

MCC TRAINING 2014



Technical training for embedded control engineers

18004 MCC

PIC24 LAB Manual

Tools Used In This Class

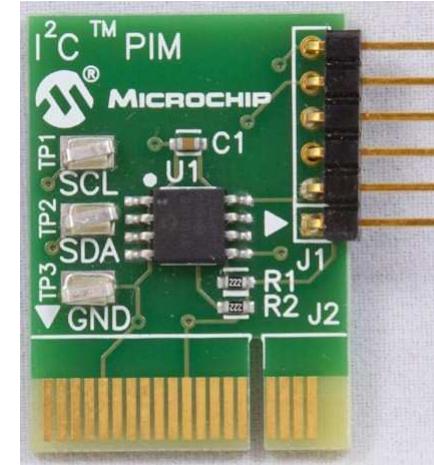
- **Explorer 16 Board**
 - Part Number:
DM240001
- **PIC24FJ128GA310
(MA240029)**
- **MCP2200 USB to RS232
Demo Board and cable**
- **Speech Playback
PiCtail™ Plus Daughter
Board (AC164125)**



Tools Used In This Class

(continued)

- **I²C EEPROM**
 - Part number
AC243003



- **MPLAB® REAL ICE™**
In-circuit emulator
 - Part number
DV244005



Installing the MCC Plugin

- The MPLAB® X IDE Code Configurator Plugin should already be installed for this MASTERs class
- To install the MCC Plugin refer to the Appendix at the end of this manual for Instructions

PIC24 Lab 1: Timer based LED Toggle

PIC24 Lab 1 Objectives

- **Start up MCC in MPLAB® X IDE**
- **Setup the configuration registers and oscillator**
- **Generate code to toggle an LED based on the Timer module.**

PIC24 Lab 1 Overview

- **Connect the hardware**
- **Open MPLAB® X IDE**
- **Open project named ‘lab1_16bit’**
 - C:\MASTERs\18004\labs\lab1\lab1_16bit.X
- **Open MPLAB® Code Configurator**
- **Configure RA7 pin as GPIO**
- **Configure TMR1 period for 1s with Prescaler 1:64**

PIC24 Lab 1

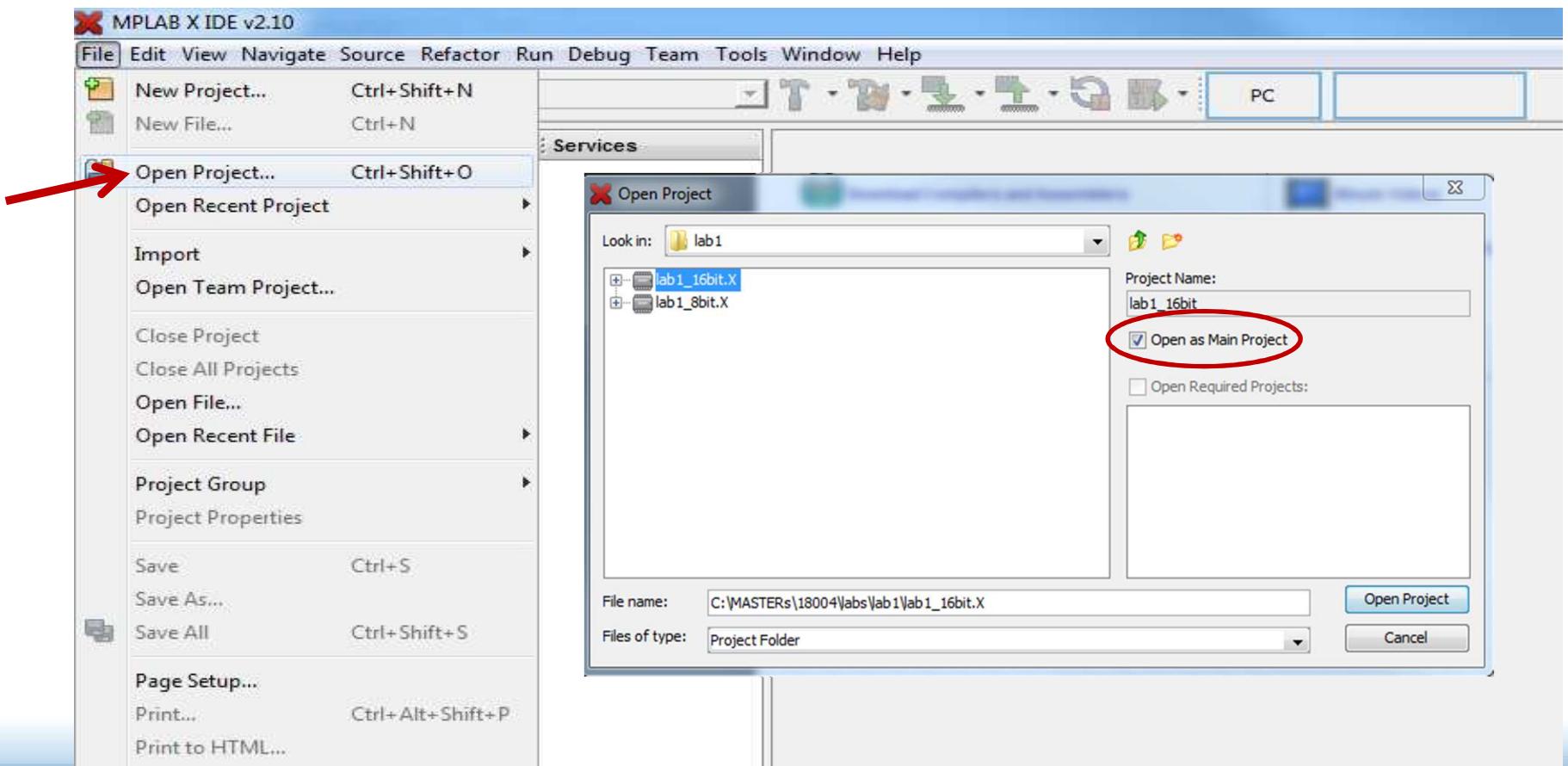
Connecting the hardware

- **Connect the power supply to the Explorer 16 Board**
- **Connect the MPLAB® REAL ICE™ to the Explorer 16 Board**
- **Connect the USB cable between the MPLAB® REAL ICE™ and the Computer**

