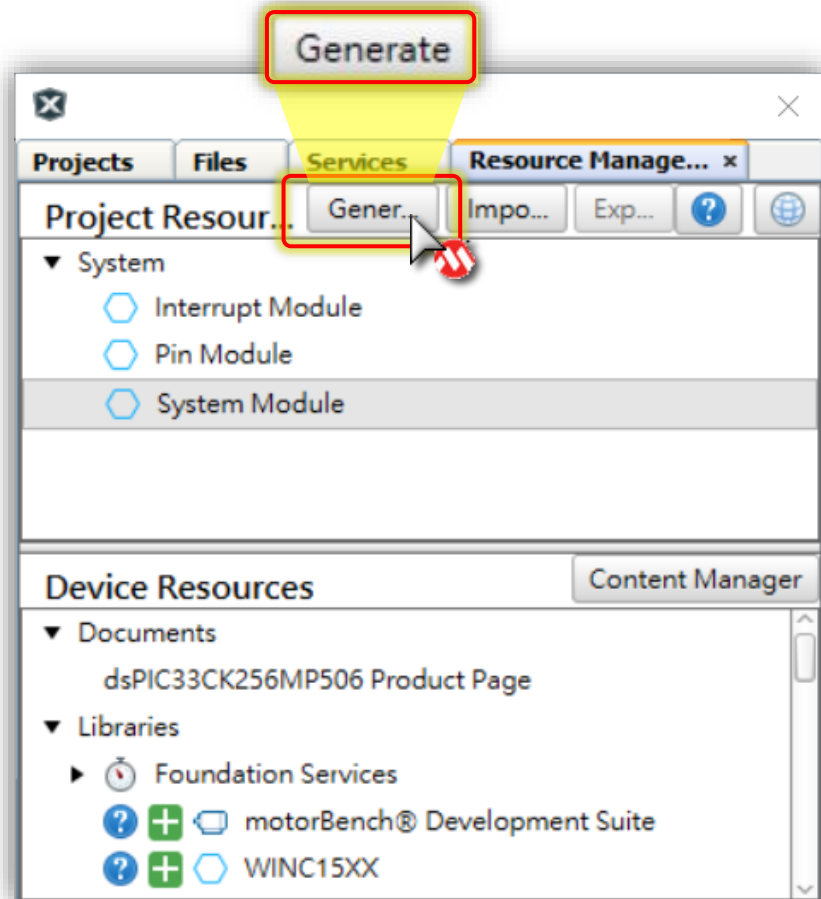
- Serial Protocol.
  - SPI ► I2C



# Lab10 Temperature Sensor

## Step 3

- Generate Code.



# Lab10 Temperature Sensor

## Code Example

```
#include "mcc_generated_files/mcc.h"
#include "mcc_generated_files/tmr2.h"
#include "mcc_generated_files/examples/i2c_master_example.h"
#include <stdio.h>

#define MCP9800_Address 0x4D
#define MCP9800_REG_TA 0x01
#define MCP9800_Register 0x60
#define MCP9800_Start 0x00

...
volatile uint8_t TMRFlag;
uint16_t MCP9800_Read_Data;

int main(void)
{
    ...
    while (1)
    {
        ...
    }
}

...

void LCD_Show(uint16_t LCD_Value)
{
    ...
}
```