PIC24 Lab 1

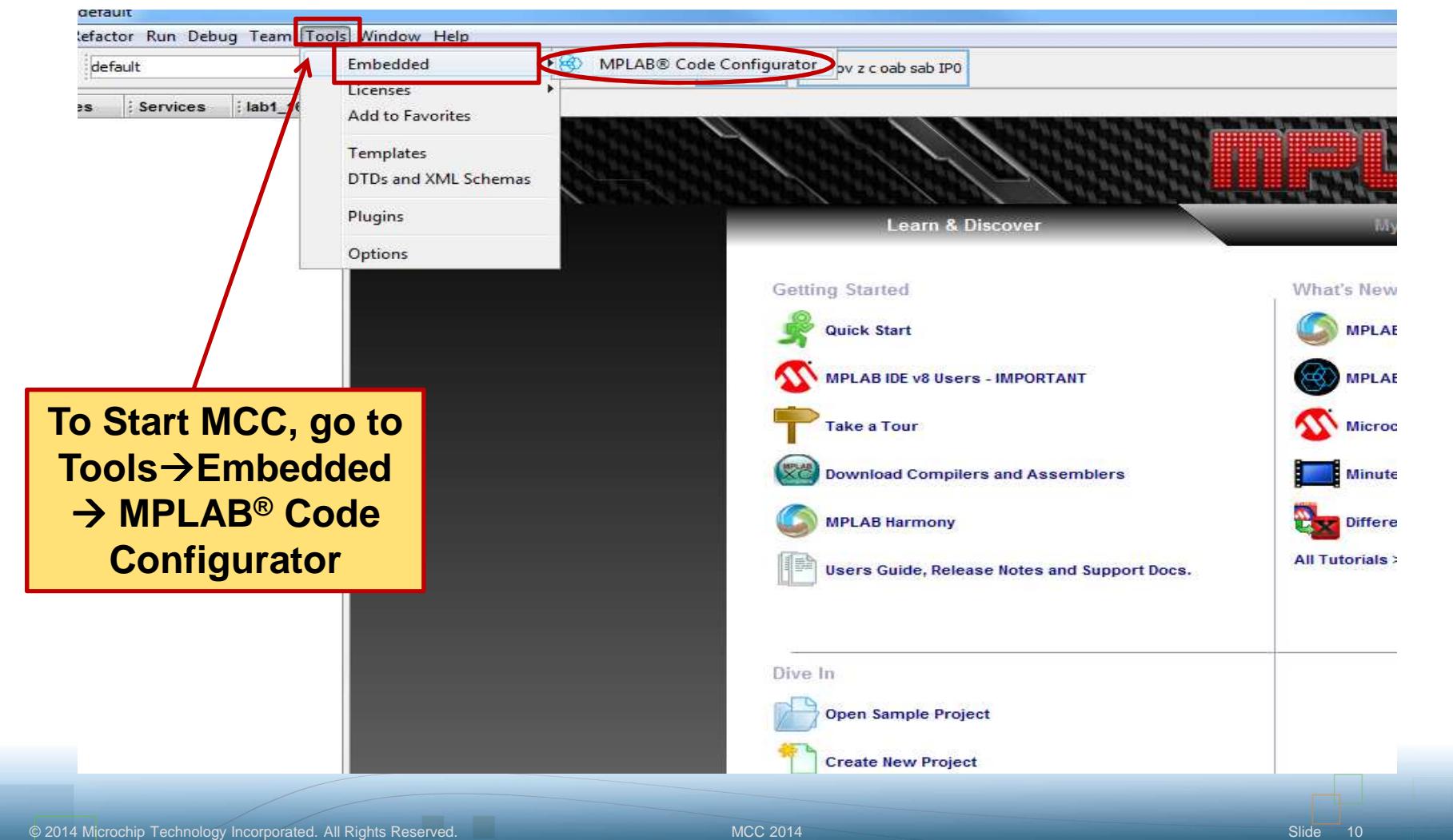
Setting up the Project

- In MPLAB® X IDE, click on Open Project and select lab1_16bit.X



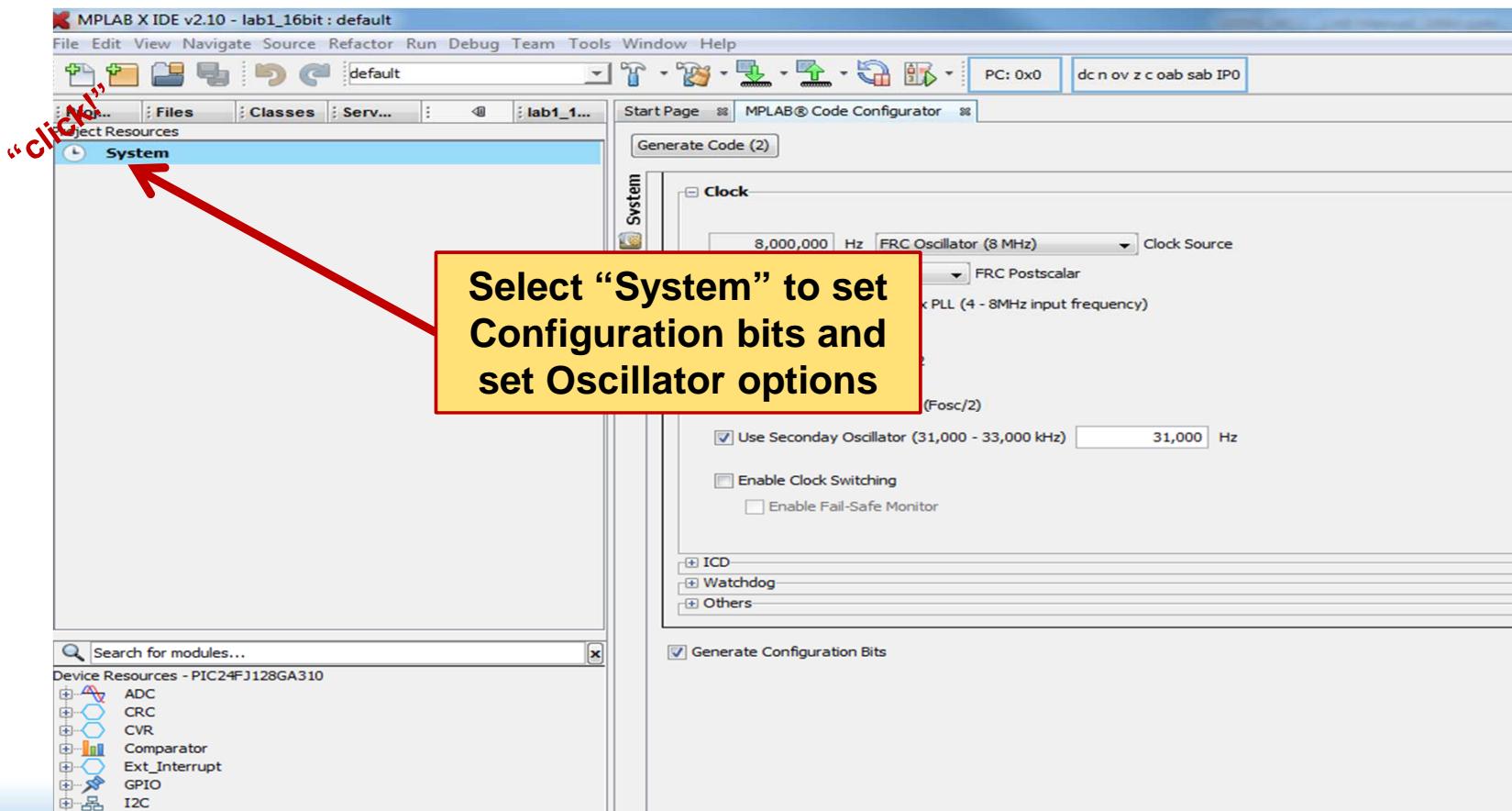
PIC24 Lab 1

● Start MPLAB® Code Configurator



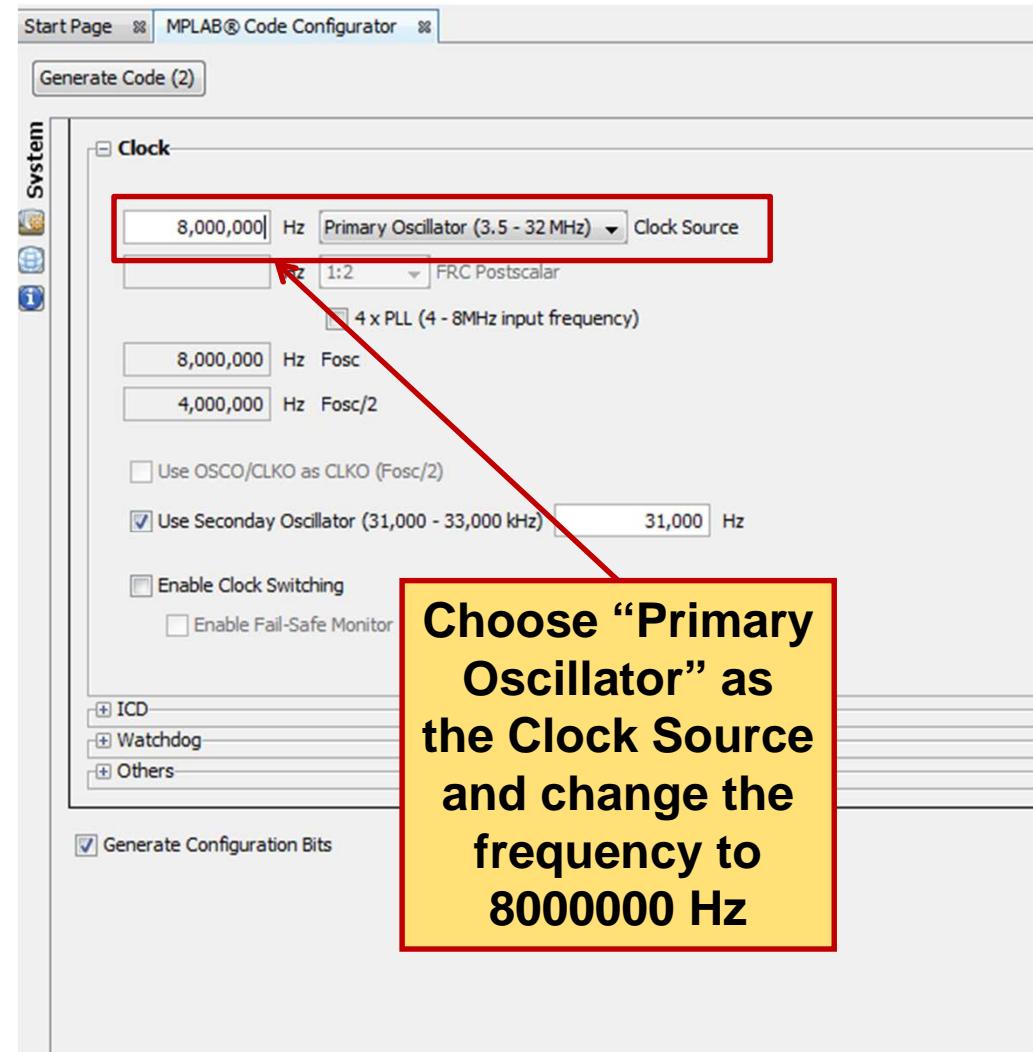
PIC24 Lab 1

- Select System to setup Configuration bits and the Oscillator



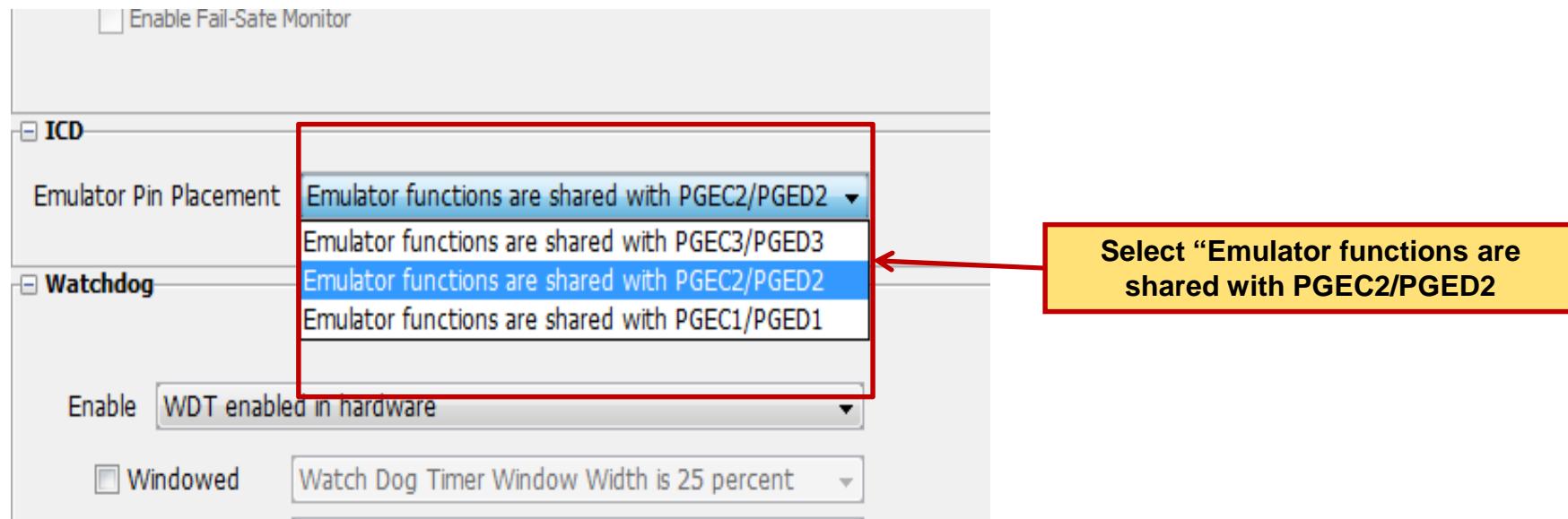
PIC24 Lab 1

- Configuration bits are preset to allow development and debugging



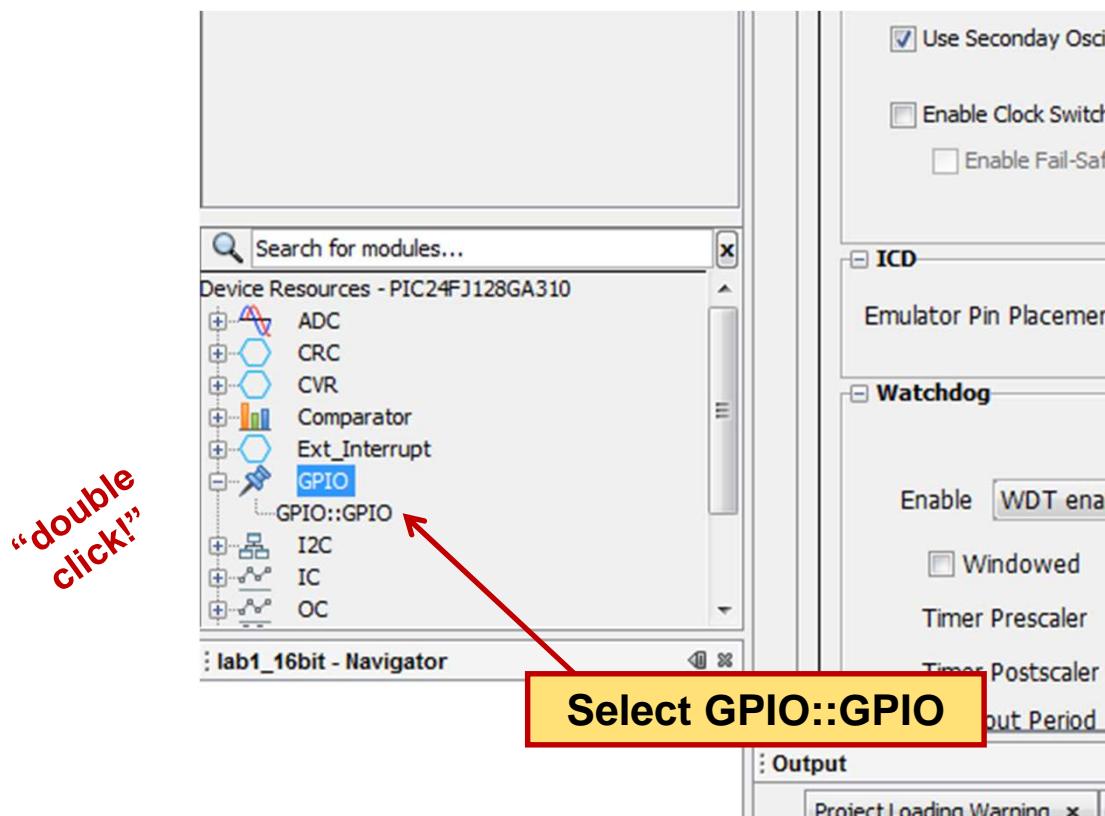
PIC24 Lab 1

- **Change Emulator Pin Placement to PGEC2/PGED2**



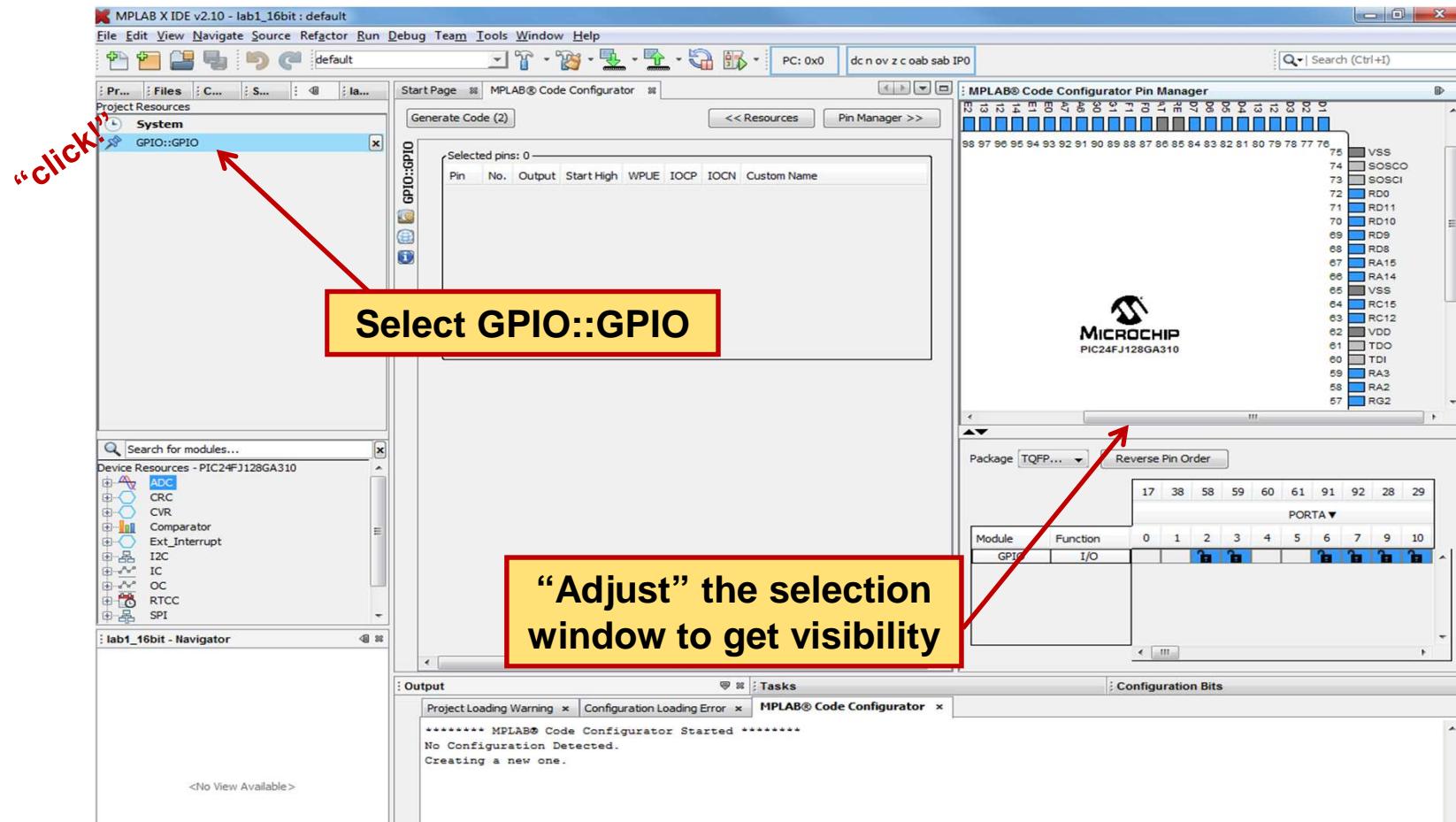
PIC24 Lab 1

- Double click on GPIO in Device Resources and then select GPIO::GPIO



PIC24 Lab 1

- Click on **GPIO::GPIO**



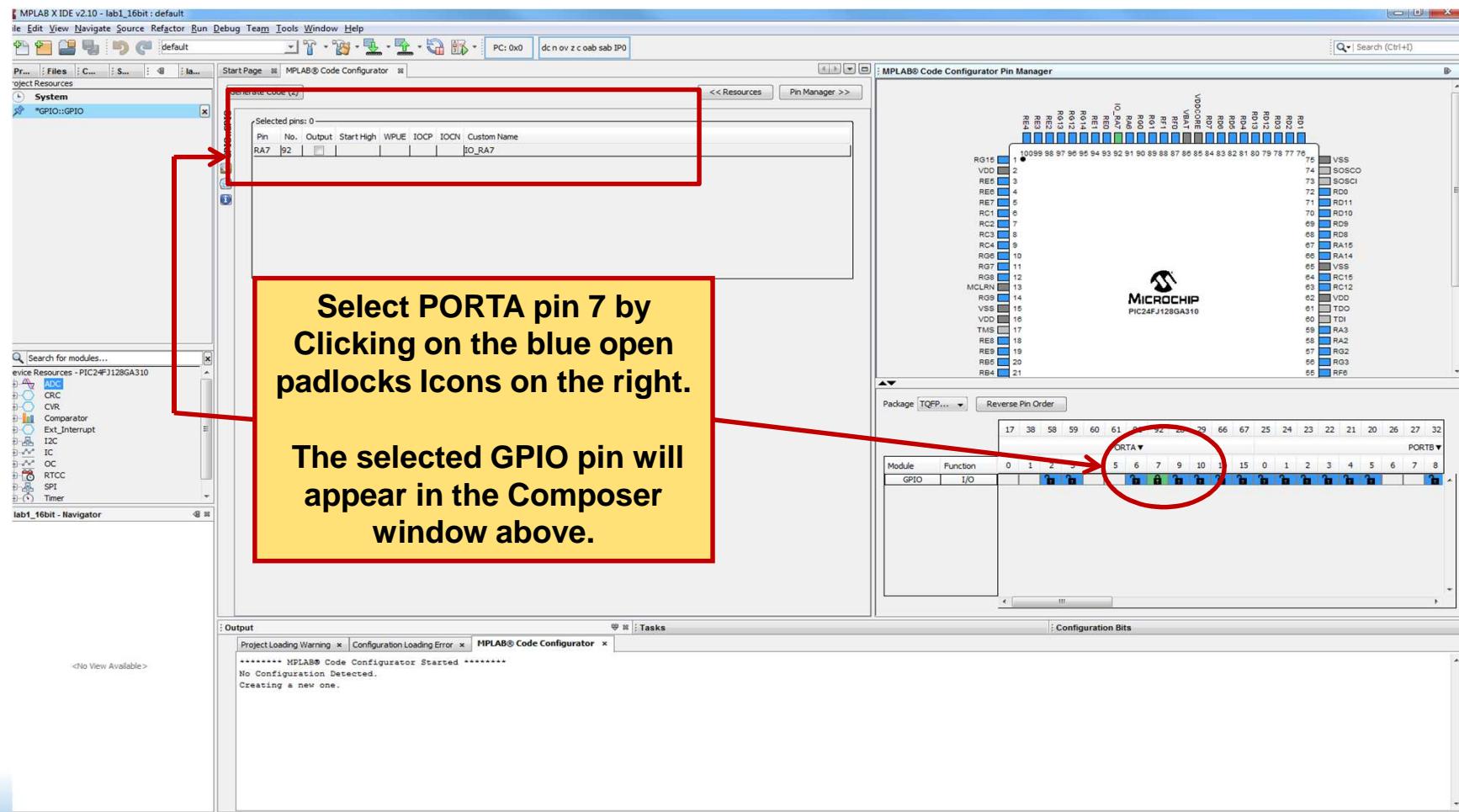
PIC24 Lab 1

- Did you know ...
- that you can configure whether or not MCC generates configuration bits for your project?



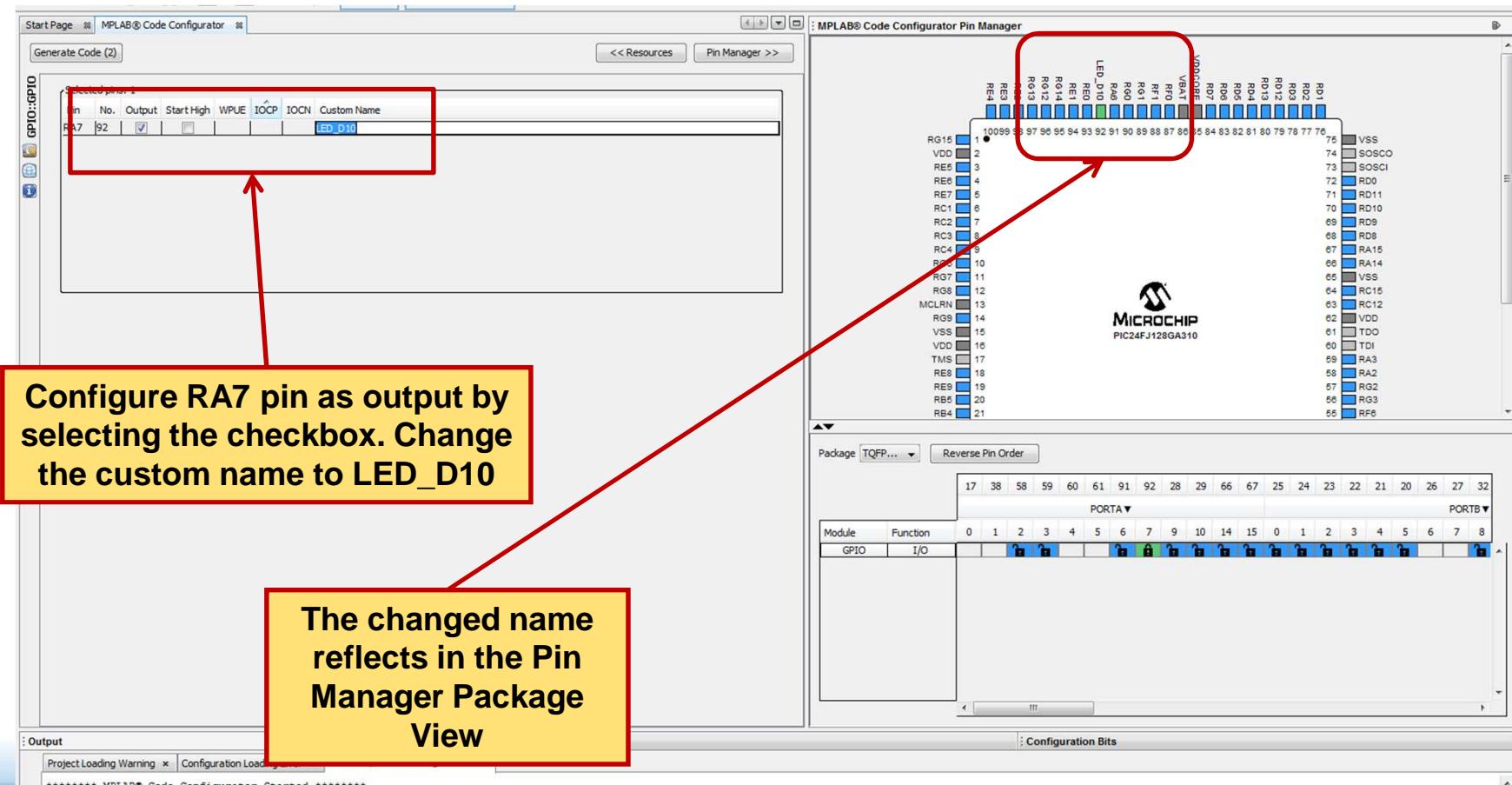
PIC24 Lab 1

- Set up RA7 pin from the Pin Manager



PIC24 Lab 1

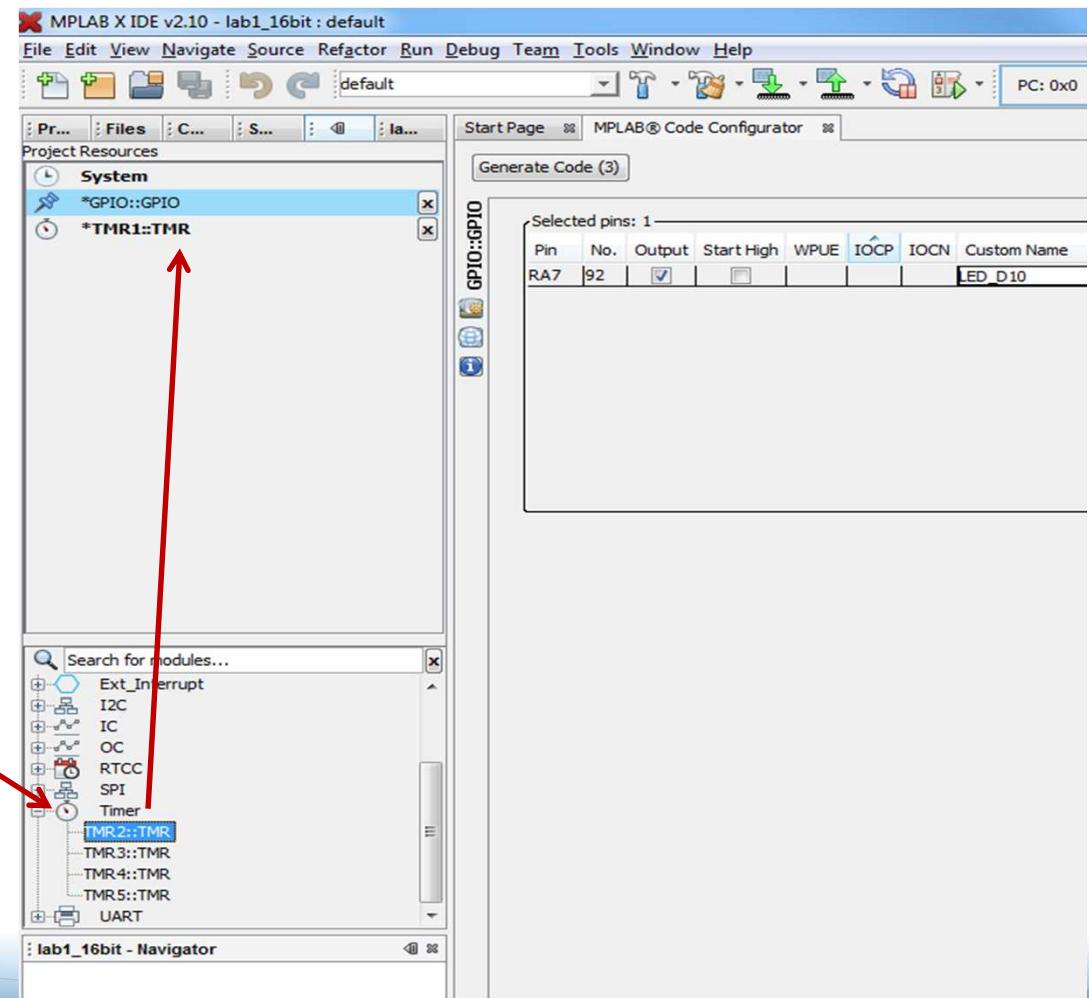
- Set RA7 pin to output and change the name to LED_D10



PIC24 Lab 1

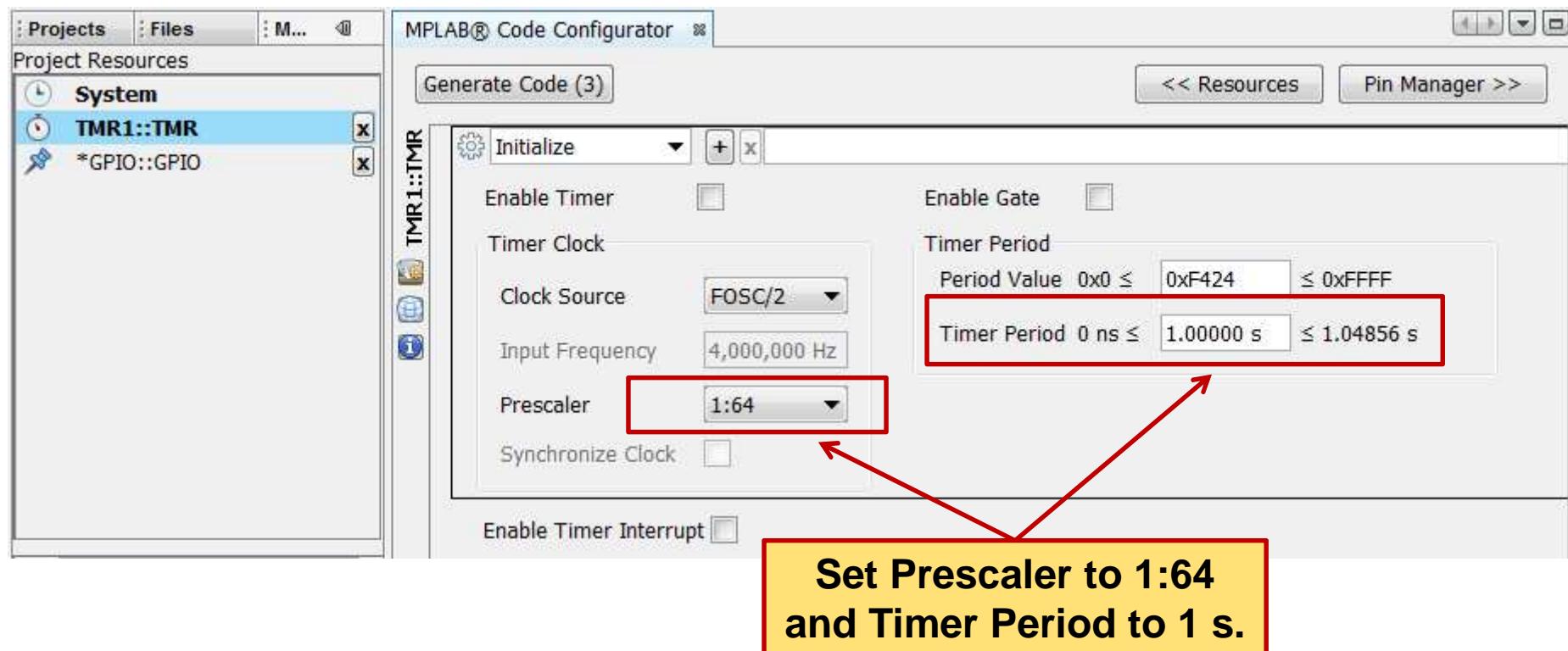
- Select TMR1 module

Double click on
“Timer” to see all
instances of the
module. Select
TMR1::TMR to see it in
the Project Resources



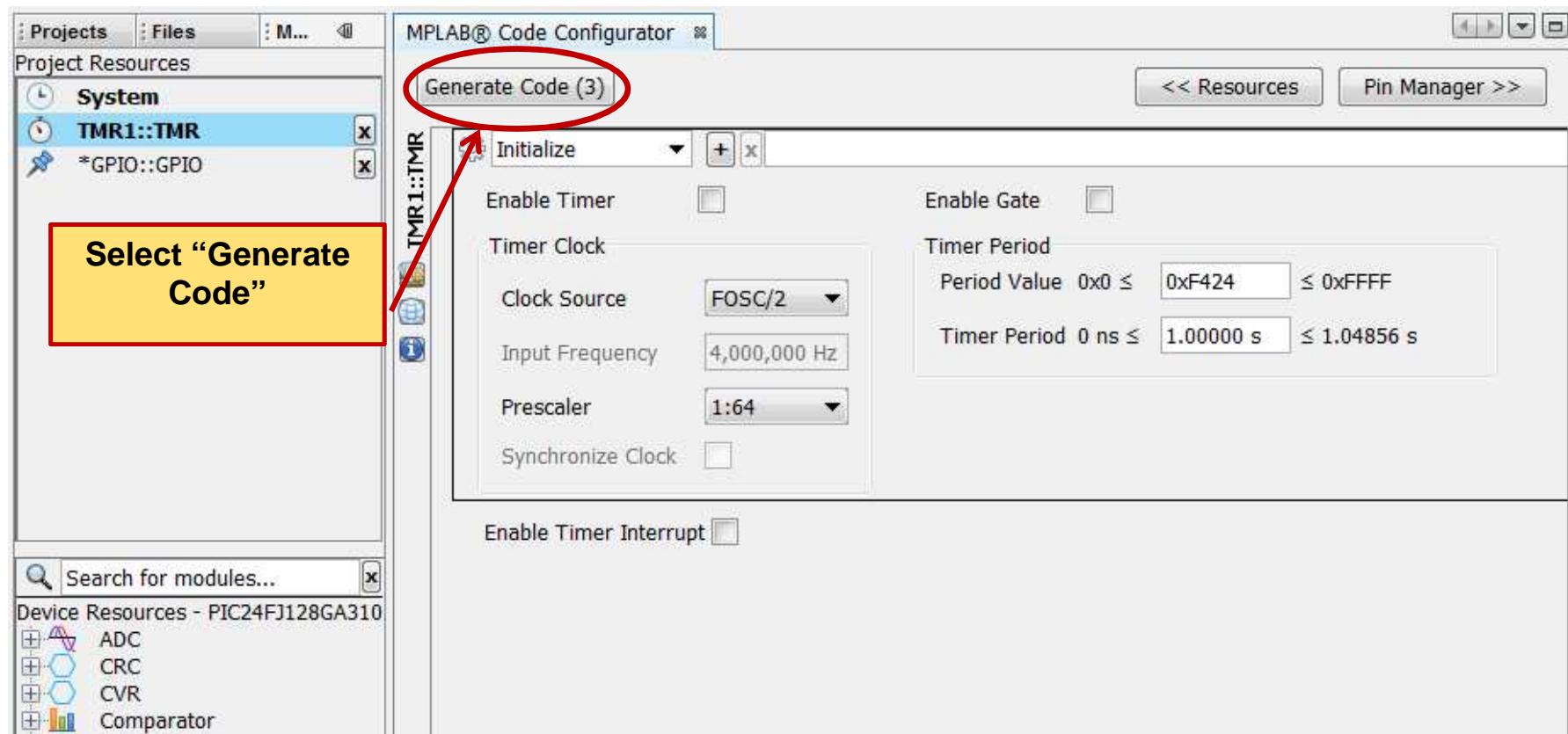
PIC24 Lab 1

- Select TMR1 module



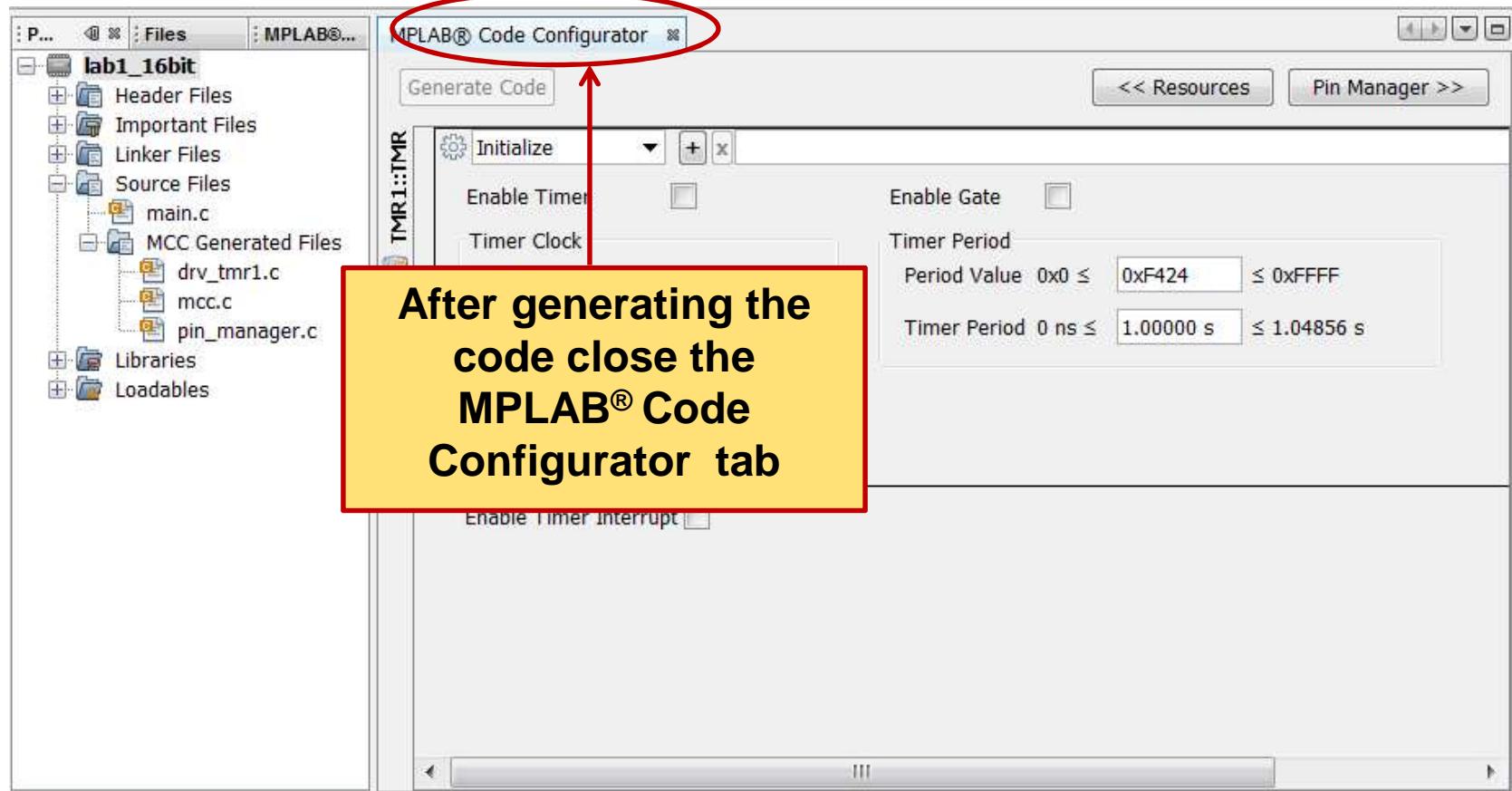
PIC24 Lab 1

- Generate the code



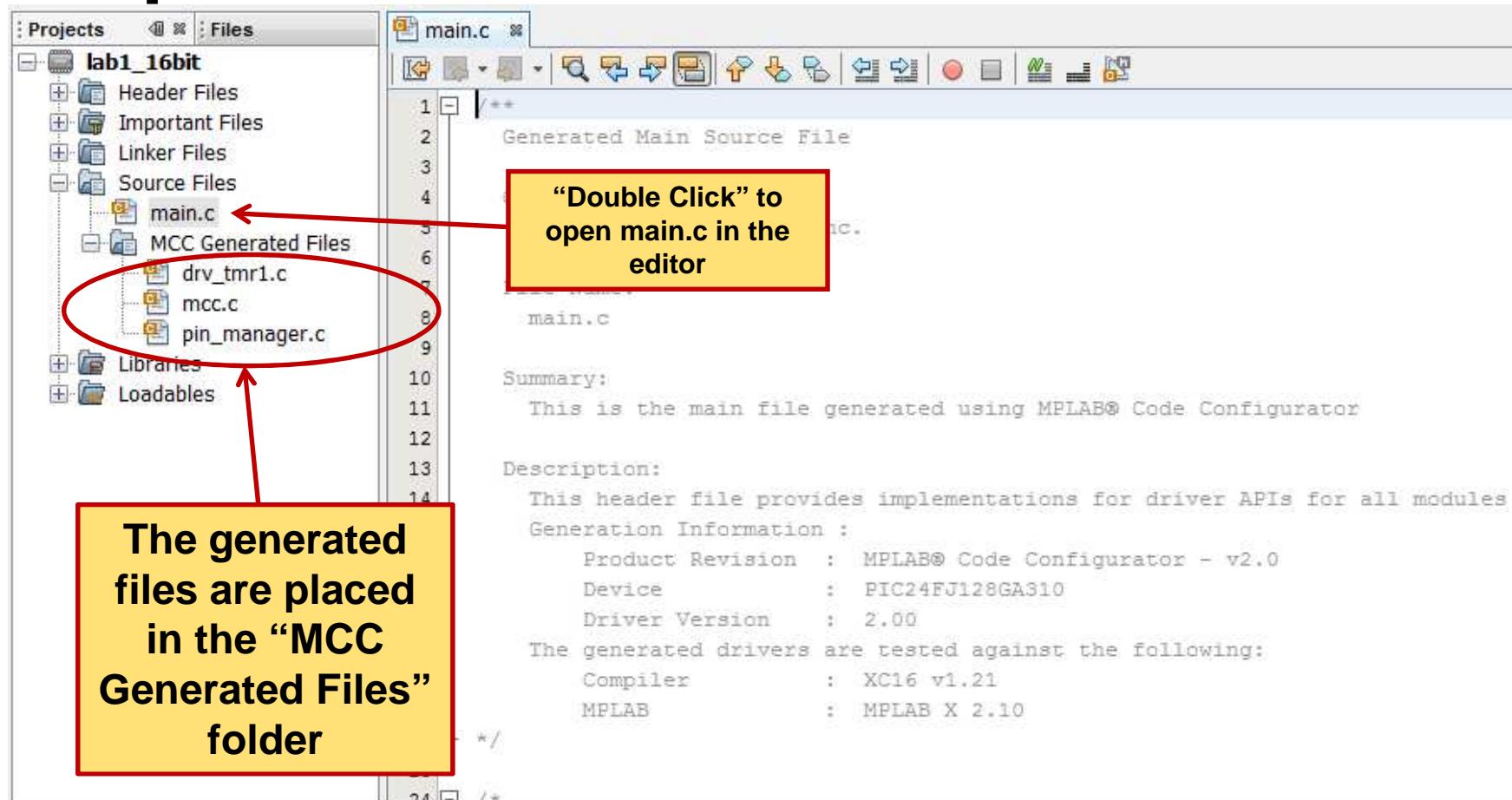
PIC24 Lab 1

- Close MPLAB® Code Configurator



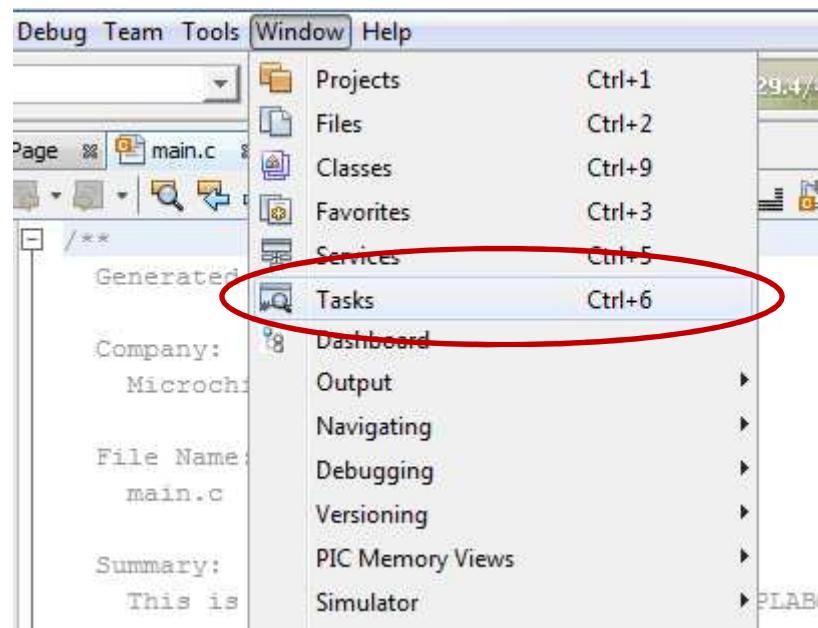
PIC24 Lab 1

- Open main.c



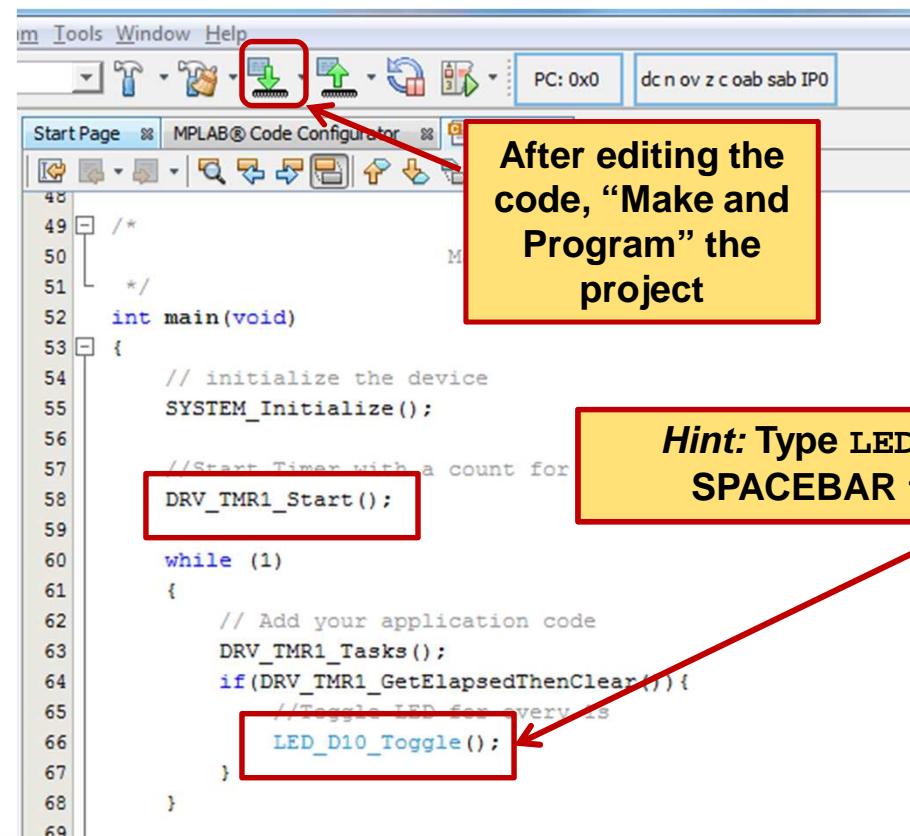
PIC24 Lab 1

- Open the MPLAB X Tasks window to see a list of TODOs for Lab 1



PIC24 Lab 1

- Add the line of code to turn on LED1 then “Make and Program” the project



The screenshot shows the MPLAB X IDE interface. The code editor displays the following C code:

```
48  /* 
49  */
50  */
51  int main(void)
52  {
53  // initialize the device
54  SYSTEM_Initialize();
55
56  //Start Timer with a count for
57  DRV_TMR1_Start();
58
59  while (1)
60  {
61  // Add your application code
62  DRV_TMR1_Tasks();
63  if(DRV_TMR1_GetElapsedThenClear()){
64  //Toggle LED for every 1s
65  LED_D10_Toggle();
66  }
67  }
68
69 }
```

A red box highlights the line `LED_D10_Toggle();`. A yellow callout box with a red border contains the text: "After editing the code, ‘Make and Program’ the project". Another yellow callout box with a red border contains the hint: "Hint: Type LED followed by <CTRL> SPACEBAR to see list of names". A red arrow points from the hint text to the word "LED" in the code.

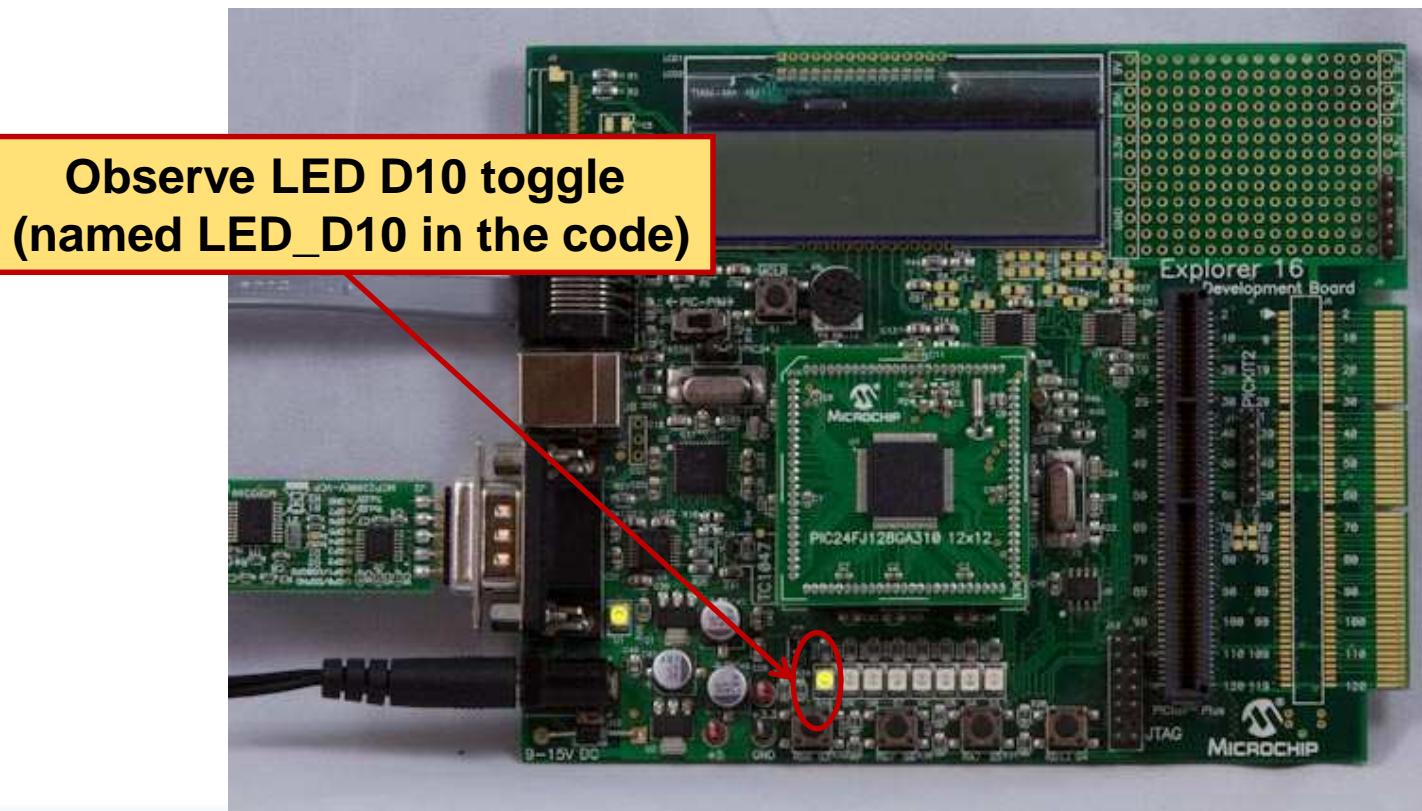
PIC24 Lab 1

- Did you know ...
- that MCC can generate a main.c file for you if one does not exist in the project the 1st time you generate code?



PIC24 Lab 1

- The LED marked D10 should be toggling every second



PIC24 Lab 1 Summary

- We quickly set up GPIO to control an LED
- We quickly set up a timer to blink an LED
- We quickly created an application for toggling an LED

PIC24 Lab 2: Timer based LED Toggle and display ADC value on UART

PIC24 Lab 2 Objectives

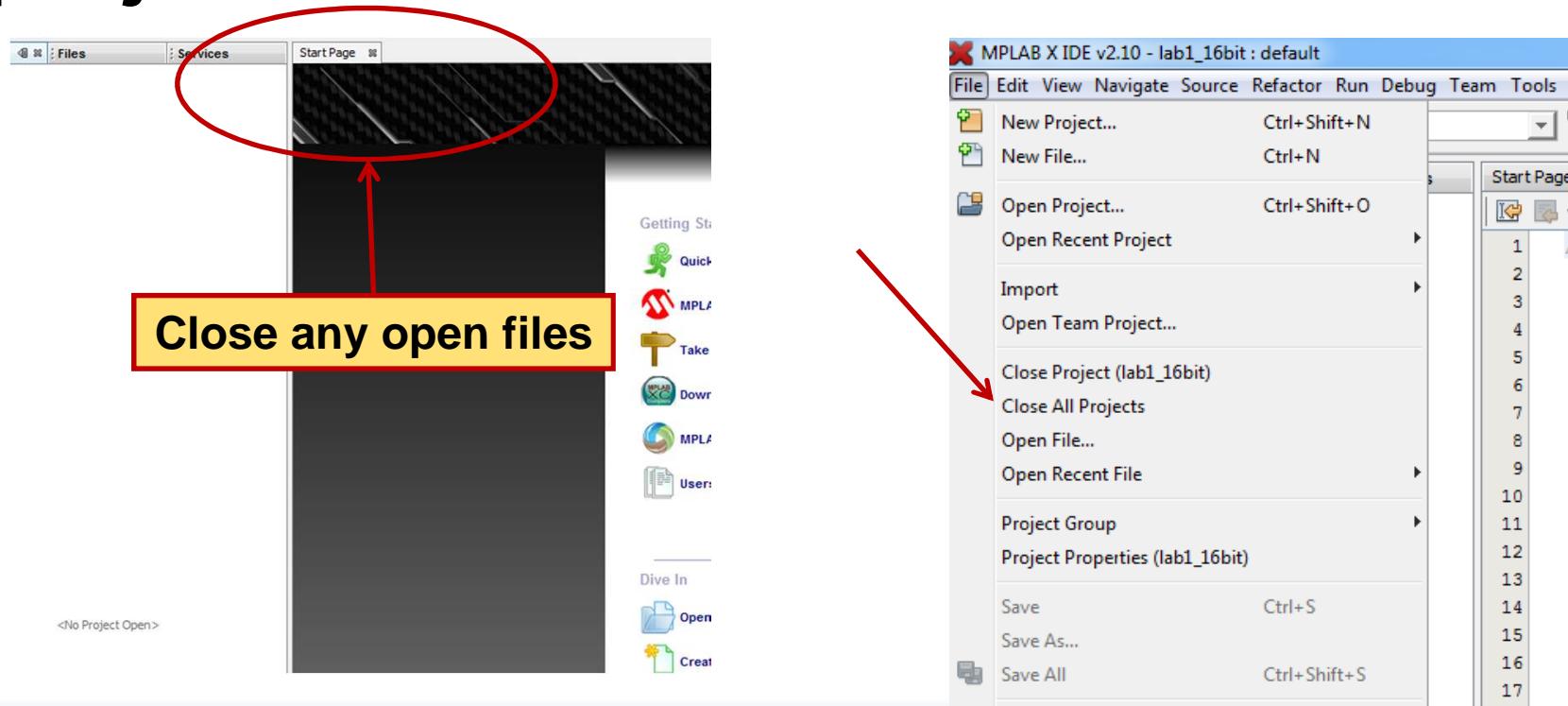
- **Set up the ADC to read the potentiometer voltage**
- **Set up the UART to output potentiometer value.**

PIC24 Lab 2 Overview

- Close any files open in the editor, and close all MPLAB® X IDE Projects
- Open lab2_16bit.X
- Configure ADC1 for Channel AN5 for auto sampling auto conversion mode
- Configure UART2 for transmission with 9600 Baud, Parity None and 1 Stop Bit