# Lab10 Temperature Sensor

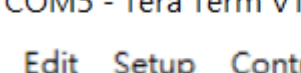
## Code Example

```
int main(void)
{
    ...
    TMR2_SetInterruptHandler(TIM2_ISR);
    I2C_Write1ByteRegister(MCP9800_Address, MCP9800_REG_TA, MCP9800_Register);
    __delay_us(100);
    I2C_Write1ByteRegister(MCP9800_Address, MCP9800_Start, 0x00);
    __delay_us(100);

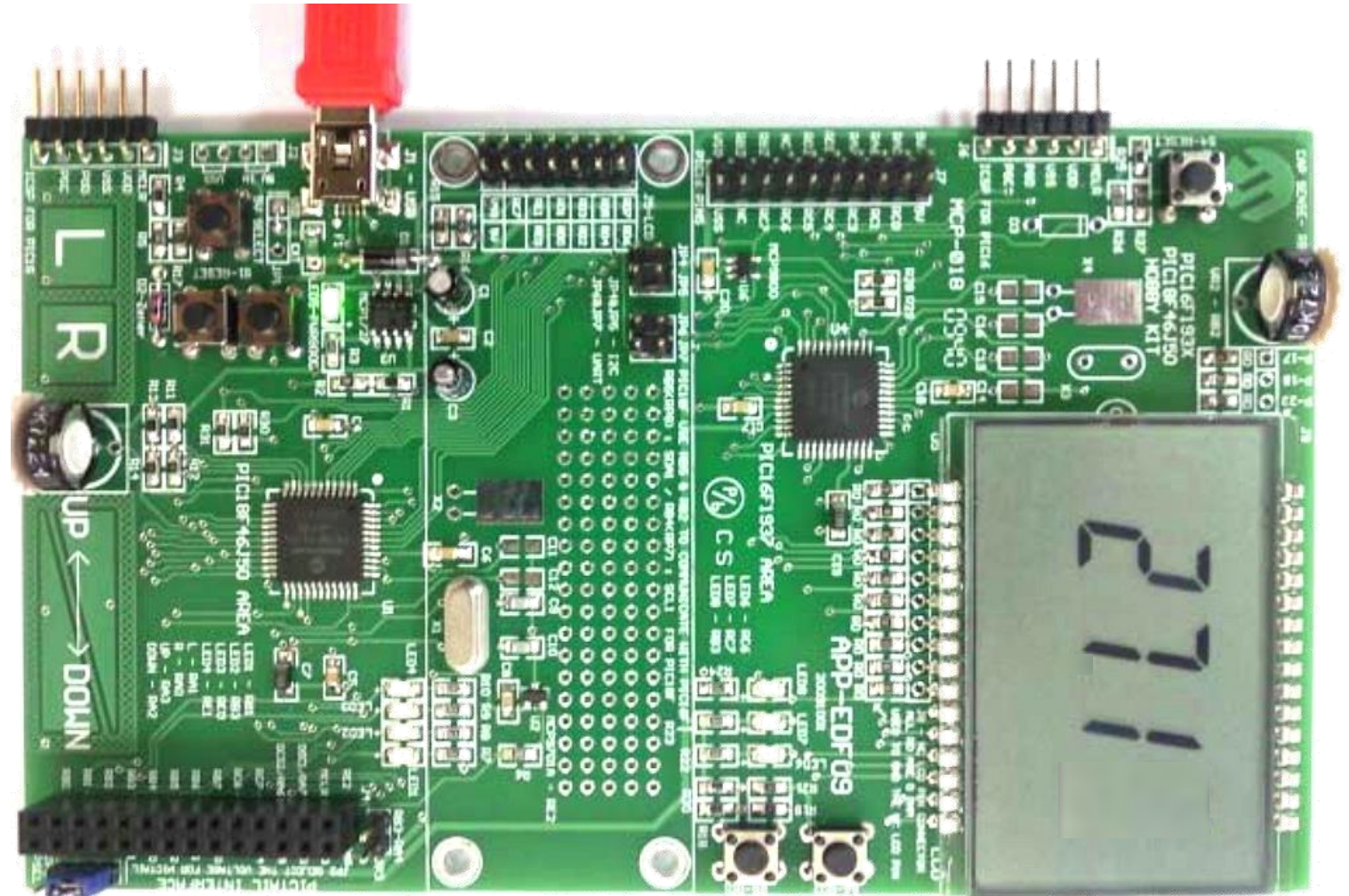
    ...
    while (1)
    {
        // Add your application code

        if (TMRFlag)
        {
            TMRFlag = 0;
            LED1_Toggle();
            ADC_Value = ADC_GetConversion(VR);
            LCD_Show(ADC_Value);
            MCP9800_Read_Data = I2C_Read2ByteRegister (MCP9800_Address, NULL);
            printf(" \033[1;1HADC_Value:%4d\r\n", ADC_Value);
            printf(" \033[2;1HTemp_9800:%2d.%3d'C", (int) ((MCP9800_Read_Data >> 8)&0x7F), (int) ((MCP9800_Read_Data&0xFF) >> 4)*1000 / 16);
        }
    }
}
```

## Result



The screenshot shows a terminal window titled "COM5 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The main display area shows two lines of text: "ADC\_Value: 271" and "Temp\_9800:26.812°C".



# 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](mailto: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