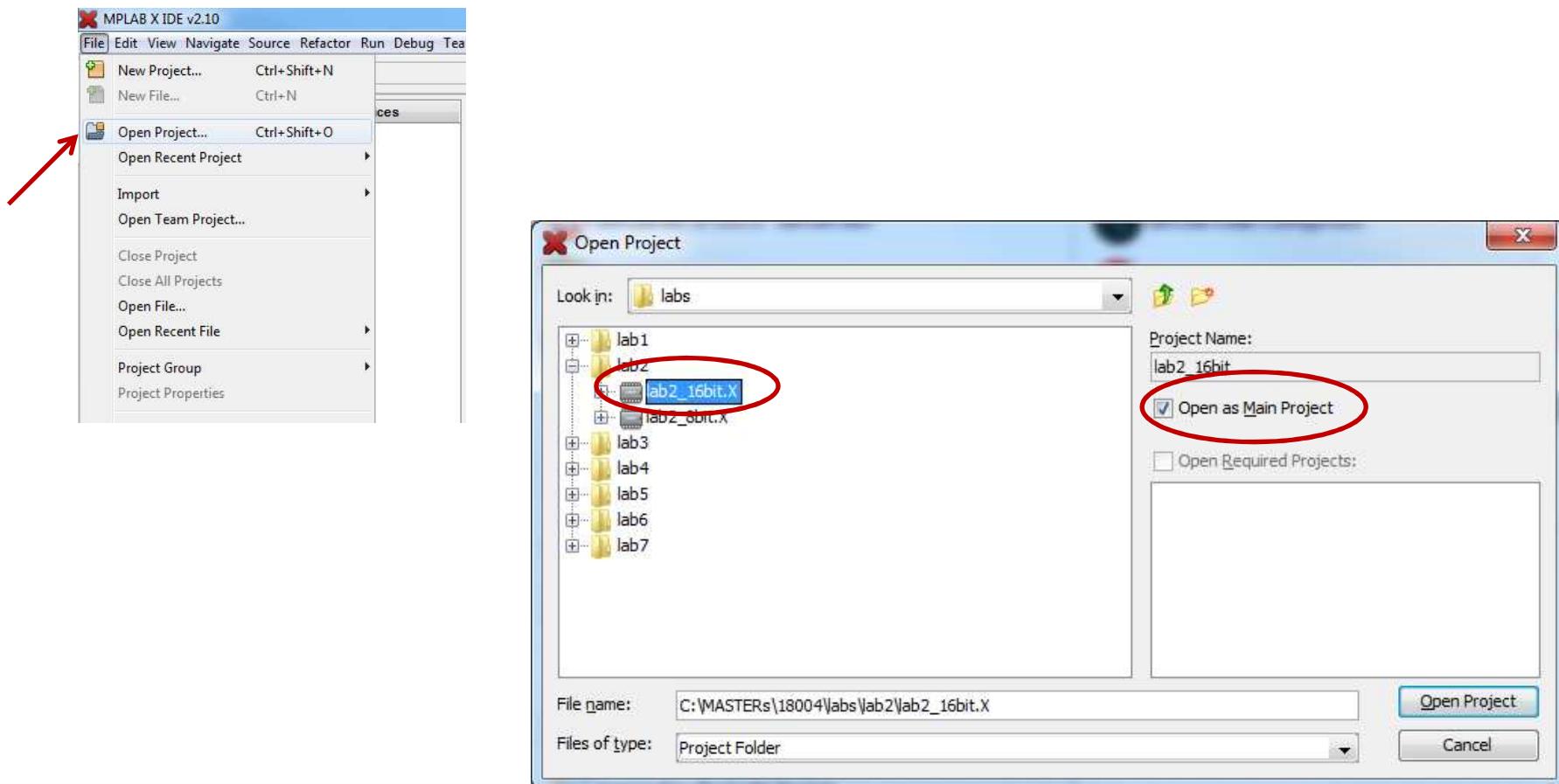
PIC24 Lab 2

- Close any open files in the editor window, and all open MPLAB® X IDE projects



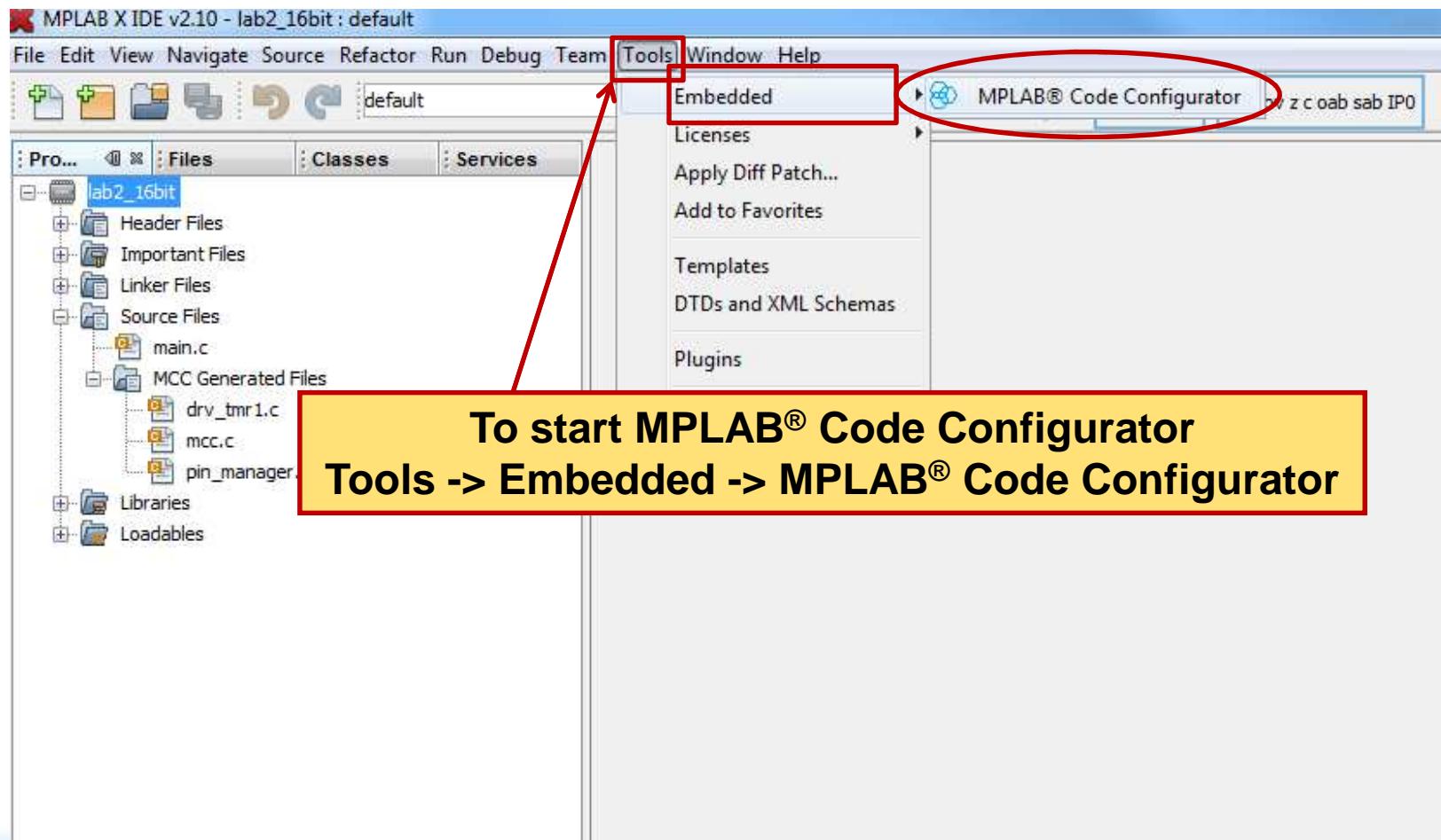
PIC24 Lab 2

- Open lab2_16bit.X



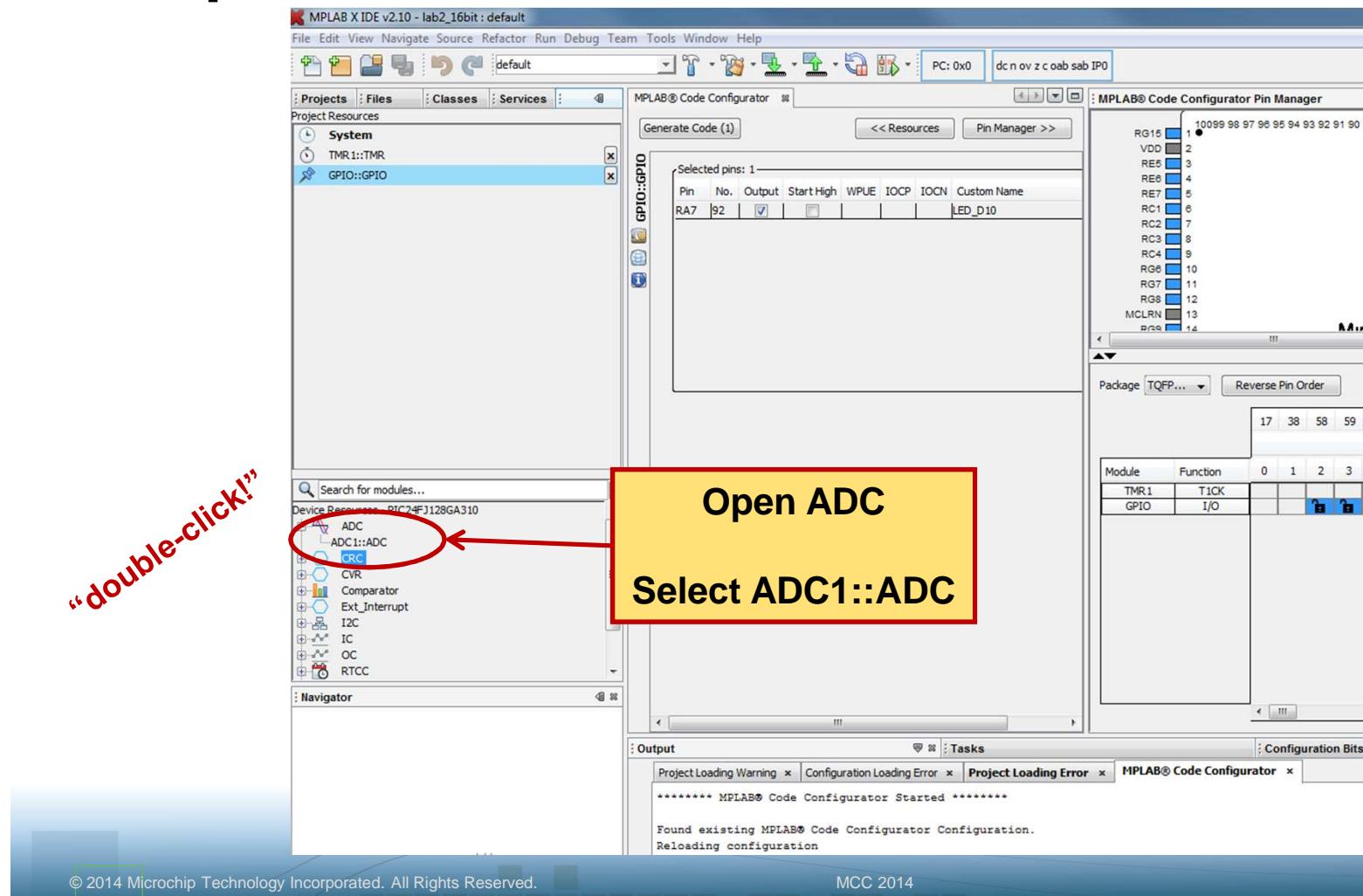
PIC24 Lab 2

- Start MPLAB® Code Configurator



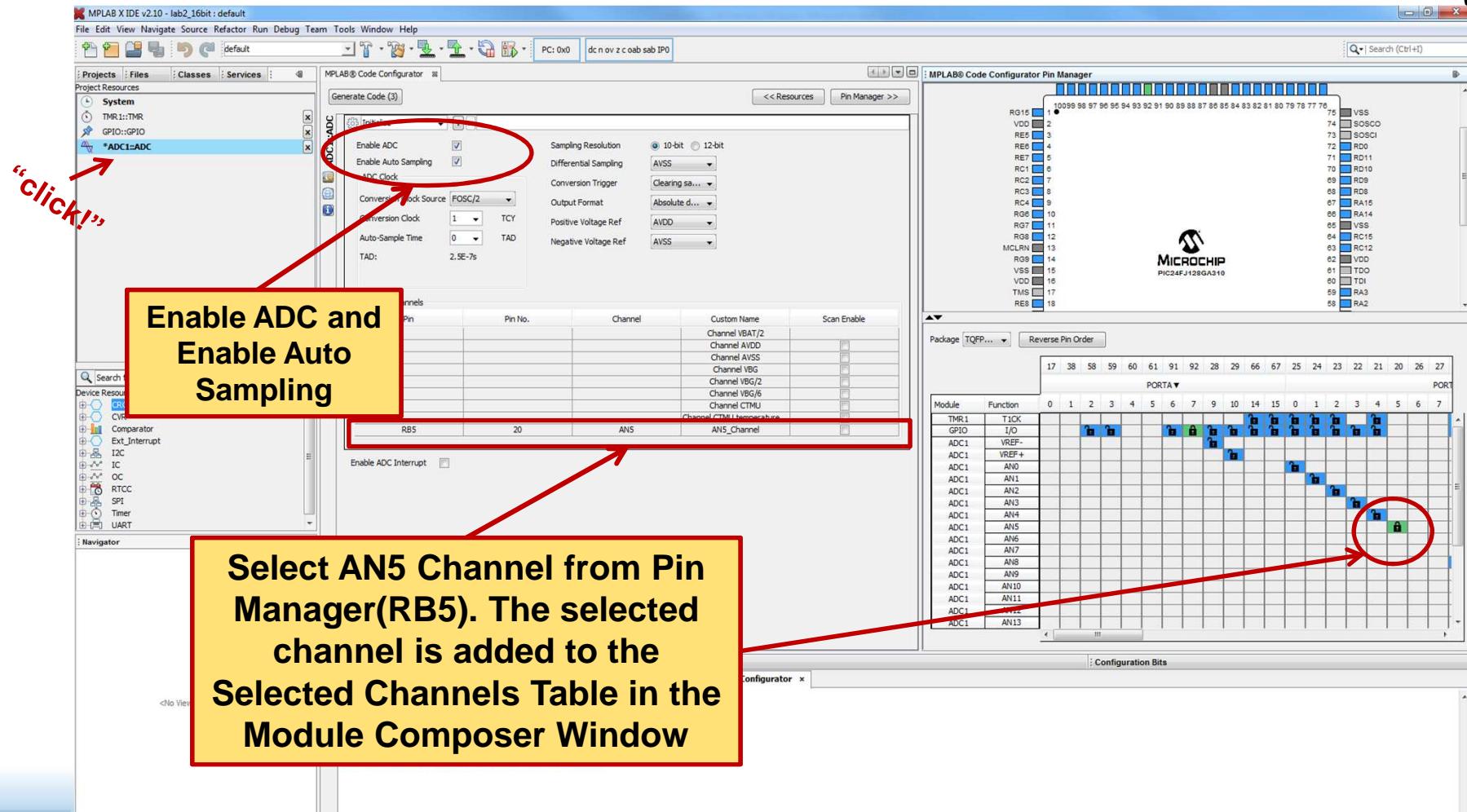
PIC24 Lab 2

- Open the ADC module and select ADC1::ADC



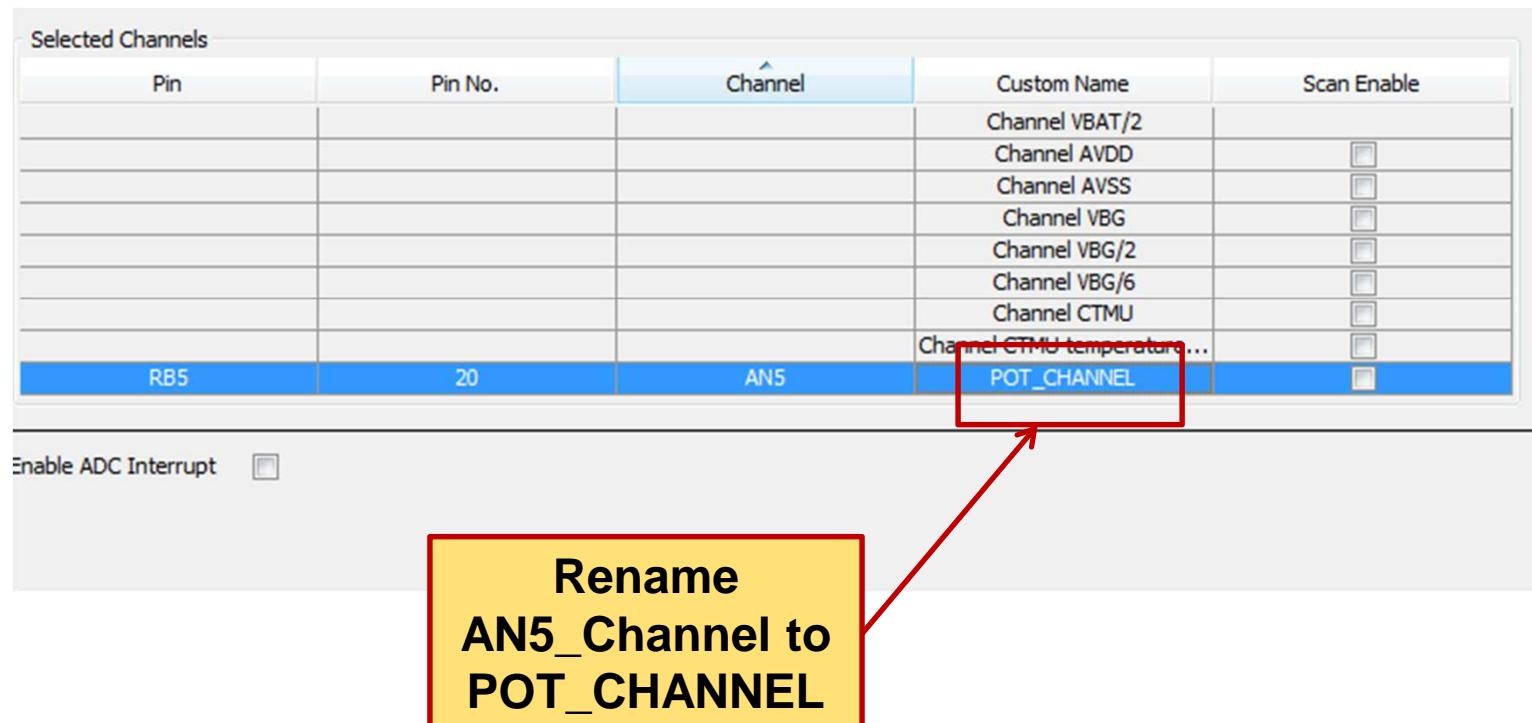
PIC24 Lab 2

- Select ADC1::ADC and do the following



PIC24 Lab 2

- Rename AN5_Channel to POT_CHANNEL



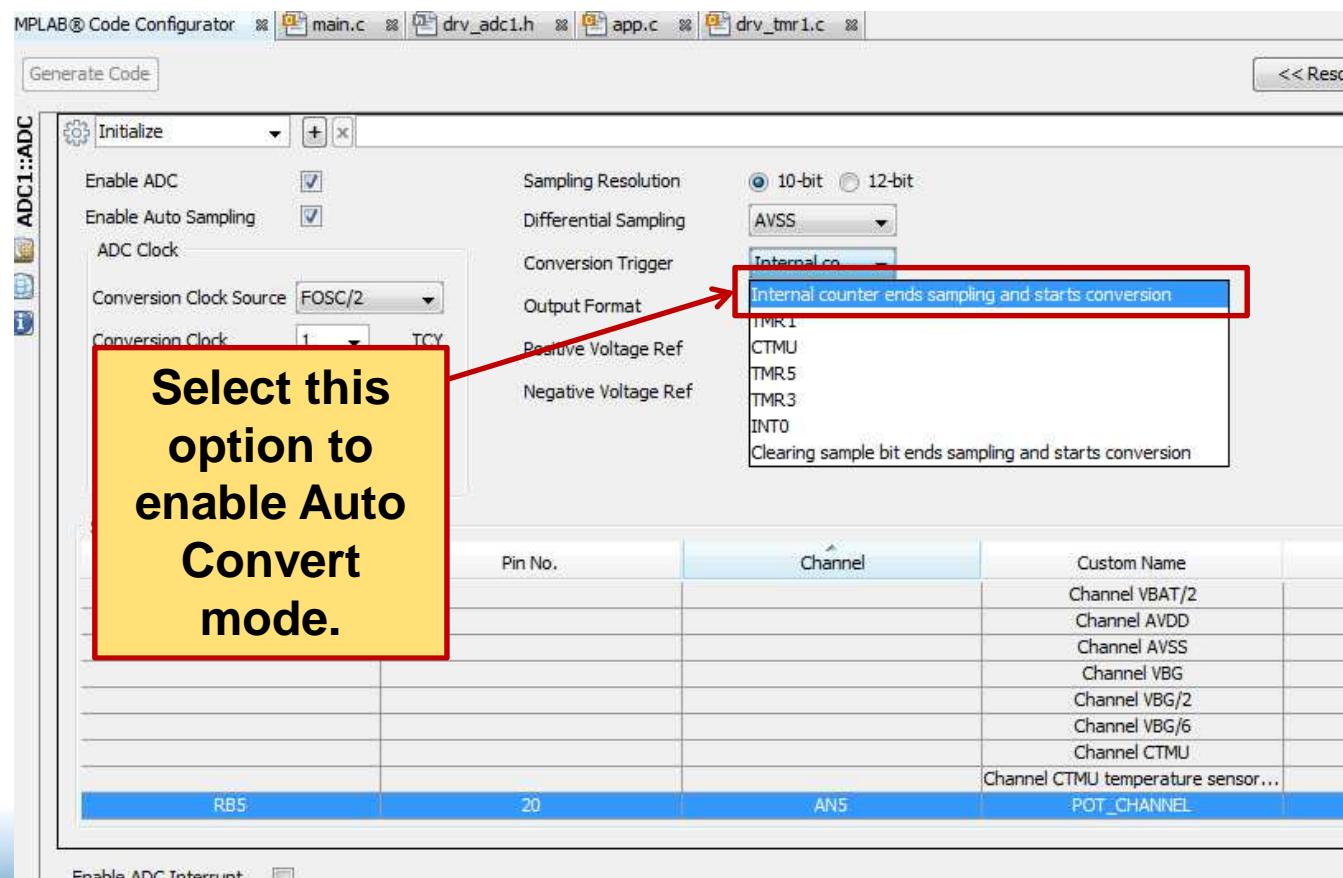
Selected Channels				
Pin	Pin No.	Channel	Custom Name	Scan Enable
			Channel VBAT/2	<input type="checkbox"/>
			Channel AVDD	<input type="checkbox"/>
			Channel AVSS	<input type="checkbox"/>
			Channel VBG	<input type="checkbox"/>
			Channel VBG/2	<input type="checkbox"/>
			Channel VBG/6	<input type="checkbox"/>
			Channel CTMU	<input type="checkbox"/>
			Channel CTMU temperature...	<input type="checkbox"/>
RB5	20	AN5	POT_CHANNEL	<input type="checkbox"/>

Enable ADC Interrupt

**Rename
AN5_Channel to
POT_CHANNEL**

PIC24 Lab 2

- Select Conversion Trigger to “Internal counter ends sampling and starts conversion”

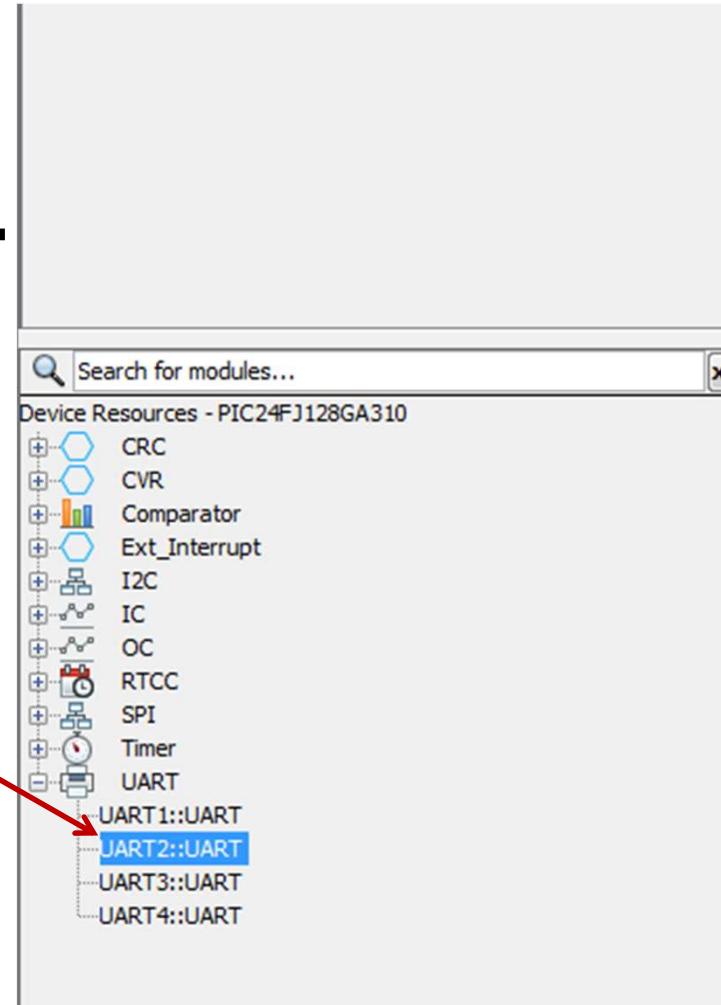


PIC24 Lab 2

- Open UART and select **UART2::UART**

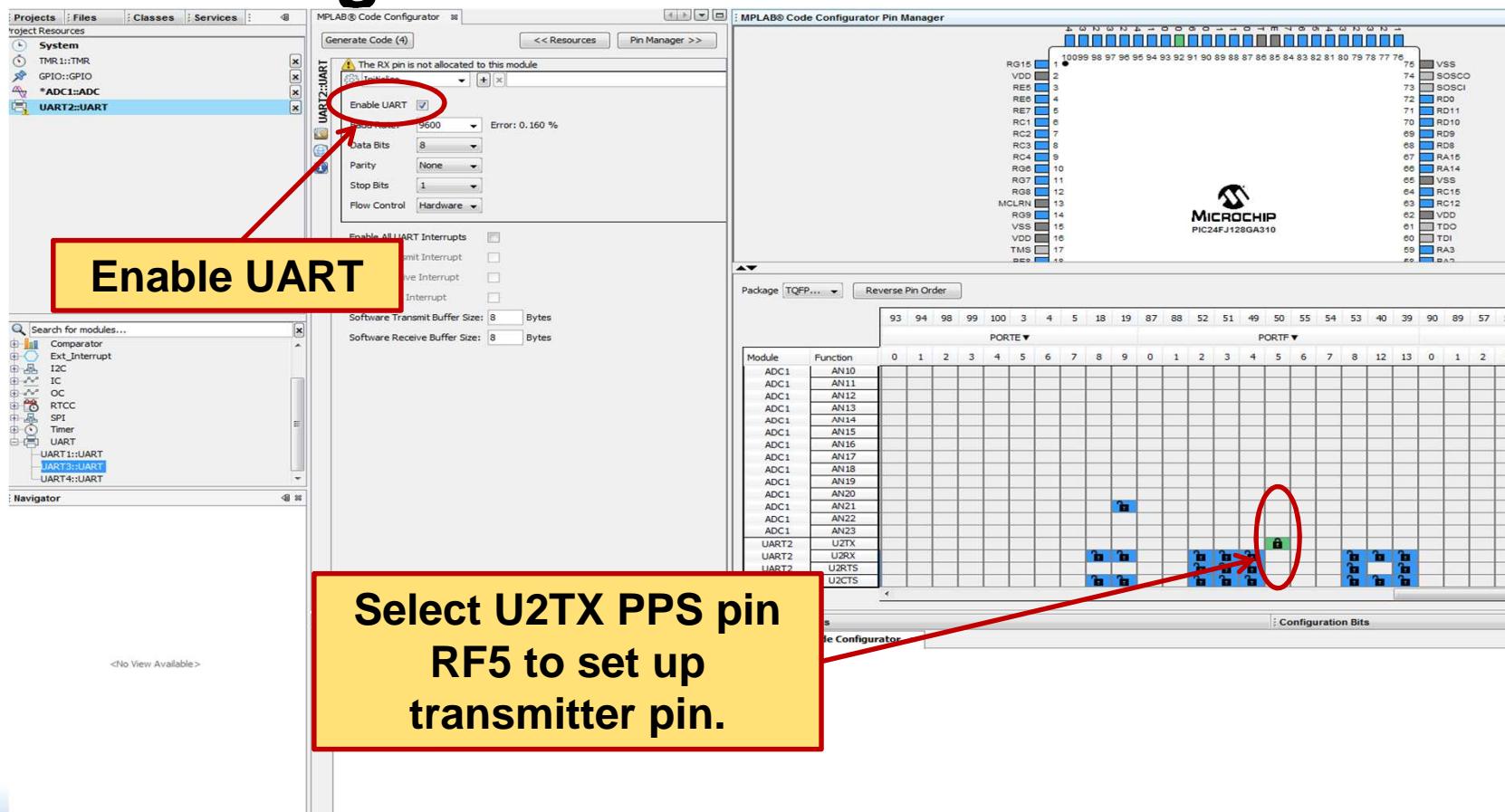
Open UART module in the Device Resources and select **UART2::UART**

“double-click!”



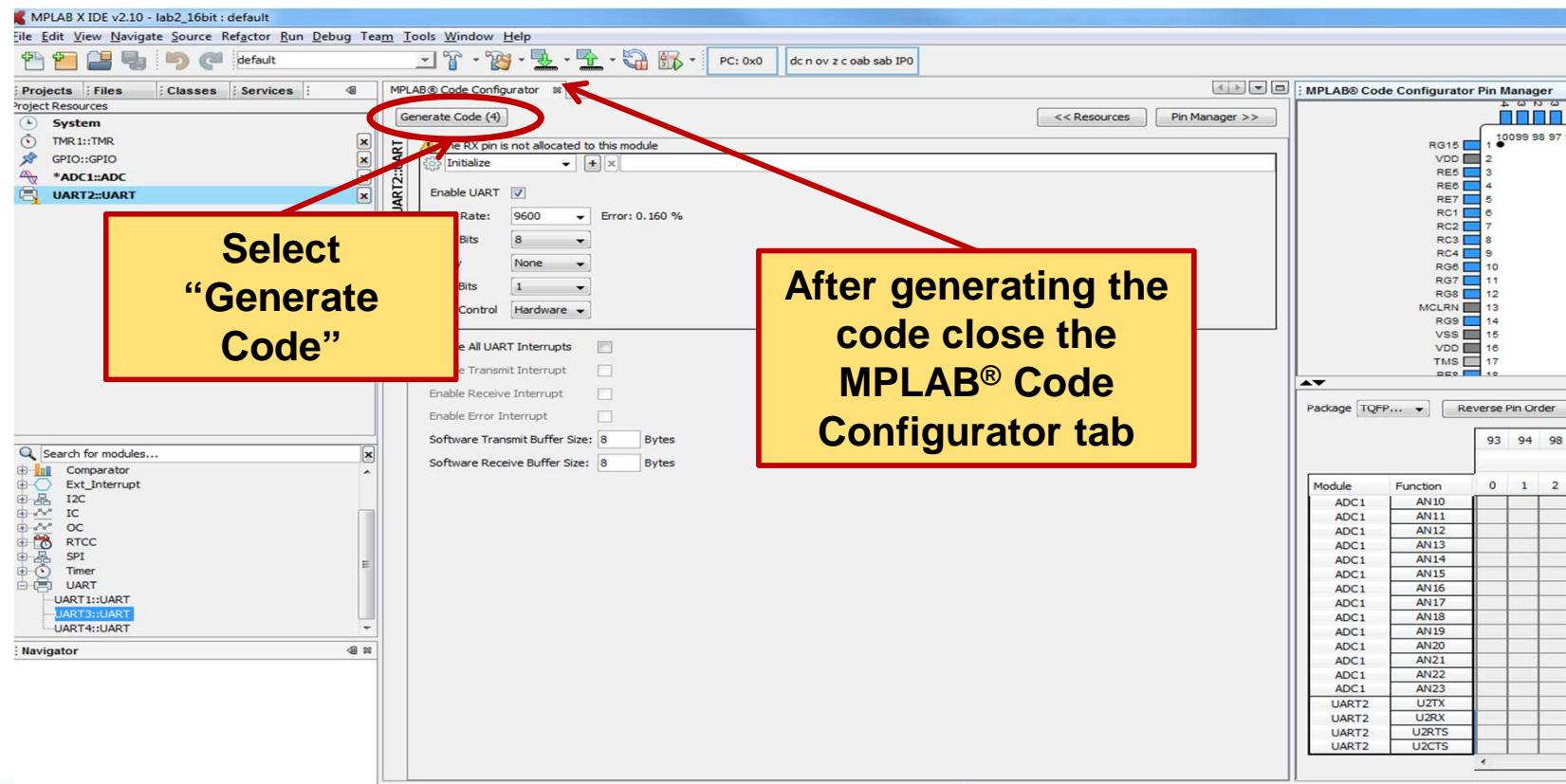
PIC24 Lab 2

- Select UART2::UART and make the following selections



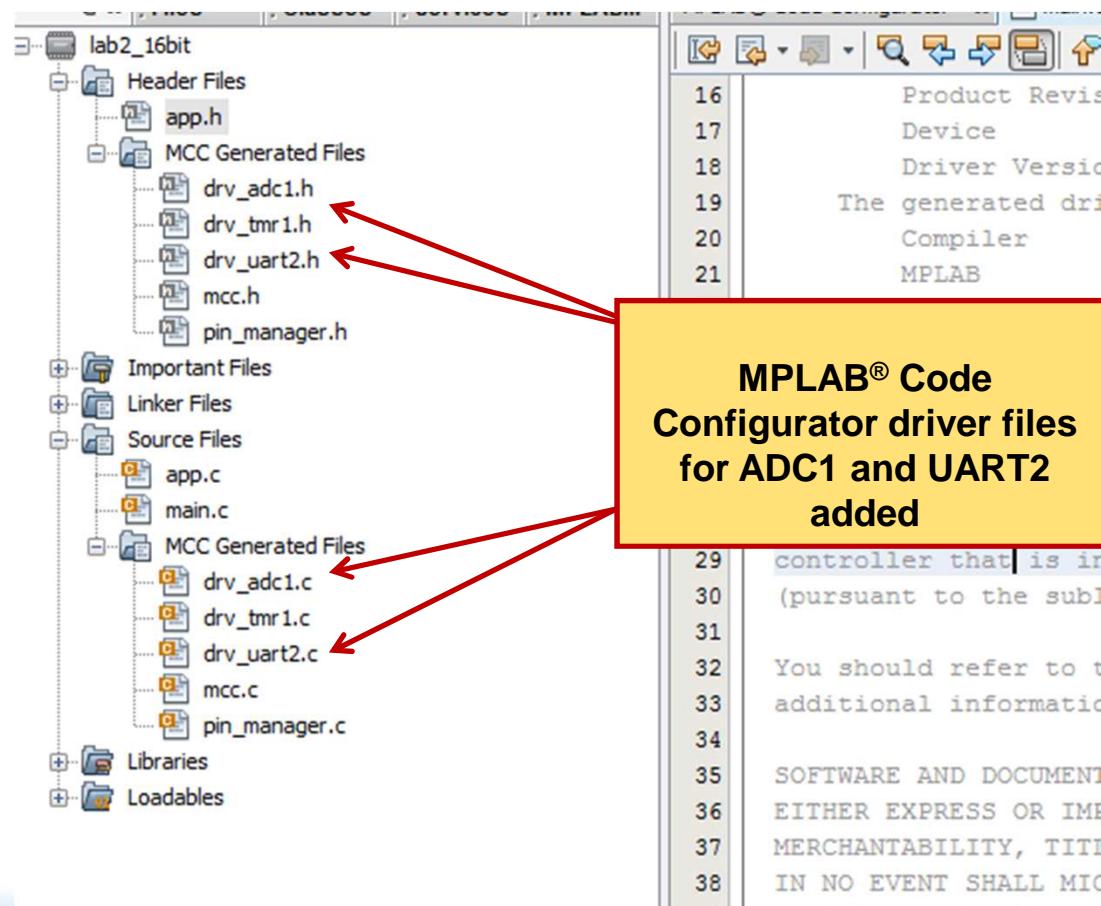
PIC24 Lab 2

- Generate the code and close MPLAB® Code Configurator



PIC24 Lab 2

- Four more files added to the project



PIC24 Lab 2

- Open main.c and add the code shown

```
68
69     /*Start the timer with Period 1s */
70     DRV_TMR1_Start();
71     /*Select the ADC channel to which Potentiometer is connected*/
72     DRV_ADC1_ChannelSelect(DRV_ADC1_POT_CHANNEL);
73     DRV_ADC1_Start();
74     while(1)
75     {
76         DRV_TMR1_Tasks();
77         if(DRV_TMR1_GetElapsedThenClear()){
78             //Toggle LED D10 every 1s
79             LED_D10_Toggle();
80
81             adcResult = DRV_ADC1_ConversionGet(); //Digital value is read
82             ConvertADCVoltage(adcResult); //adc result value is converted to analog voltage
83             sprintf(bufferWrite, "Potentiometer Analog Voltage is = %d.%d%d V \r\n",adones, adten
84             Write2UART(bufferWrite, strlen(bufferWrite)); //Analog voltage value is transmitted ov
85     }
```

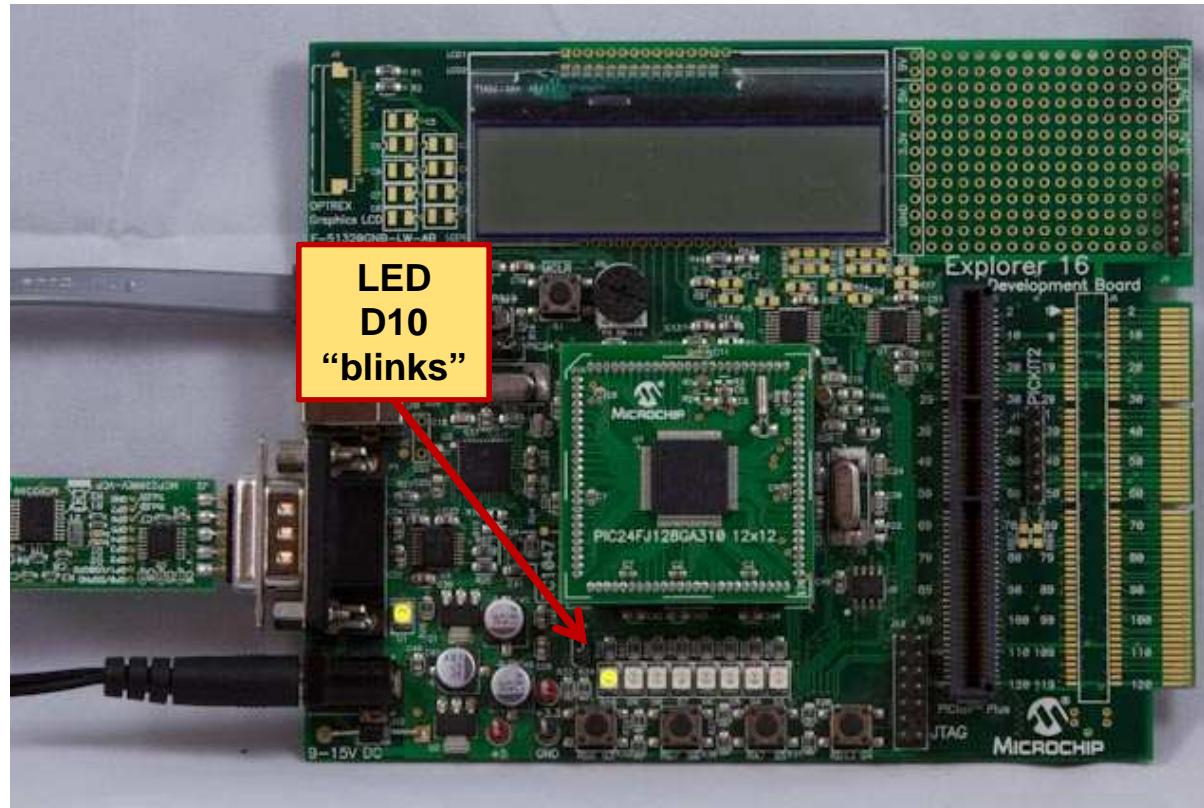
PIC24 Lab 2

- “Make and Program” the project



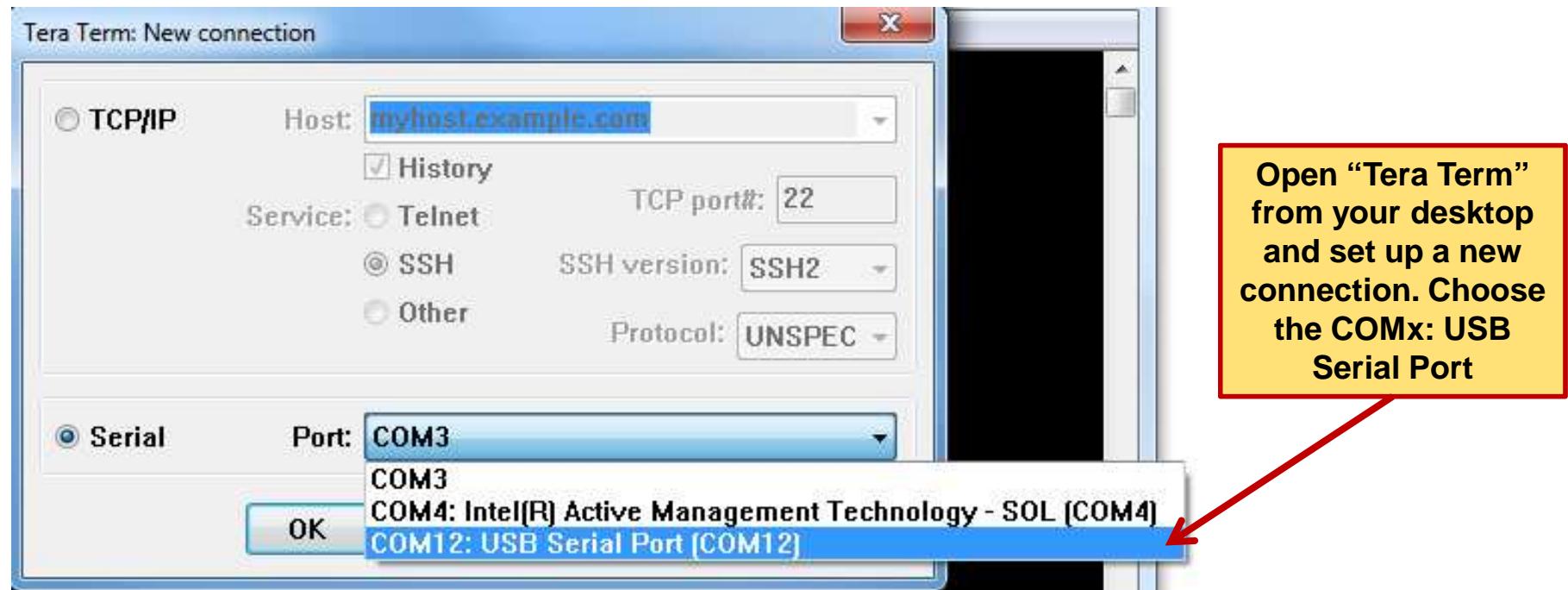
PIC24 Lab 2

- **LED_D10 toggles every second**



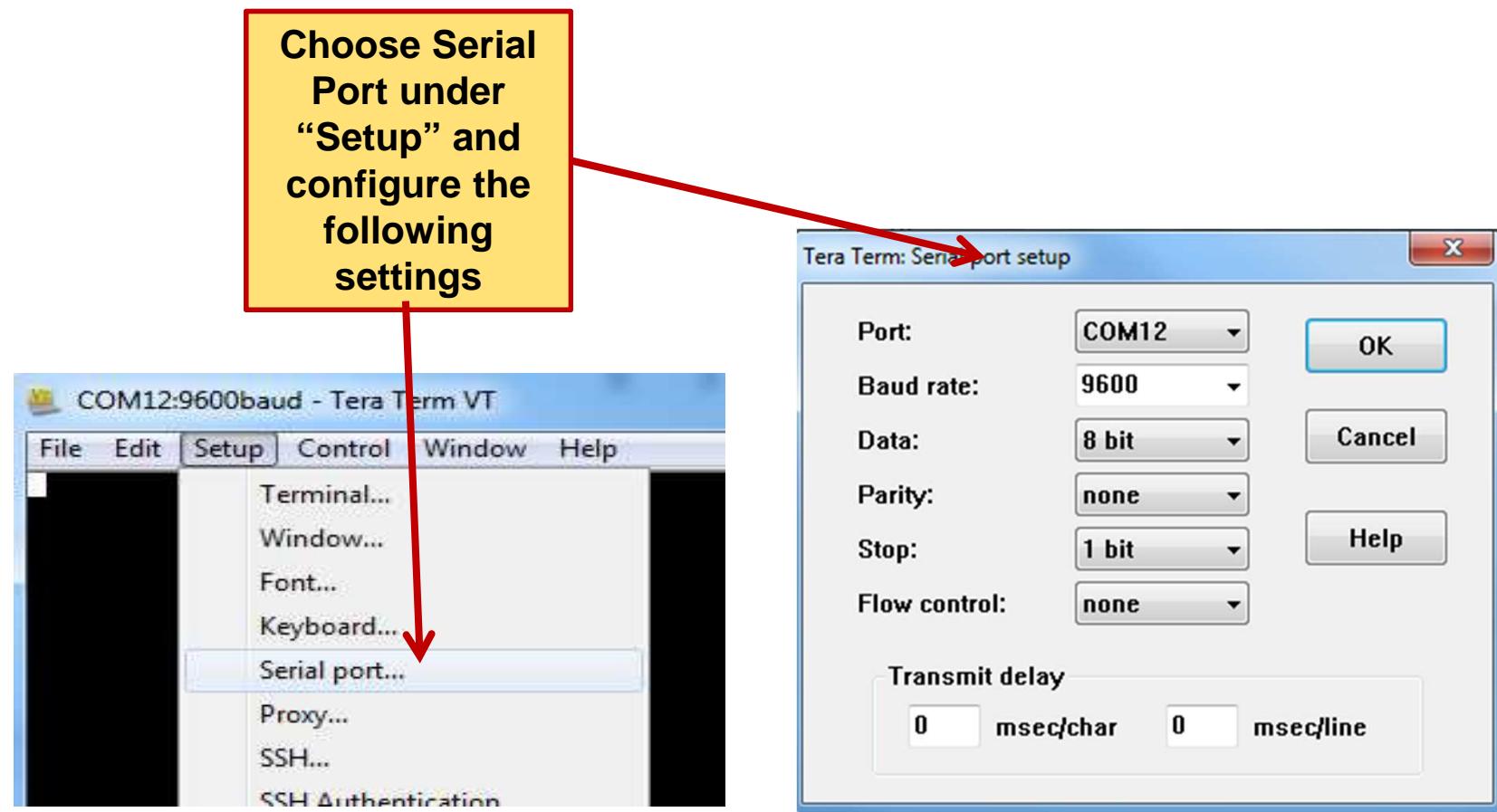
PIC24 Lab 2

- Start Tera Term
- Select the USB Serial Port



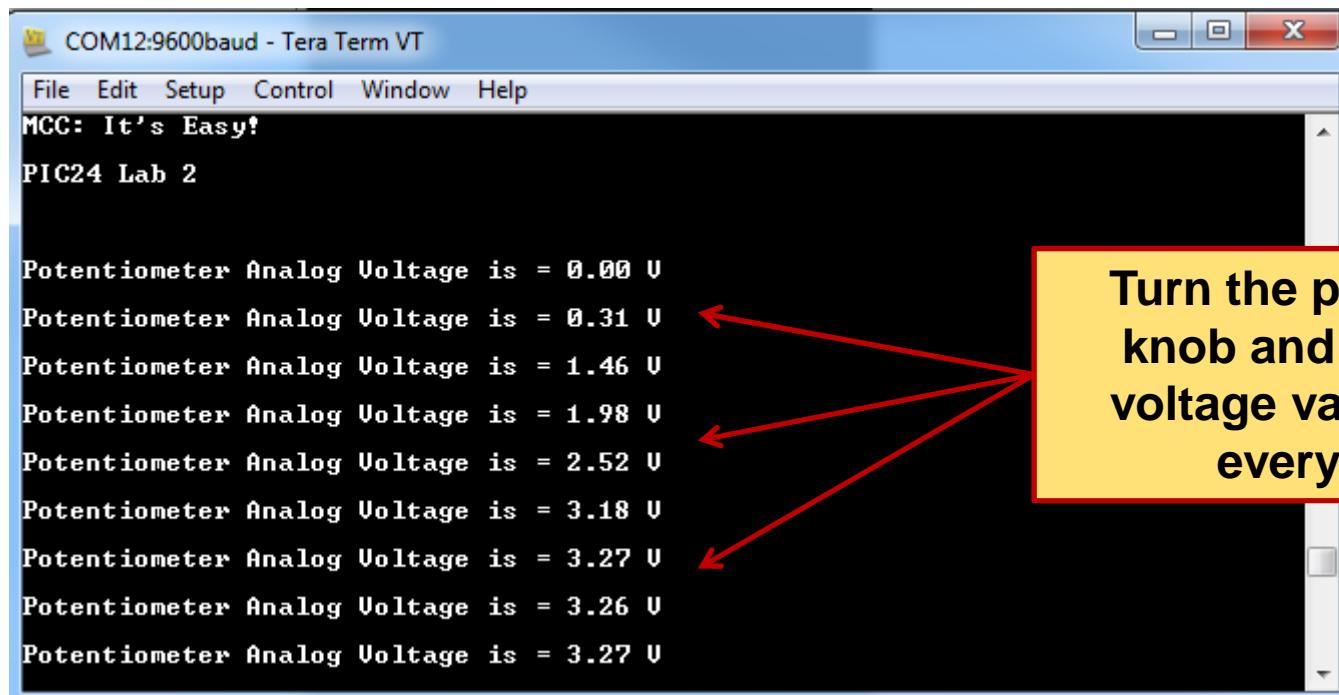
PIC24 Lab 2

- **Setup the Serial Port**



PIC24 Lab 2

- Tera Term shows the potentiometer voltage updating every second



COM12:9600baud - Tera Term VT

File Edit Setup Control Window Help

MCC: It's Easy!

PIC24 Lab 2

```
Potentiometer Analog Voltage is = 0.00 V
Potentiometer Analog Voltage is = 0.31 V
Potentiometer Analog Voltage is = 1.46 V
Potentiometer Analog Voltage is = 1.98 V
Potentiometer Analog Voltage is = 2.52 V
Potentiometer Analog Voltage is = 3.18 V
Potentiometer Analog Voltage is = 3.27 V
Potentiometer Analog Voltage is = 3.26 V
Potentiometer Analog Voltage is = 3.27 V
```

Turn the potentiometer knob and observe the voltage value updating every second

PIC24 Lab 2 Summary

- We configured ADC1 to sample the Potentiometer channel and convert it to a voltage.
- We configured UART2 to transmit the converted potentiometer voltage onto Tera Term

PIC24 Lab 3: Using Interrupts

PIC24 Lab 3 Objectives

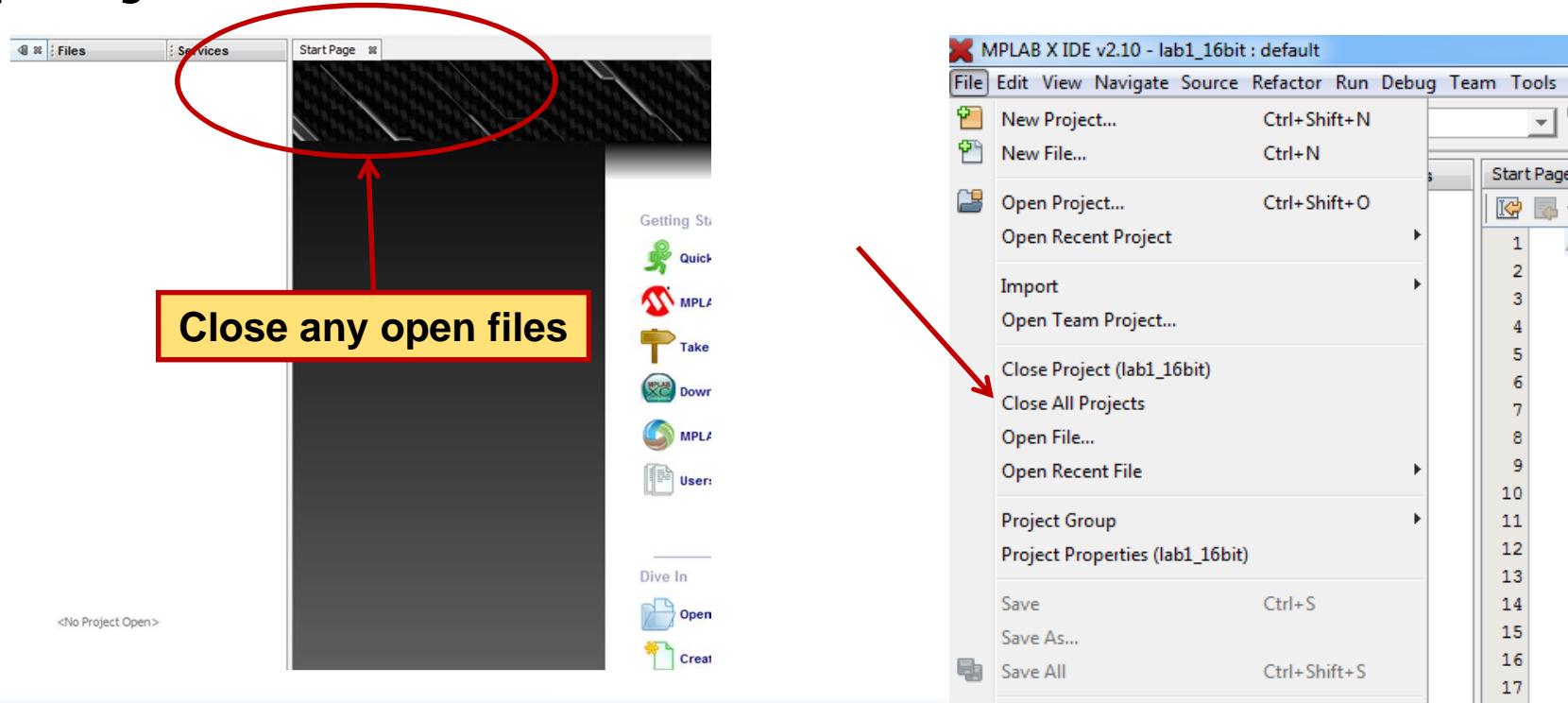
- **Converting Lab 2 into an Interrupt Mode application**
- **Read potentiometer voltage using ADC and display it on serial port using UART and toggle an LED every second using a Timer**

PIC24 Lab 3 Overview

- Close any files open in the editor, and close all MPLAB® X IDE Projects
- Open lab3_16bit.X
- Configure TMR1 module period 1s in Interrupt Mode
- Configure UART2 for 9600 Baud, Parity None, 1 Stop Bit and Interrupt Mode

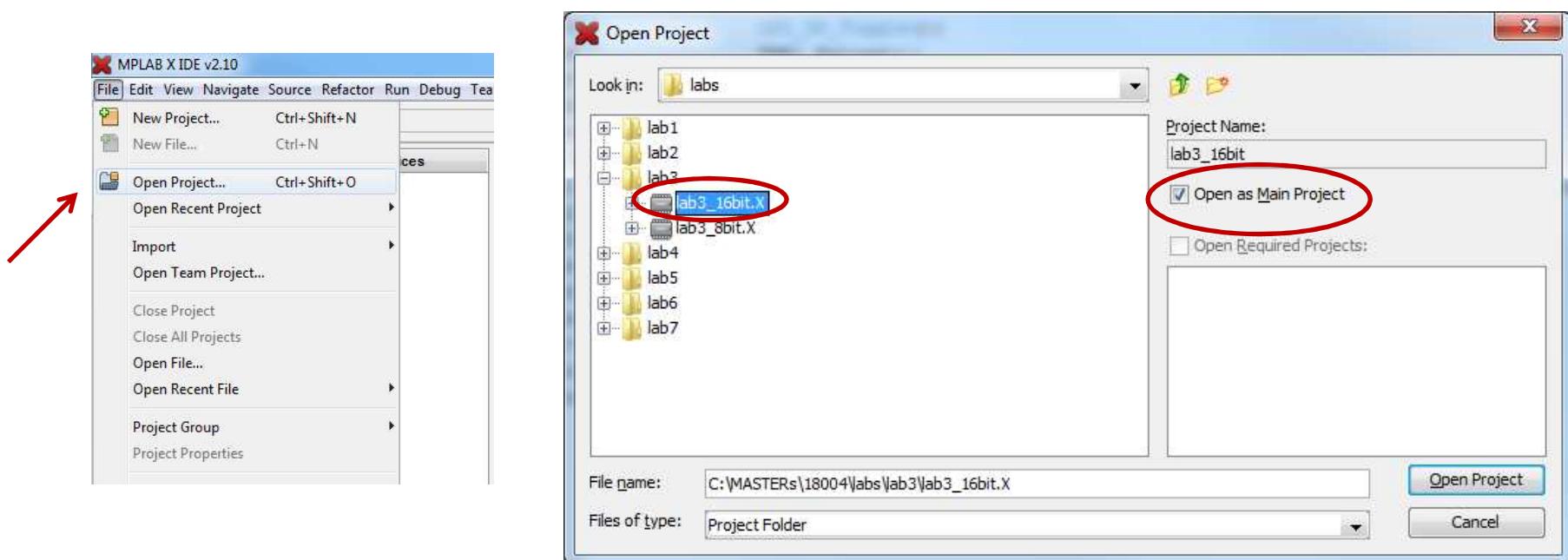
PIC24 Lab 3

- Close any open files in the editor window, and all open MPLAB® X IDE projects



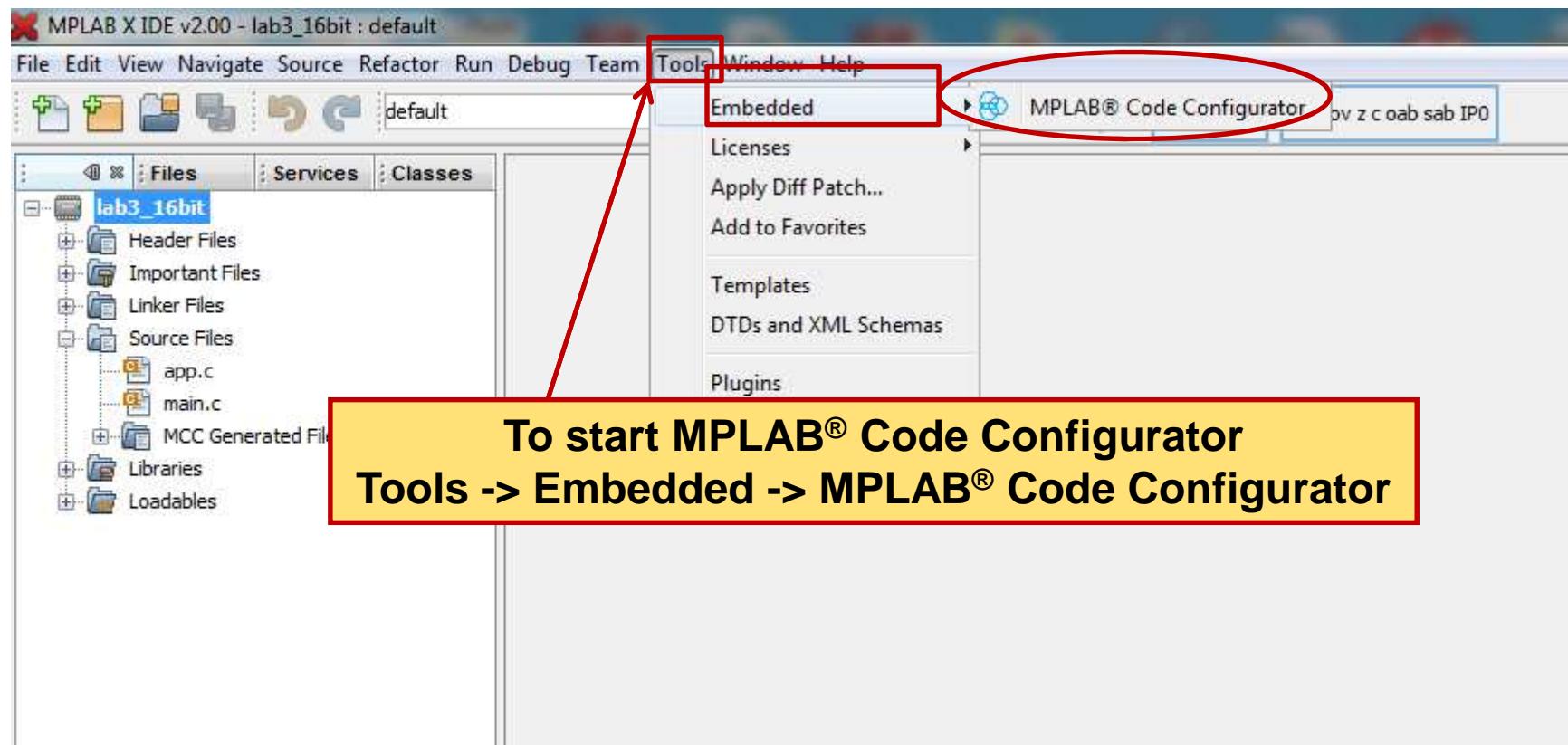
PIC24 Lab 3

- Open lab3_16bit.X



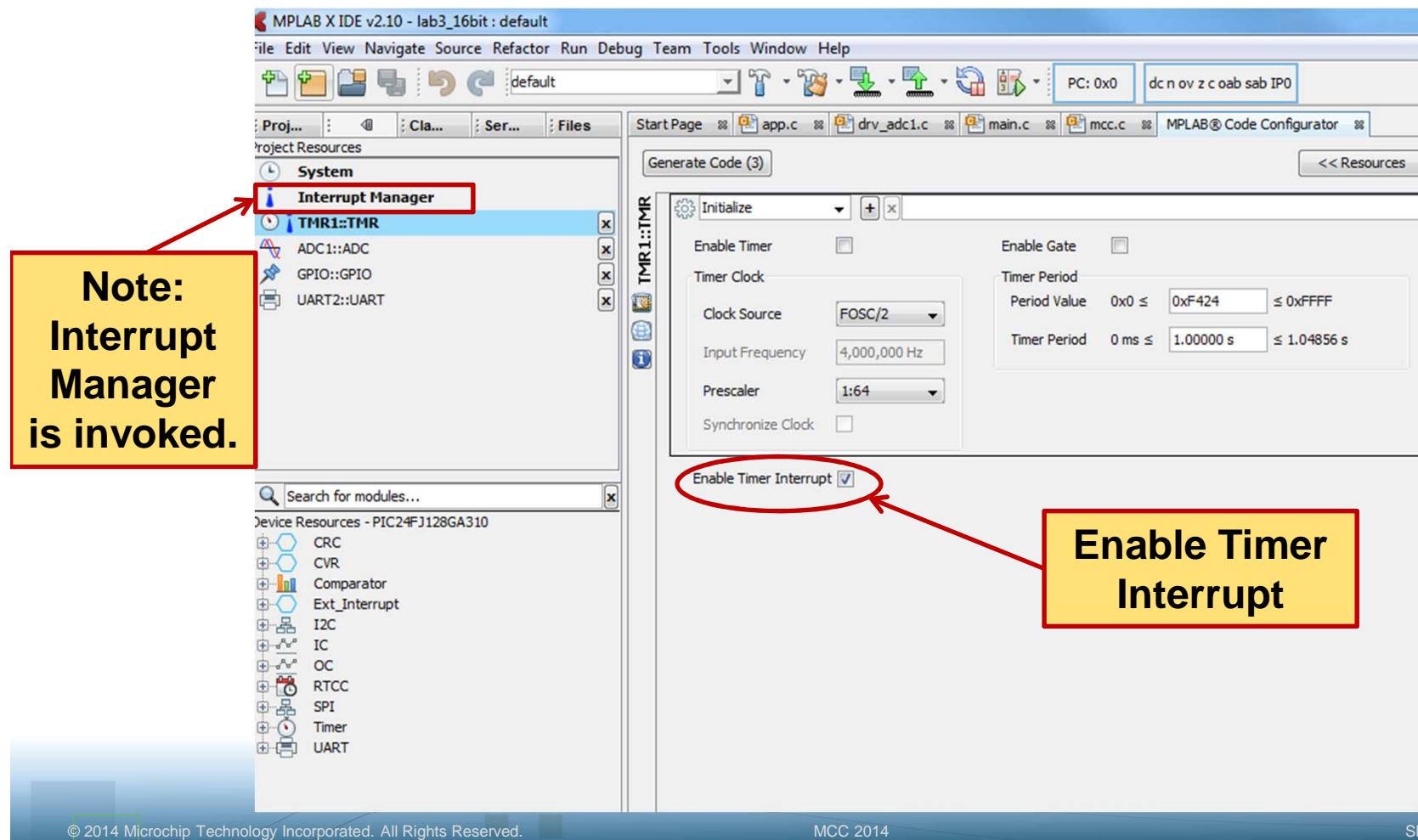
PIC24 Lab 3

- Start MPLAB® Code Configurator



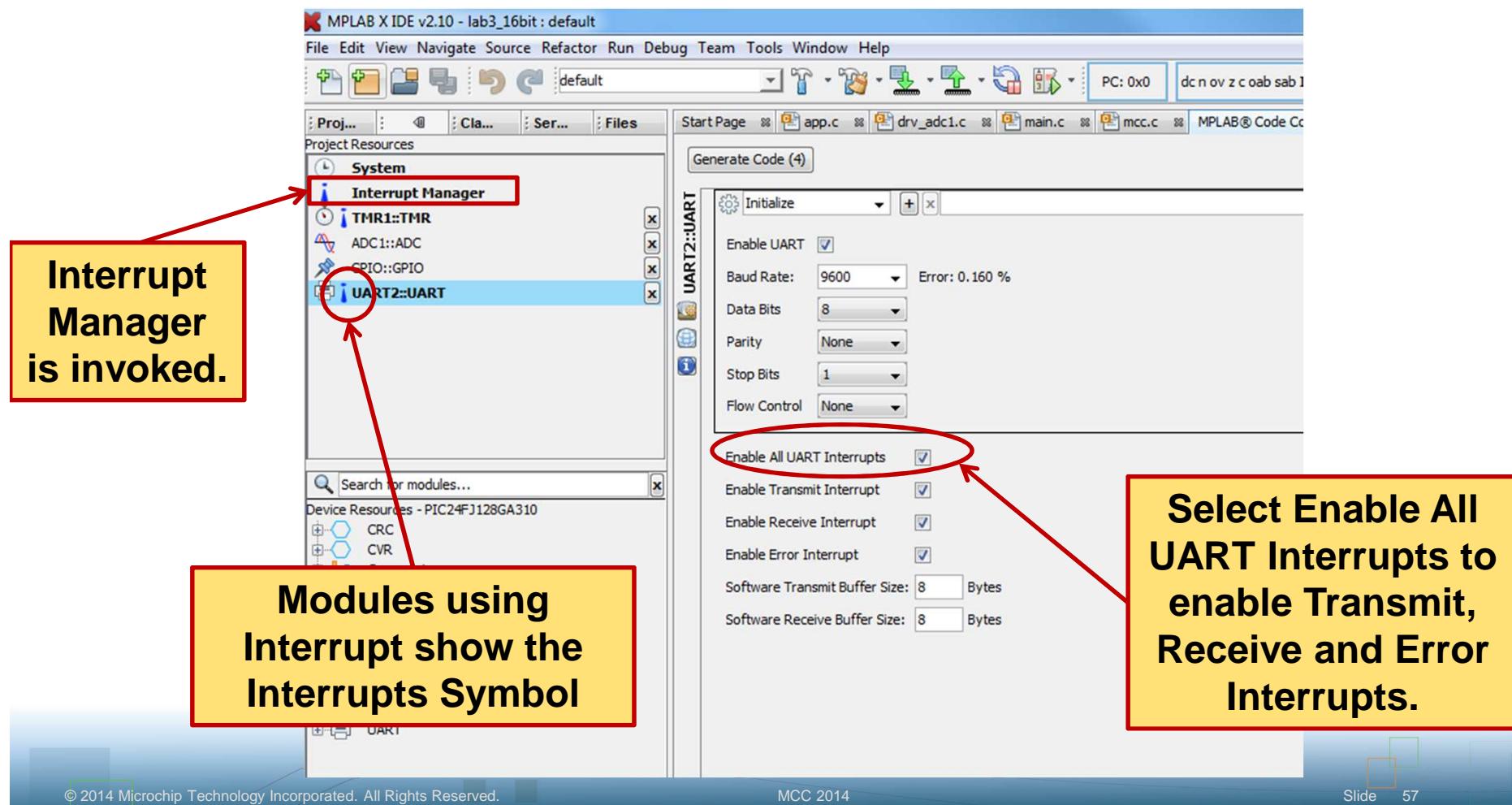
PIC24 Lab 3

- Select Timer 1 from the Project Resources and Enable Timer Interrupt.



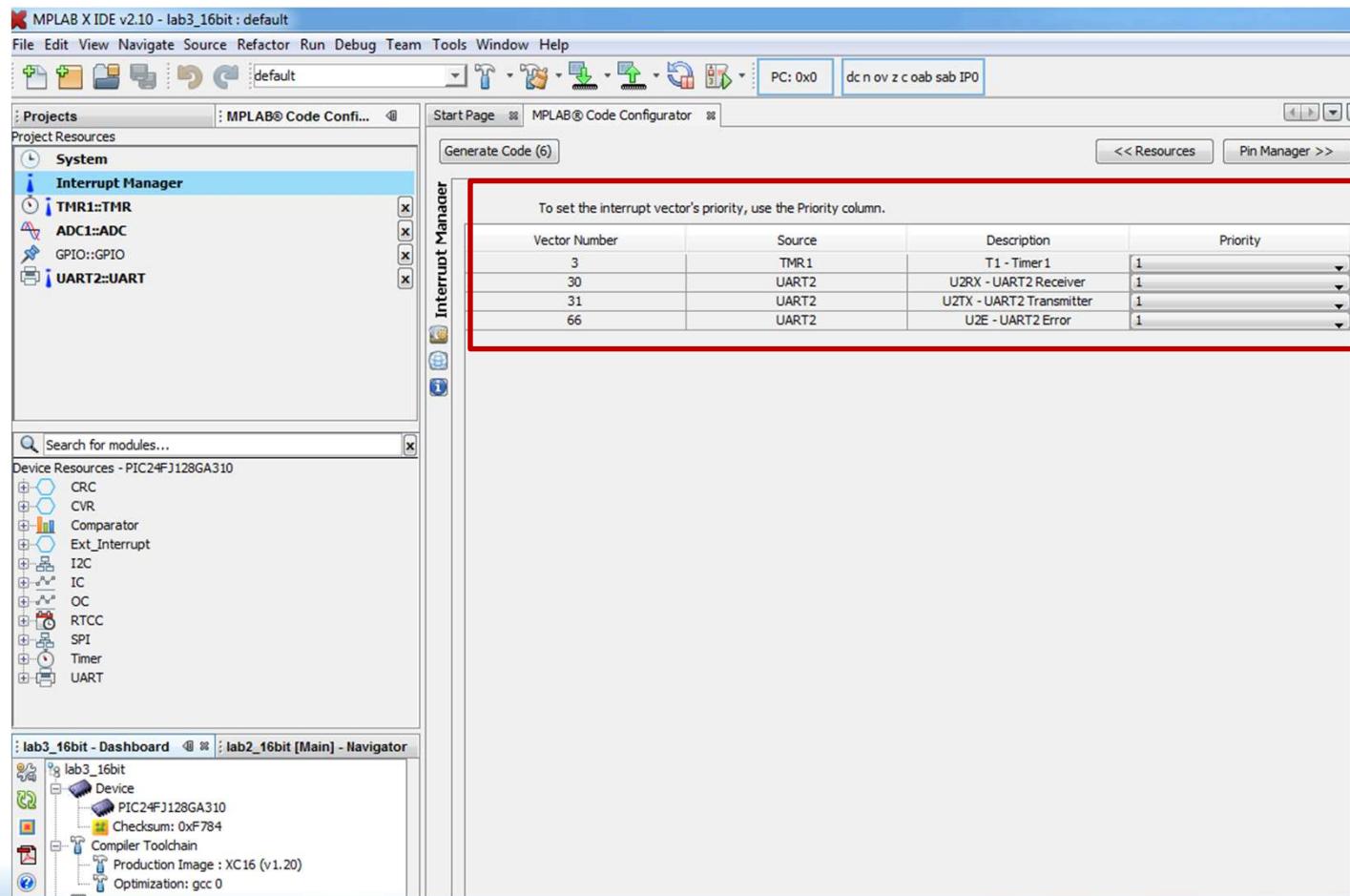
PIC24 Lab 3

- Select UART2 and Enable All UART Interrupts.



PIC24 Lab 3

- Open Interrupt Manager to see the UART and Timer Interrupts added



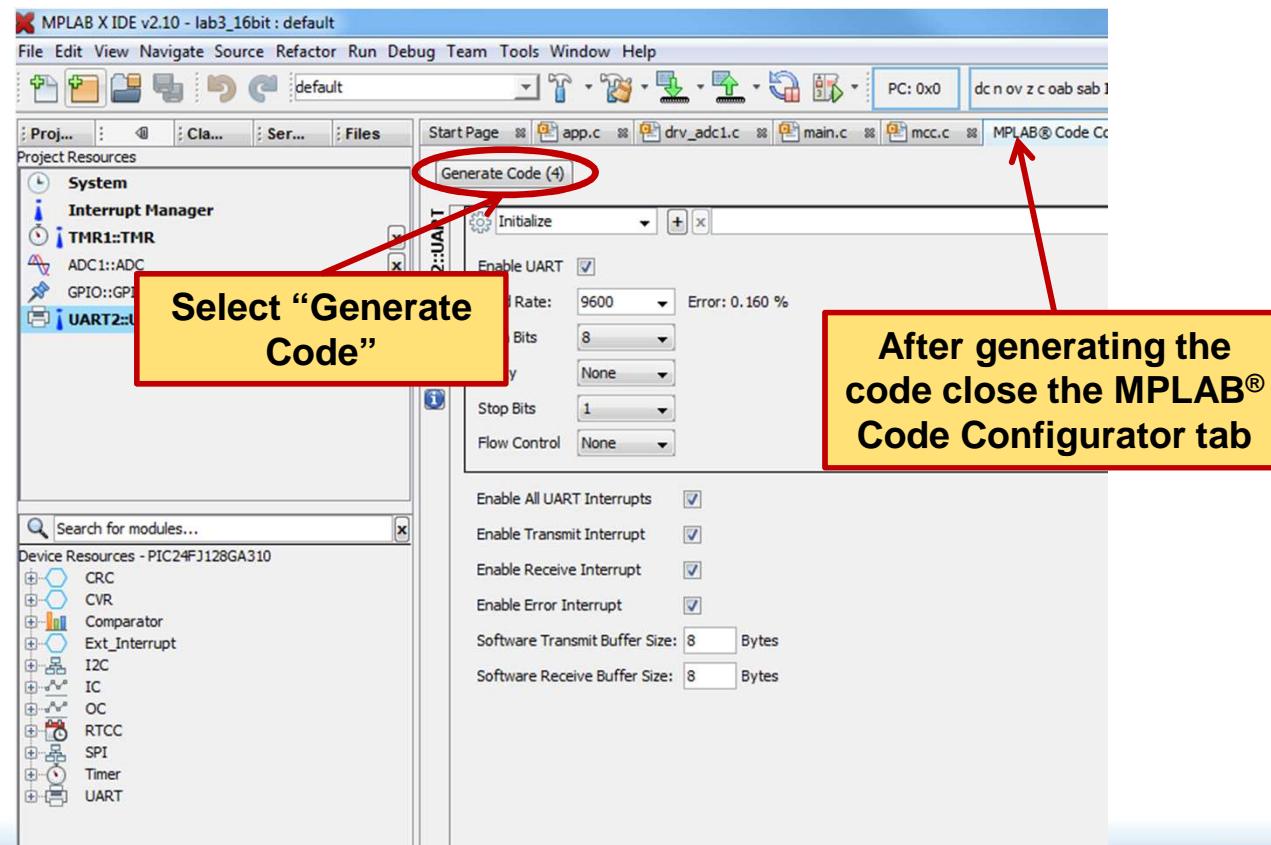
PIC24 Lab 3

- Did you know ...
- that between the Project Resources and Composer windows, the MCC icons provide helpful links?



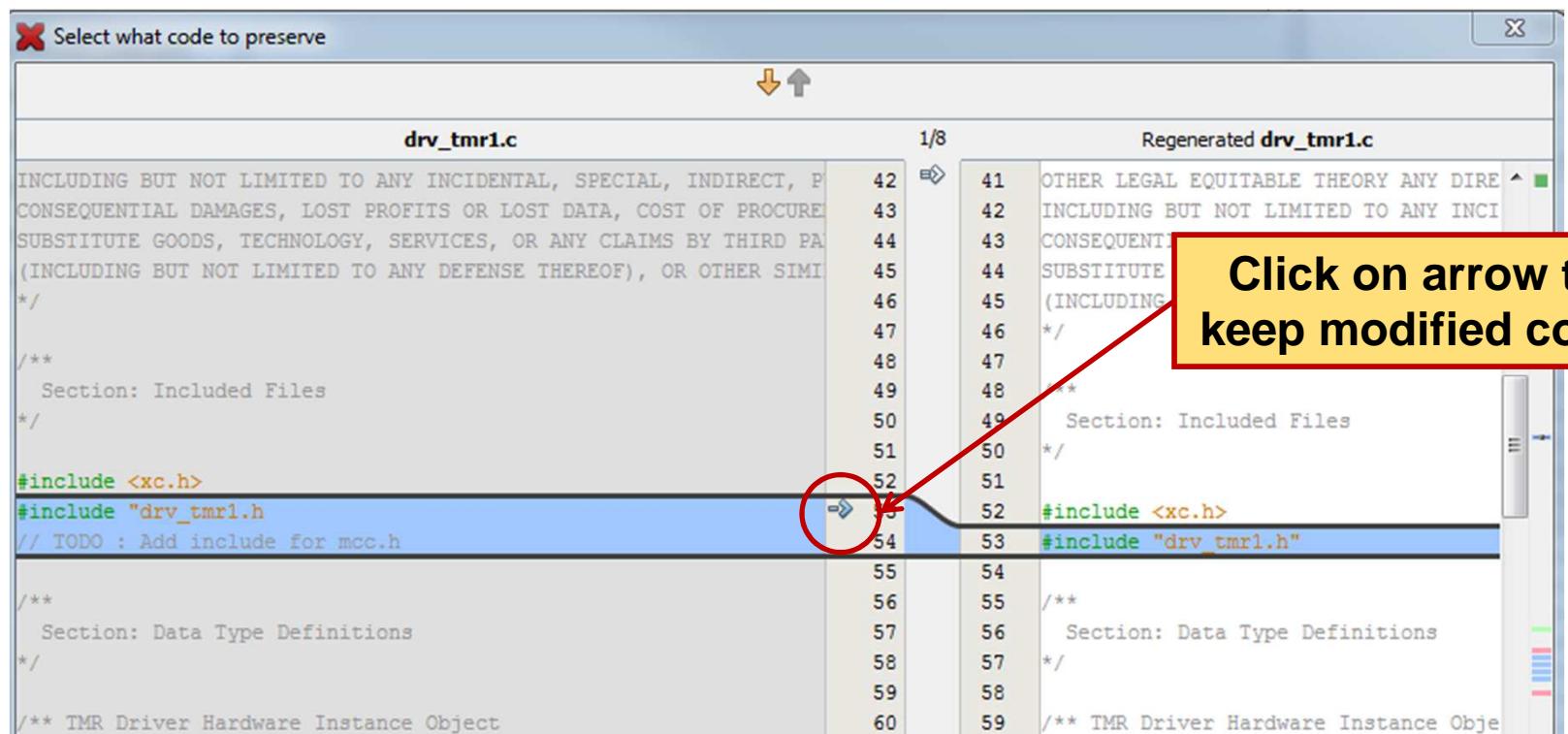
PIC24 Lab 3

- Generate the code and close MPLAB® Code Configurator



PIC24 Lab 3

- Use the difference window to keep the line of modified code



Select what code to preserve

drv_tmr1.c

```
INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, P 42
CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCURE 43
SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PA 44
(INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMI 45
*/
/** Section: Included Files */
#include <xc.h>
#include "drv_tmrl.h"
// TODO : Add include for mcc.h

/*
Section: Data Type Definitions
*/
/* TMR Driver Hardware Instance Object
```

1/8

Regenerated drv_tmr1.c

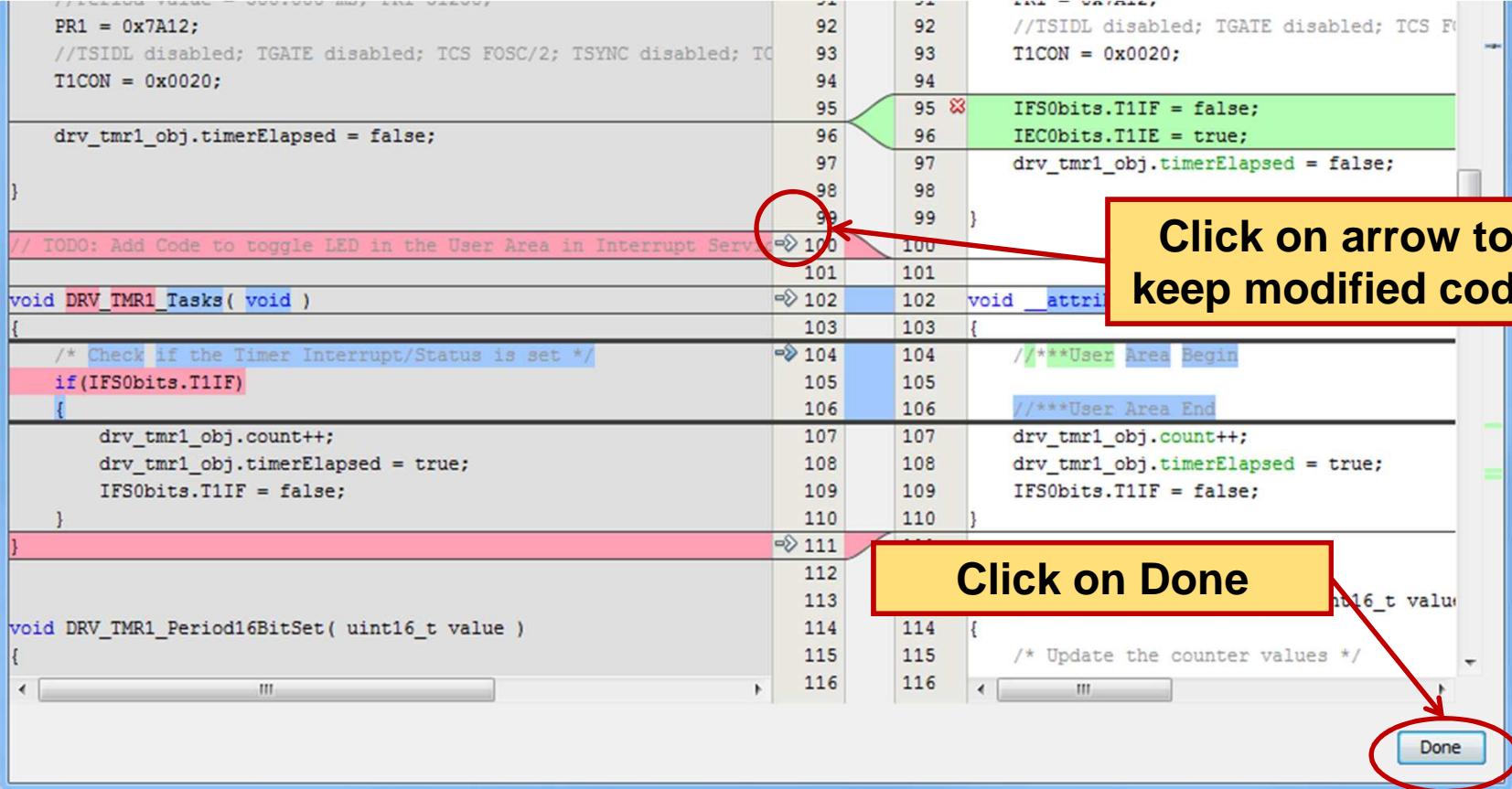
```
OTHER LEGAL EQUITABLE THEORY ANY DIRE 41
 INCLUDING BUT NOT LIMITED TO ANY INCI 42
 CONSEQUENT 43
 SUBSTITUTE 44
 (INCLUDING 45
 */
/*
Section: Included Files
*/
#include <xc.h>
#include "drv_tmrl.h"

/*
Section: Data Type Definitions
*/
/** TMR Driver Hardware Instance Obj
```

Click on arrow to keep modified code

PIC24 Lab 3

- Use the difference window to keep the line of modified code highlighted.



```

; File: DRV_TMR1.S
;
; PR1 = 0x7A12;
; //TSIDL disabled; TGATE disabled; TCS FOSC/2; TSYNC disabled; TO
; T1CON = 0x0020;
;
;drv_tmr1_obj.timerElapsed = false;
;
; // TODO: Add Code to toggle LED in the User Area in Interrupt Serv
;void DRV_TMR1_Tasks( void )
{
    /* Check if the Timer Interrupt/Status is set */
    if(IFSObits.T1IF)
    {
        drv_tmr1_obj.count++;
        drv_tmr1_obj.timerElapsed = true;
        IFS0bits.T1IF = false;
    }
}
;
void DRV_TMR1_Period16BitSet( uint16_t value )
{
}
;
```

Click on arrow to keep modified code

Click on Done

PIC24 Lab 3

- Did you know ...
- that you can resize the package view in the Pin Manager by pressing **<CTRL>** and using the scroll wheel on the mouse



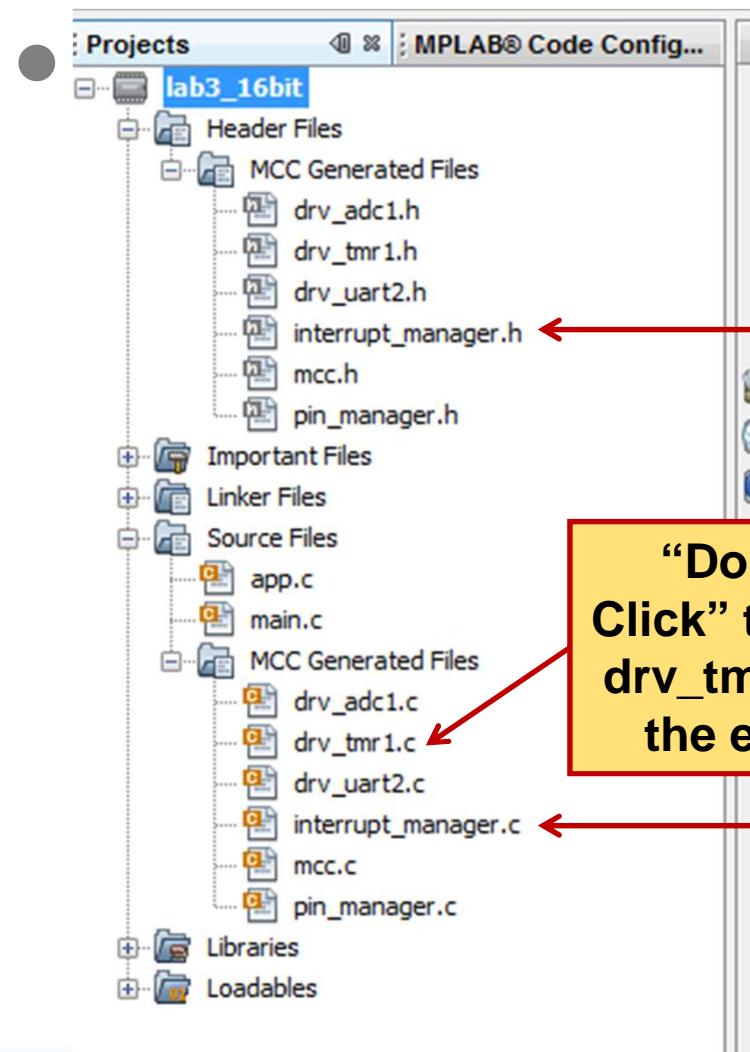
PIC24 Lab 3

- Did you know ...
- that you can bring up the Project and Device resource windows by pressing the ‘Resources’ button in the Composer window



PIC24 Lab 3

Open `drv_tmr1.c`

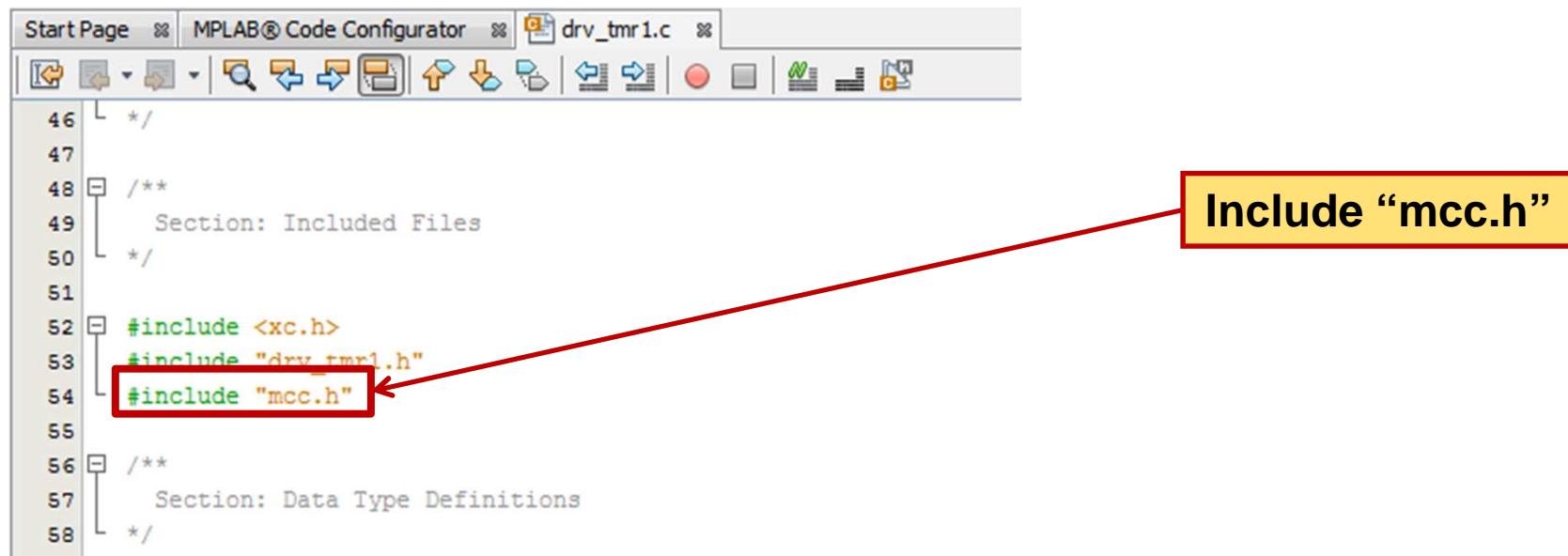


**“Double Click” to open
`drv_tmr1.c` in
the editor**

Note:
`interrupt_manager.c`
and
`interrupt_manager.h`
files are added to
the project.

PIC24 Lab 3

- Open **drv_tmr1.c** and Include “**mcc.h**”

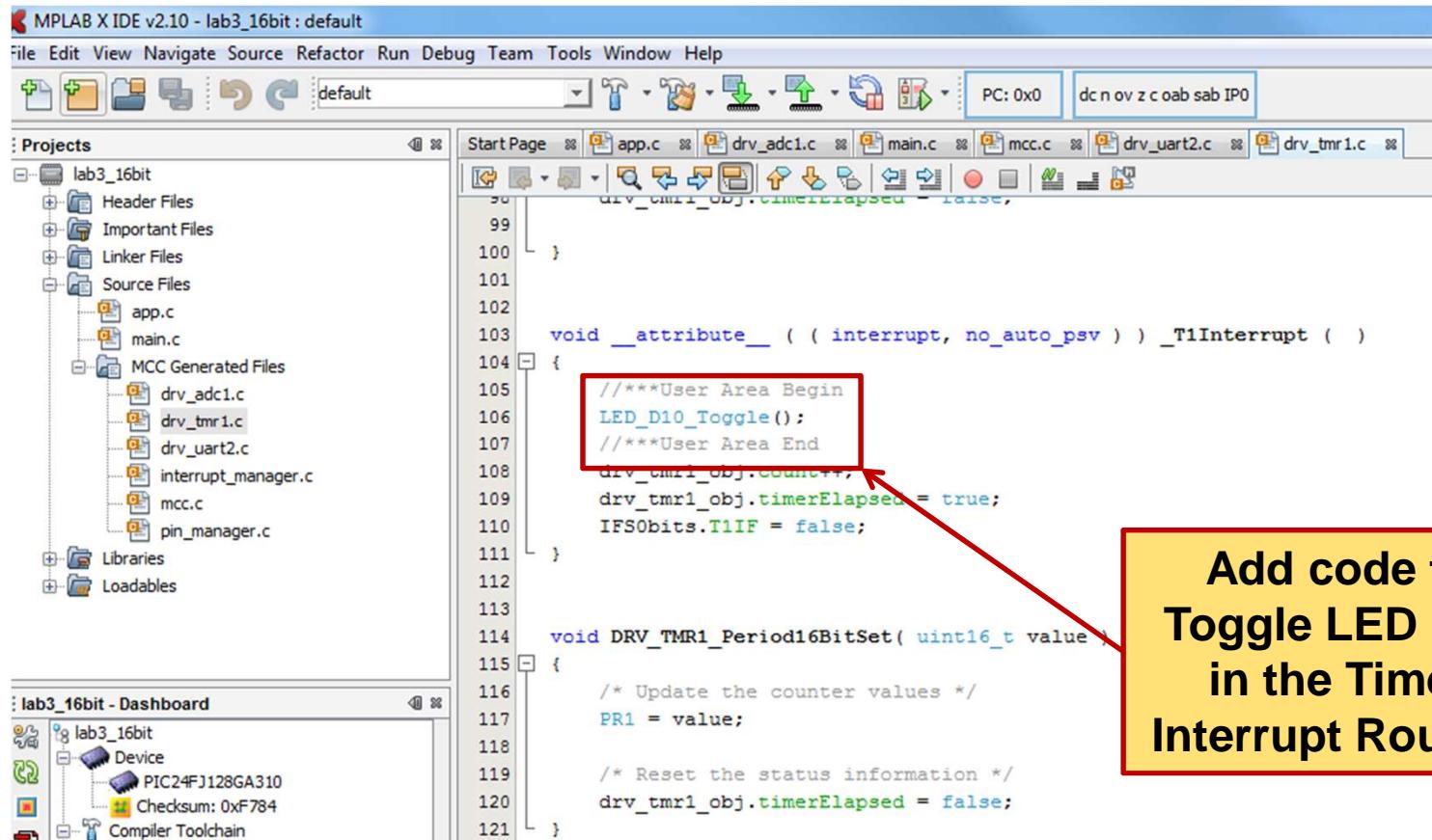


```
Start Page  ❀  MPLAB® Code Configurator  ❀  drv_tmr1.c  ❀
46  */
47
48  /**
49   * Section: Included Files
50  */
51
52  #include <xc.h>
53  #include "drv_tmr1.h"
54  #include "mcc.h"  #include "mcc.h"
55
56  /**
57   * Section: Data Type Definitions
58  */
```

Include “mcc.h”

PIC24 Lab 3

- Add code to toggle LED D10



The screenshot shows the MPLAB X IDE interface. The Source Editor window displays C code for a PIC24 microcontroller. A red box highlights the following code block in the `_T1Interrupt` routine:

```
105 //***User Area Begin
106 LED_D10_Toggle();
107 //***User Area End
108
109     drv_tmr1_obj.count++,
110     drv_tmr1_obj.timerElapsed = true;
111     IFS0bits.T1IF = false;
112
113
114 void DRV_TMR1_Period16BitSet( uint16_t value )
115 {
116     /* Update the counter values */
117     PR1 = value;
118
119     /* Reset the status information */
120     drv_tmr1_obj.timerElapsed = false;
121 }
```

A yellow callout box with a red border contains the instruction: "Add code to Toggle LED D10 in the Timer Interrupt Routine". A red arrow points from this callout to the highlighted code block.

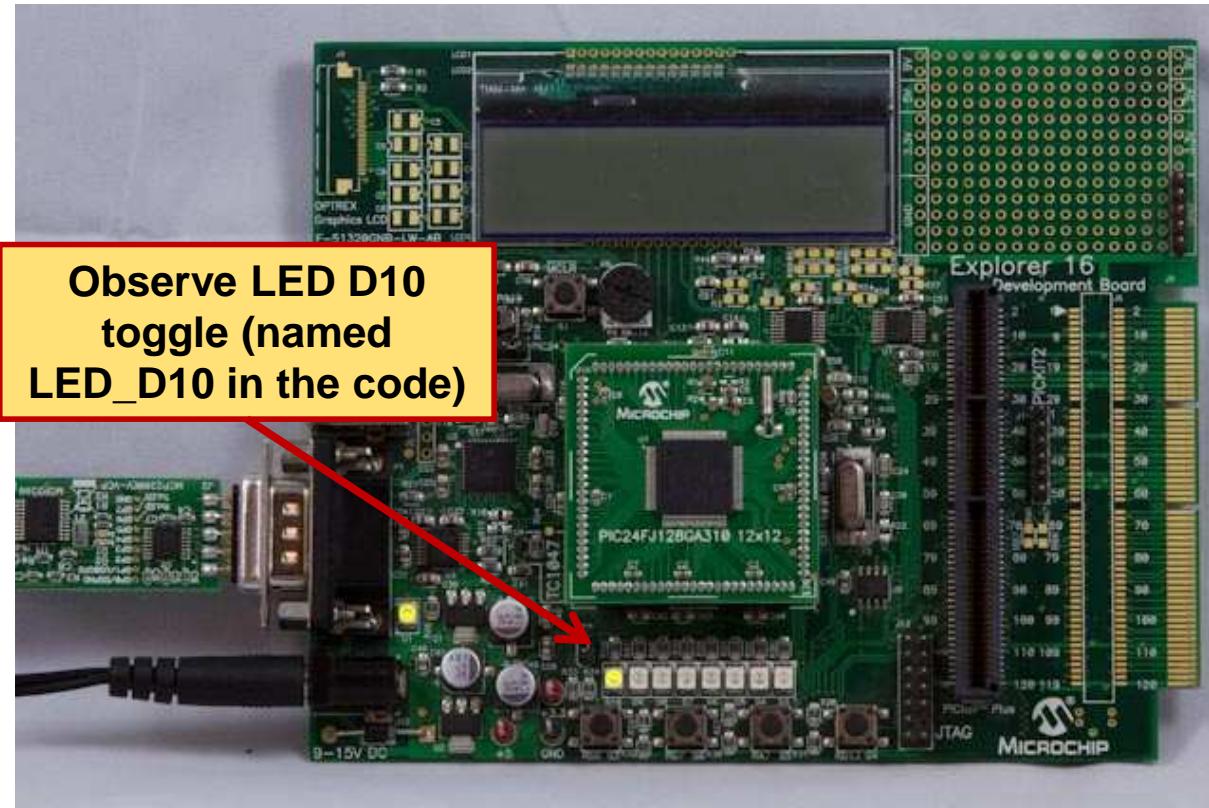
PIC24 Lab 3

- “Make and Program” the project



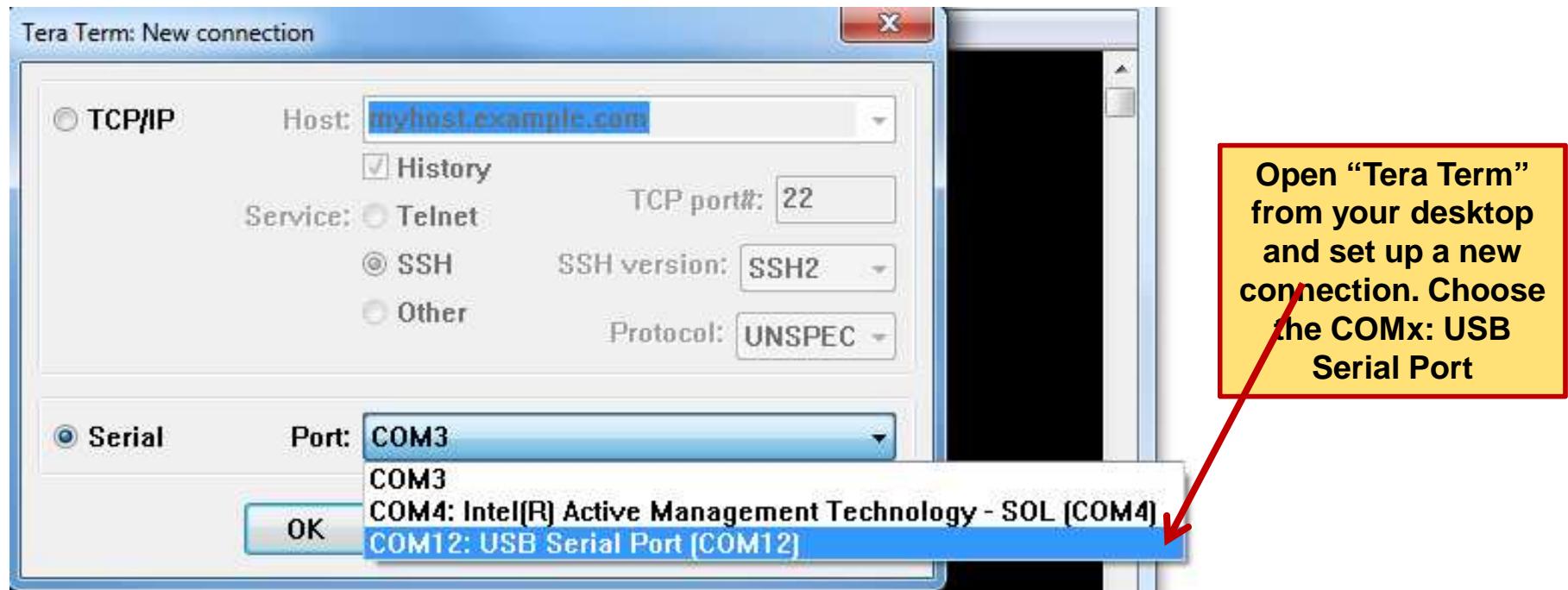
PIC24 Lab 3

- **LED_D10 toggles every second**



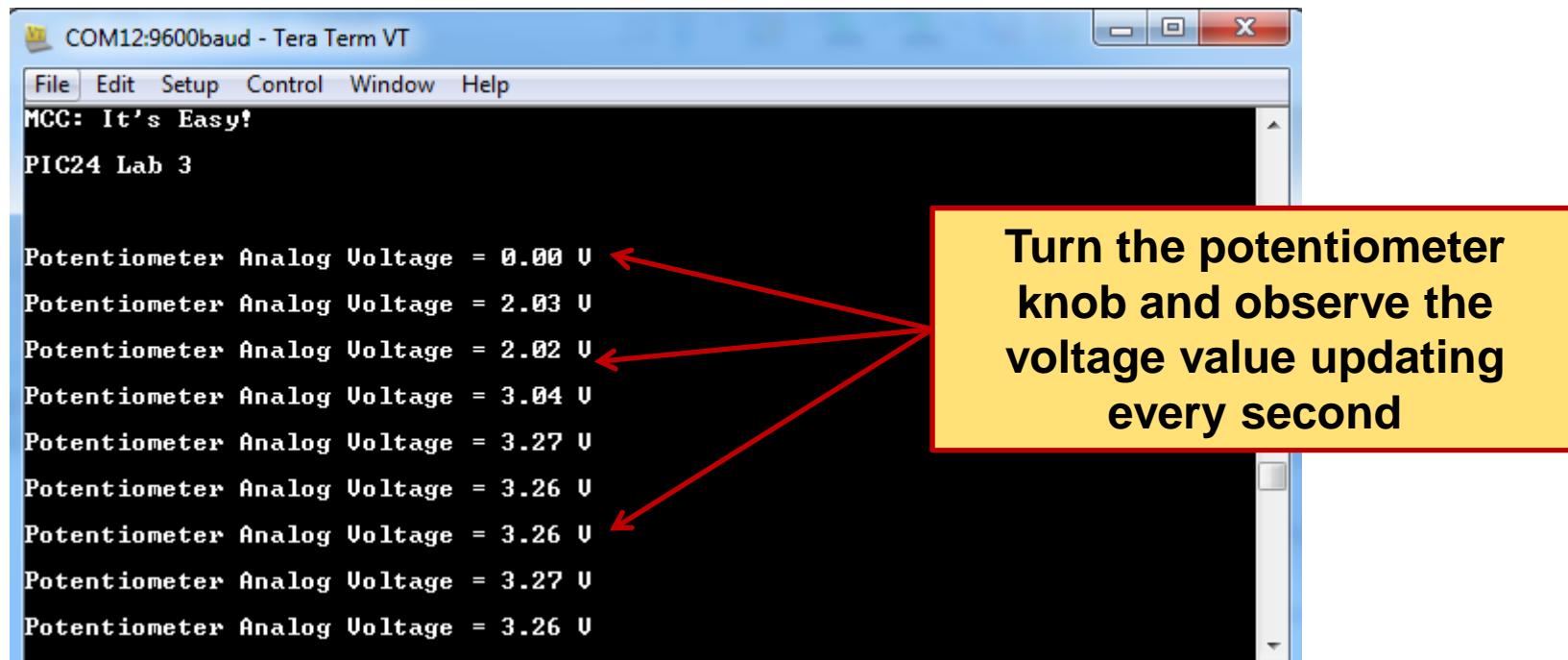
PIC24 Lab 3

- Start Tera Term
- Select the USB Serial Port



PIC24 Lab 3

- Tera Term shows the potentiometer voltage updating every second



```
COM12:9600baud - Tera Term VT
File Edit Setup Control Window Help
MCC: It's Easy!
PIC24 Lab 3

Potentiometer Analog Voltage = 0.00 V
Potentiometer Analog Voltage = 2.03 V
Potentiometer Analog Voltage = 2.02 V
Potentiometer Analog Voltage = 3.04 V
Potentiometer Analog Voltage = 3.27 V
Potentiometer Analog Voltage = 3.26 V
Potentiometer Analog Voltage = 3.26 V
Potentiometer Analog Voltage = 3.27 V
Potentiometer Analog Voltage = 3.26 V
```

Turn the potentiometer knob and observe the voltage value updating every second

PIC24 Lab 3 Summary

- We configured TMR1 to use interrupts
- We configured UART2 to use interrupts
- We wrote code to be executed every time the TMR1 interrupt is triggered

PIC24 Lab 4: Building a Small Application

PIC24 Lab 4 Objectives

- **Create a menu of commands**
- **Select functions from the menu using push-buttons**
- **From the menu choose between the following operations:**
 - Read Potentiometer using ADC
 - Read Temperature sensor using ADC
 - Toggle a LED once per second

PIC24 Lab 4 Overview

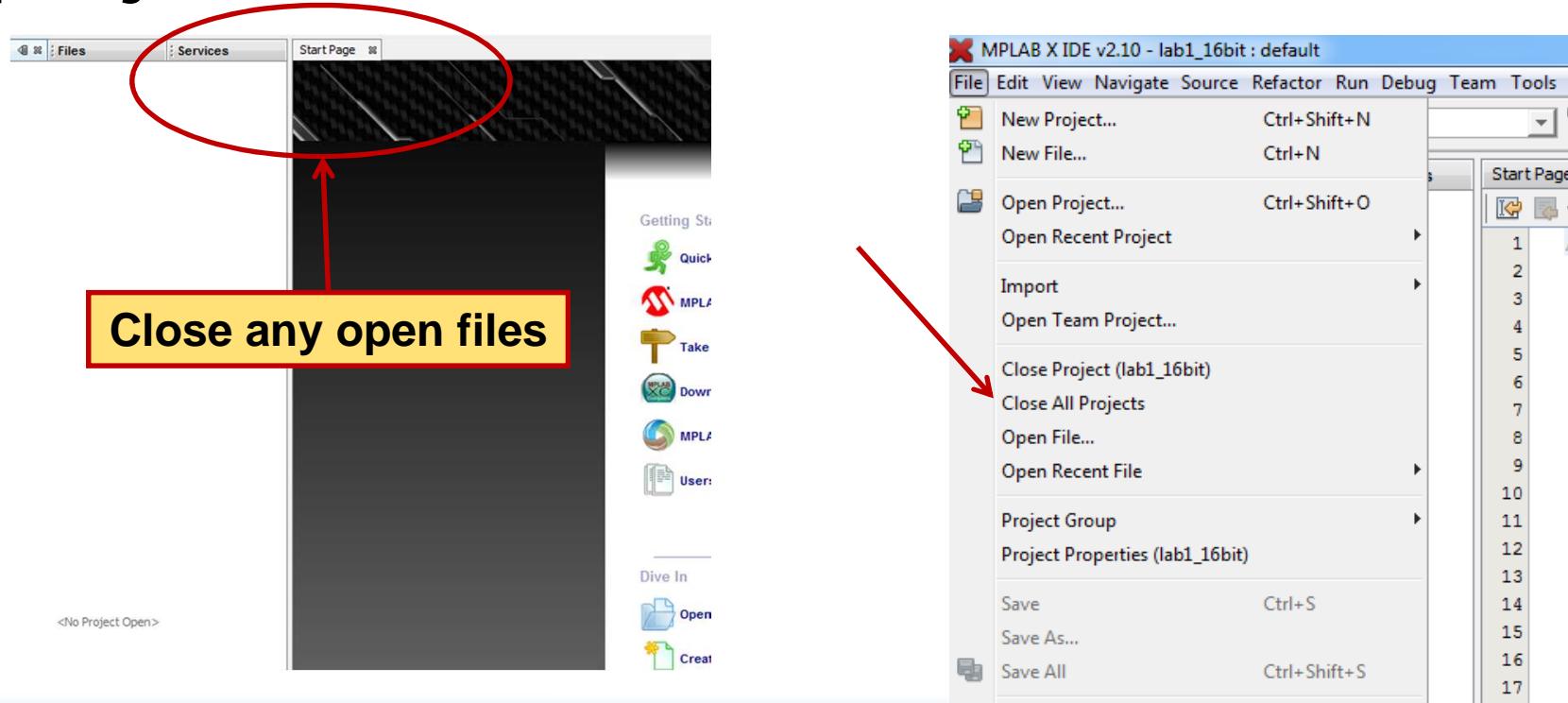
- Close any files open in the editor, and close all MPLAB® X IDE Projects
- Open lab4_16bit.X

PIC24 Lab 4 Overview

- **Setup TMR1 for multiple initializers**
 - DebounceDelay - 0.03 s
 - DisplayPotDelay - 0.3 s
 - DisplayTempDelay - 0.6 s
 - ToggleDelay - 1 s
- **Setup ADC1**
 - Add AN4, name: TEMP_CHANNEL
- **Setup GPIO**
 - Setup Switch S3 and Switch S6

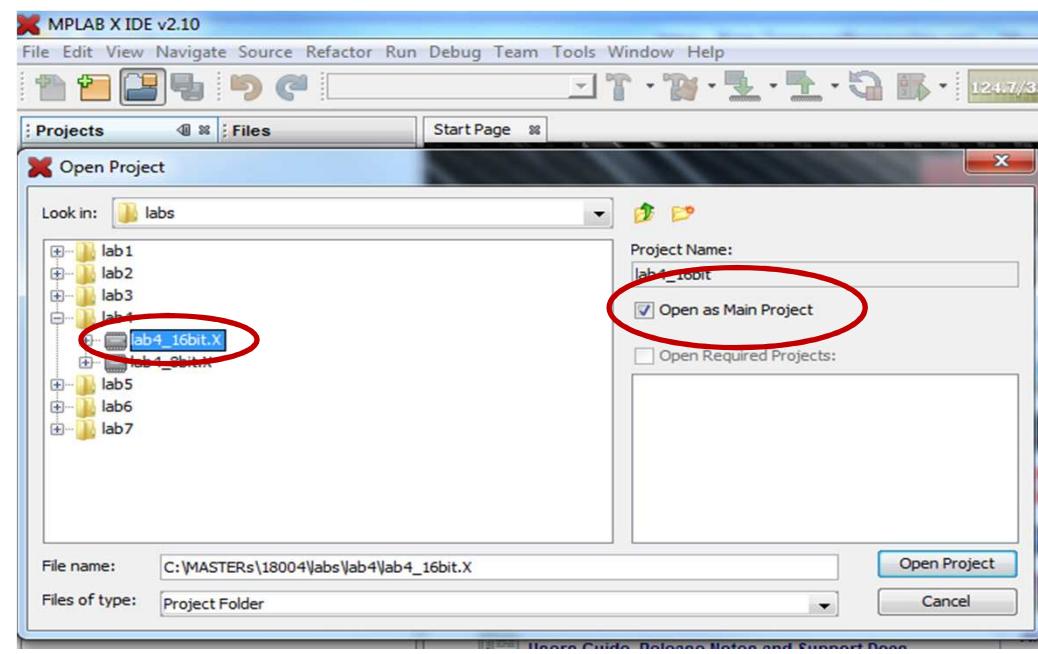
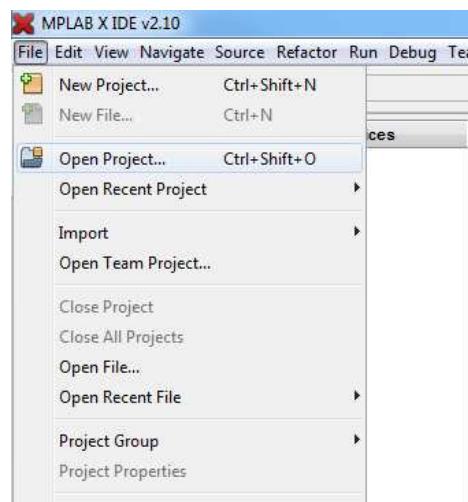
PIC24 Lab 4

- Close any open files in the editor window, and all open MPLAB® X IDE projects



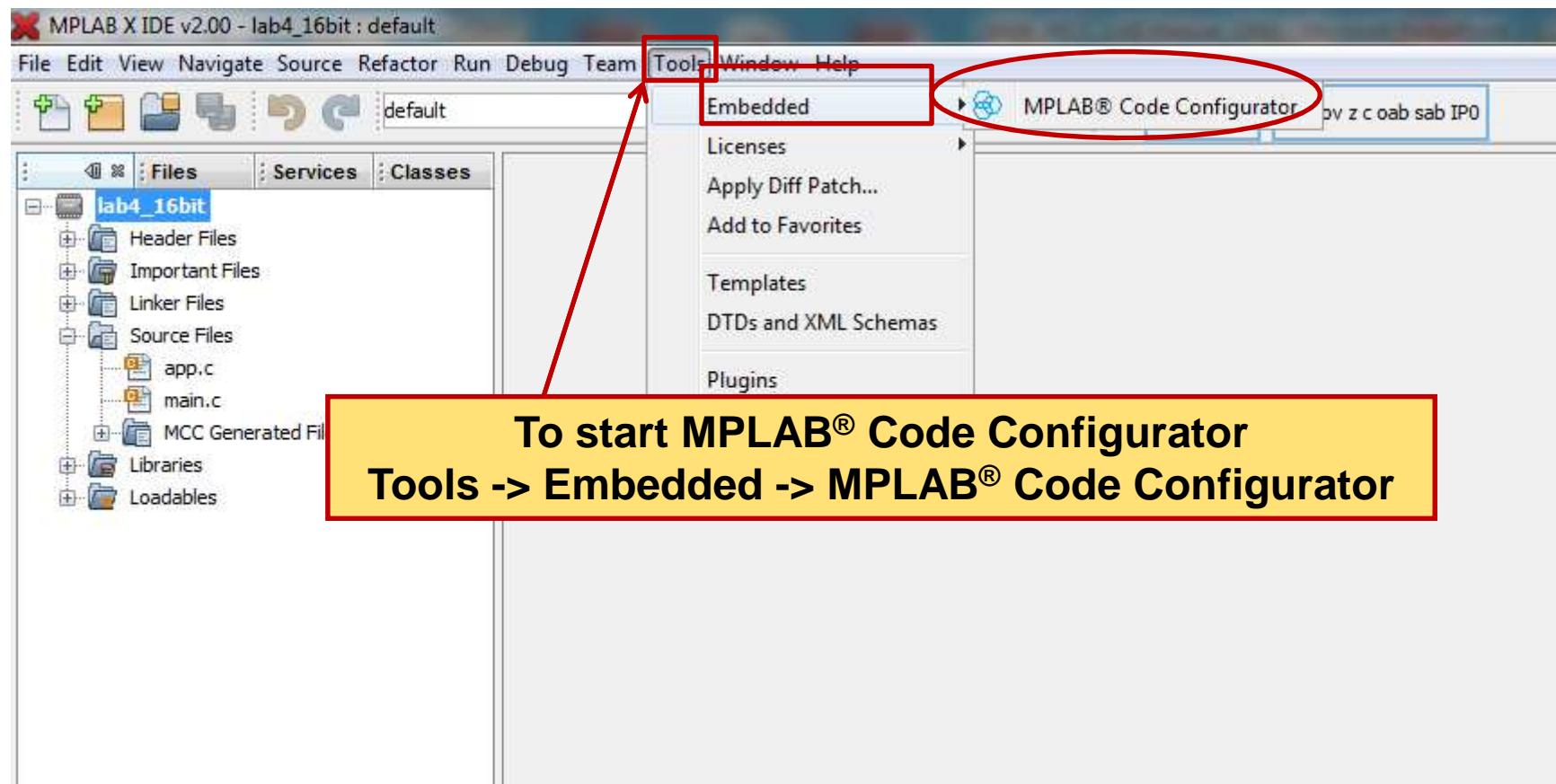
PIC24 Lab 4

- Open lab4_16bit.X



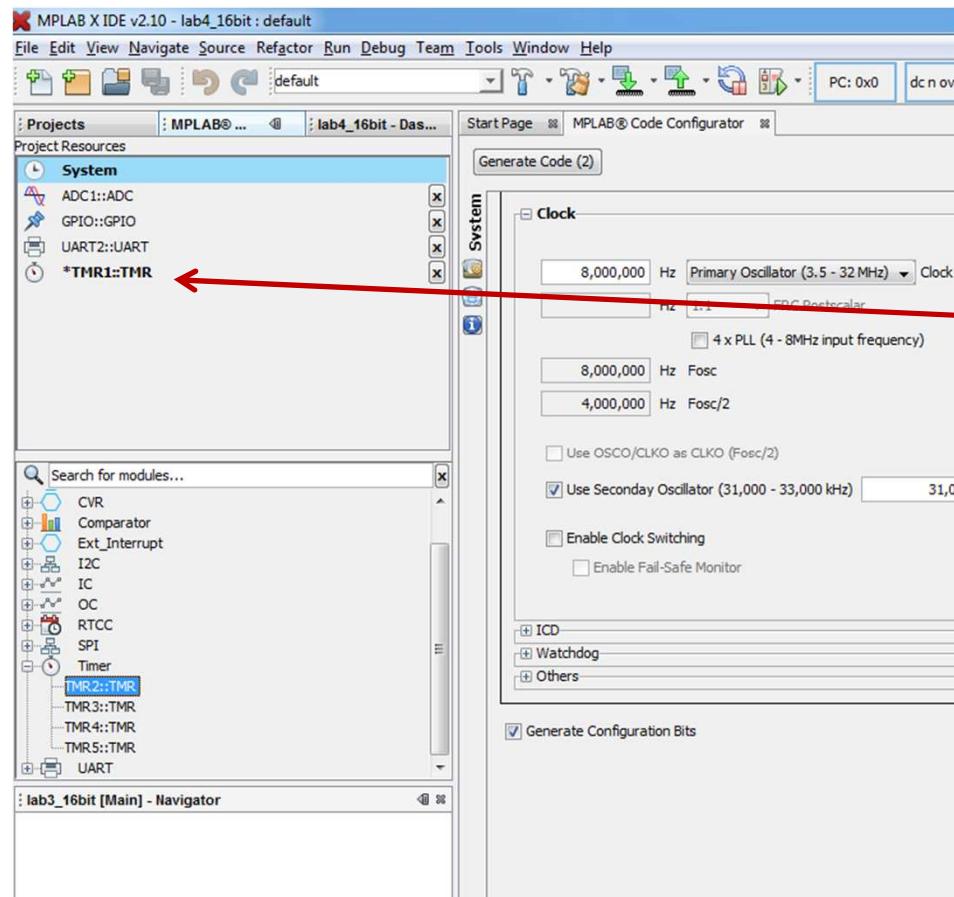
PIC24 Lab 4

- Start MPLAB® Code Configurator



PIC24 Lab 4

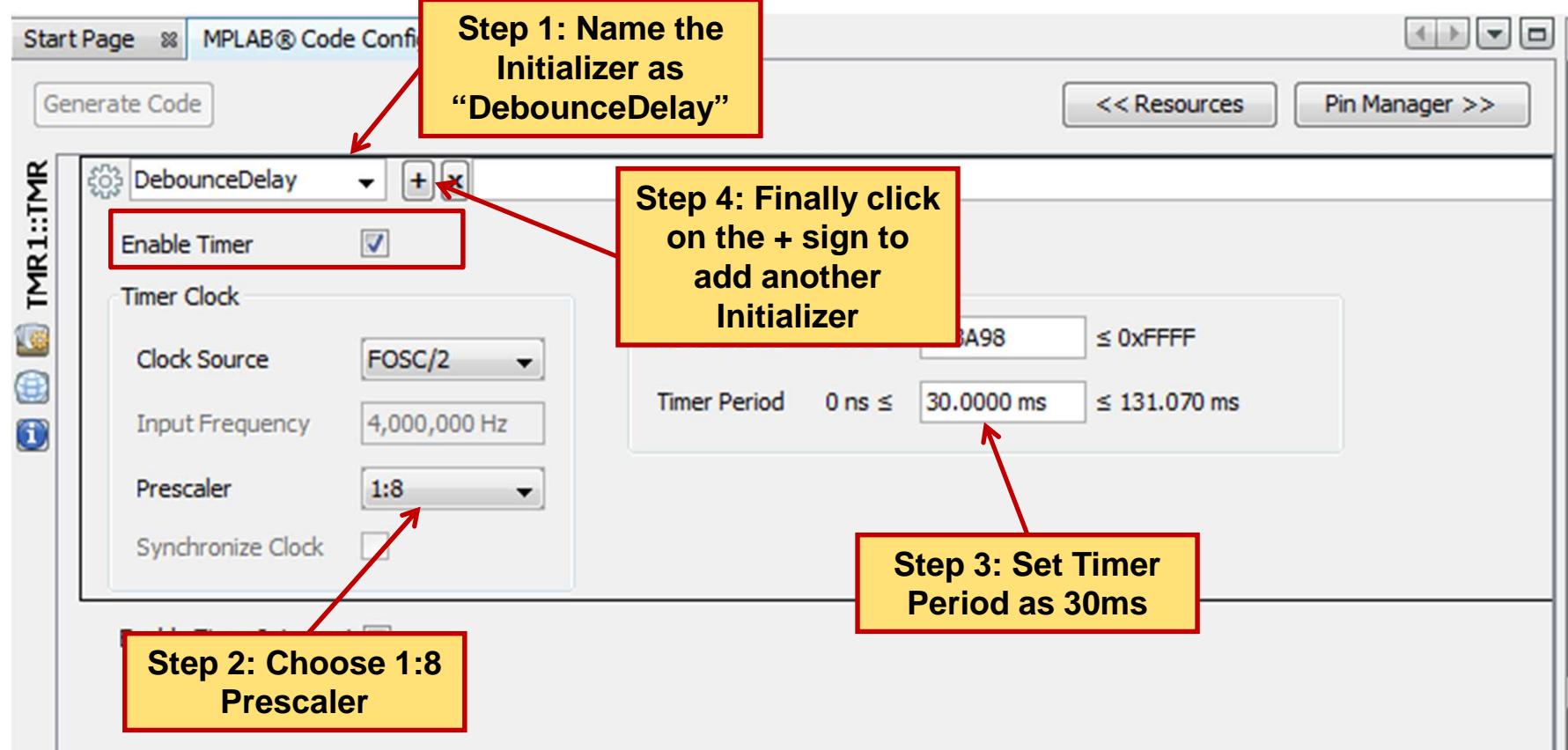
● Select TMR1



Select
TMR1::TMR from
Device Resources
and move it to
Project Resources.

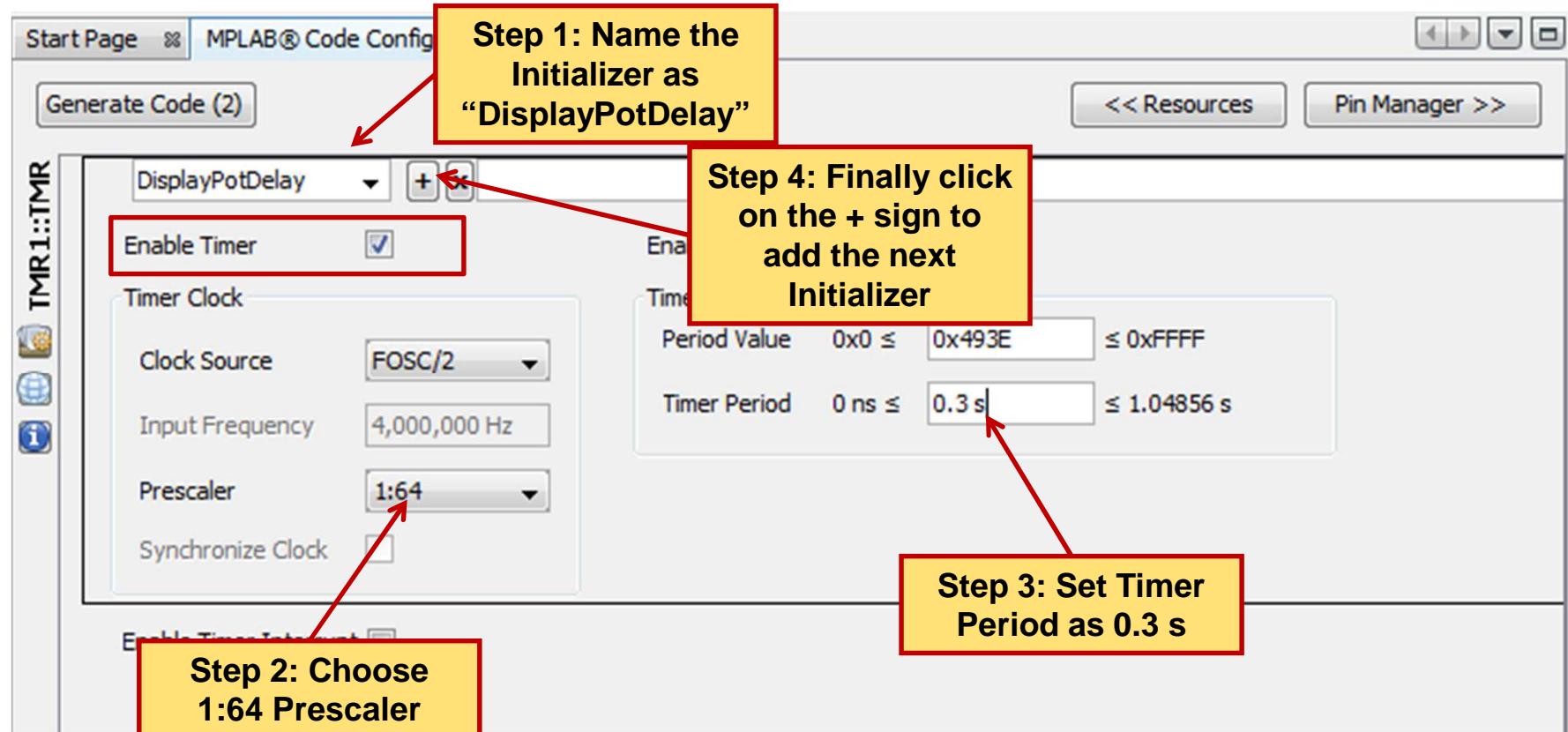
PIC24 Lab 4

- Add DebounceDelay Initializer



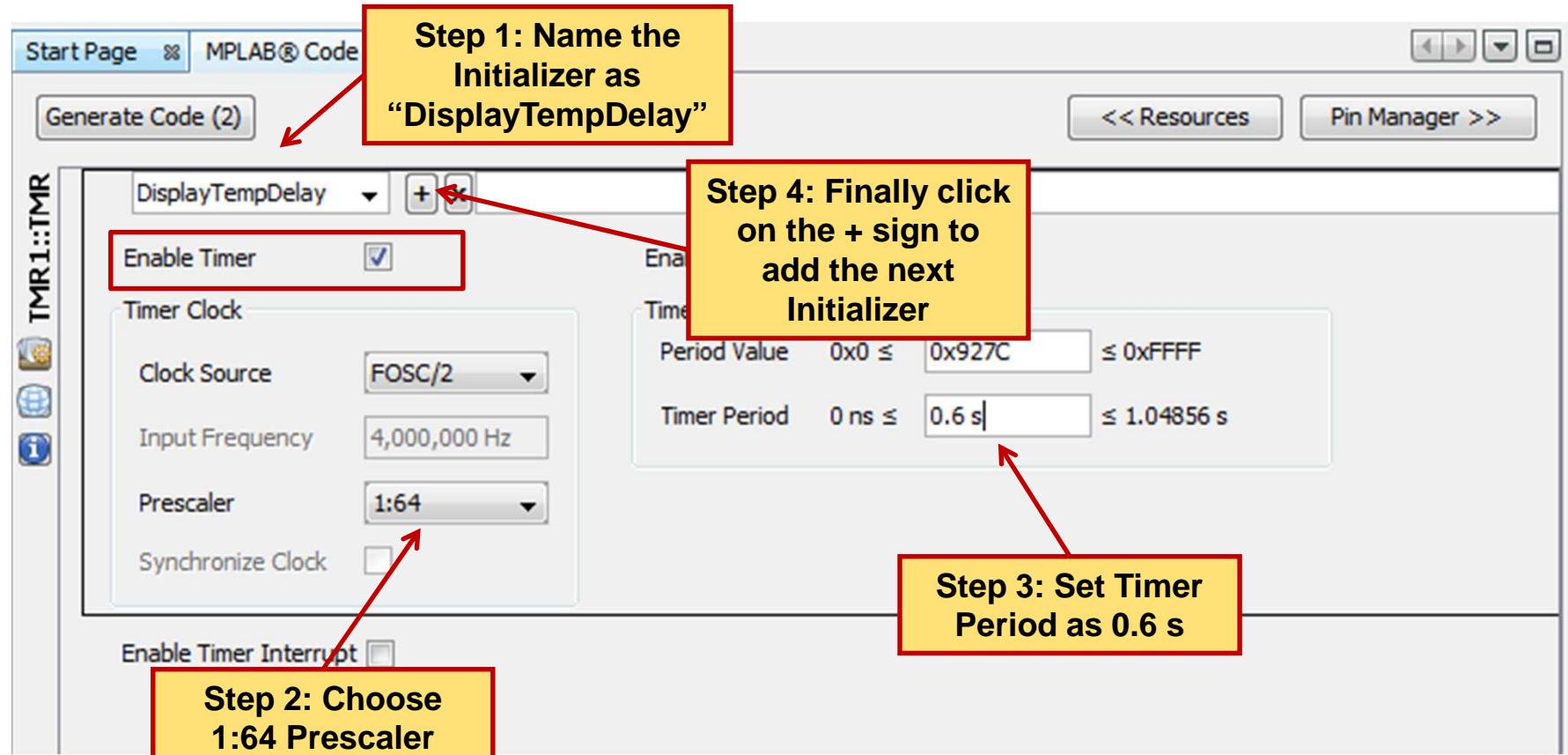
PIC24 Lab 4

● Add DisplayPotDelay Initializer



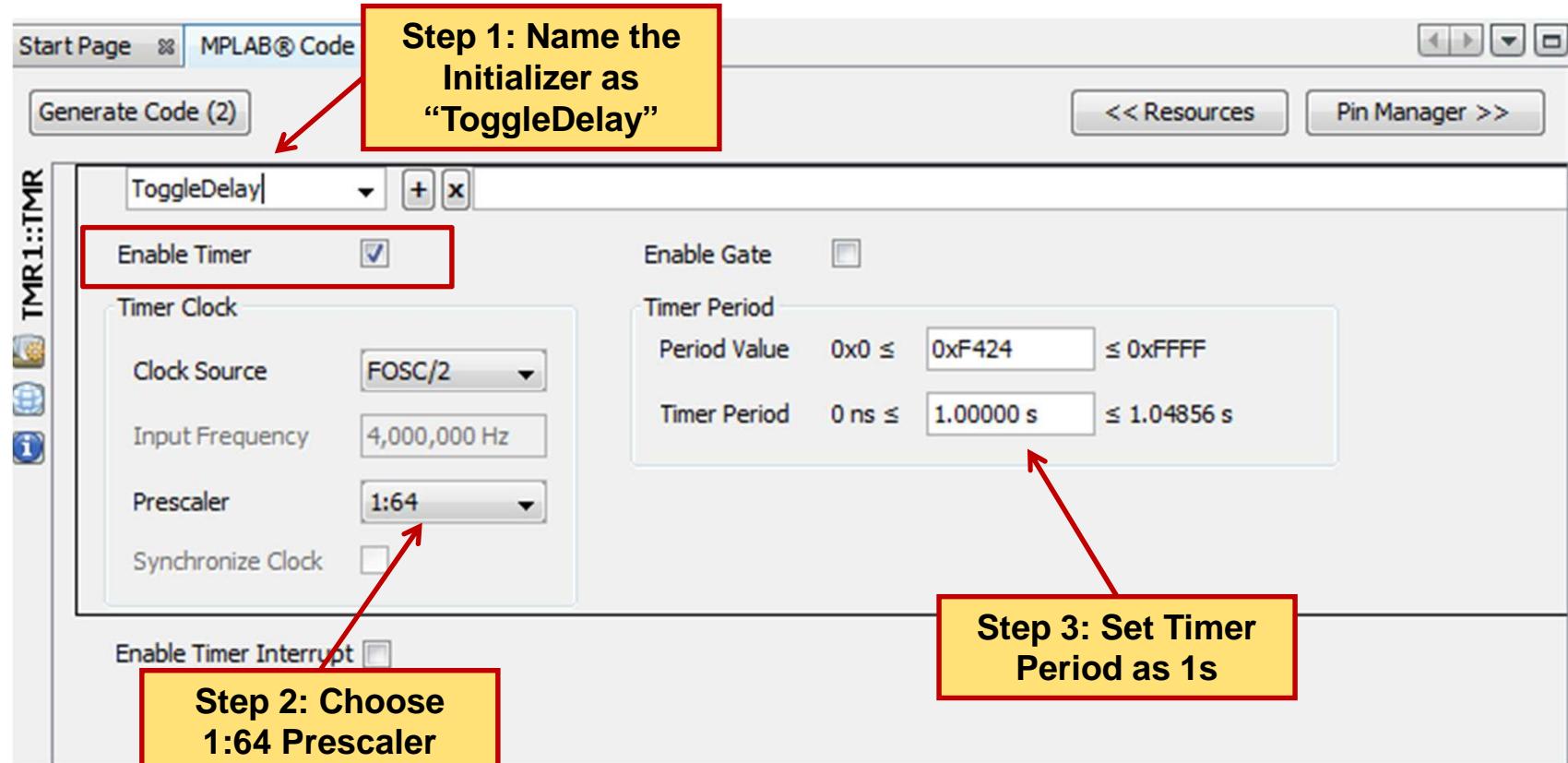
PIC24 Lab 4

● Add DisplayTempDelay Initializer



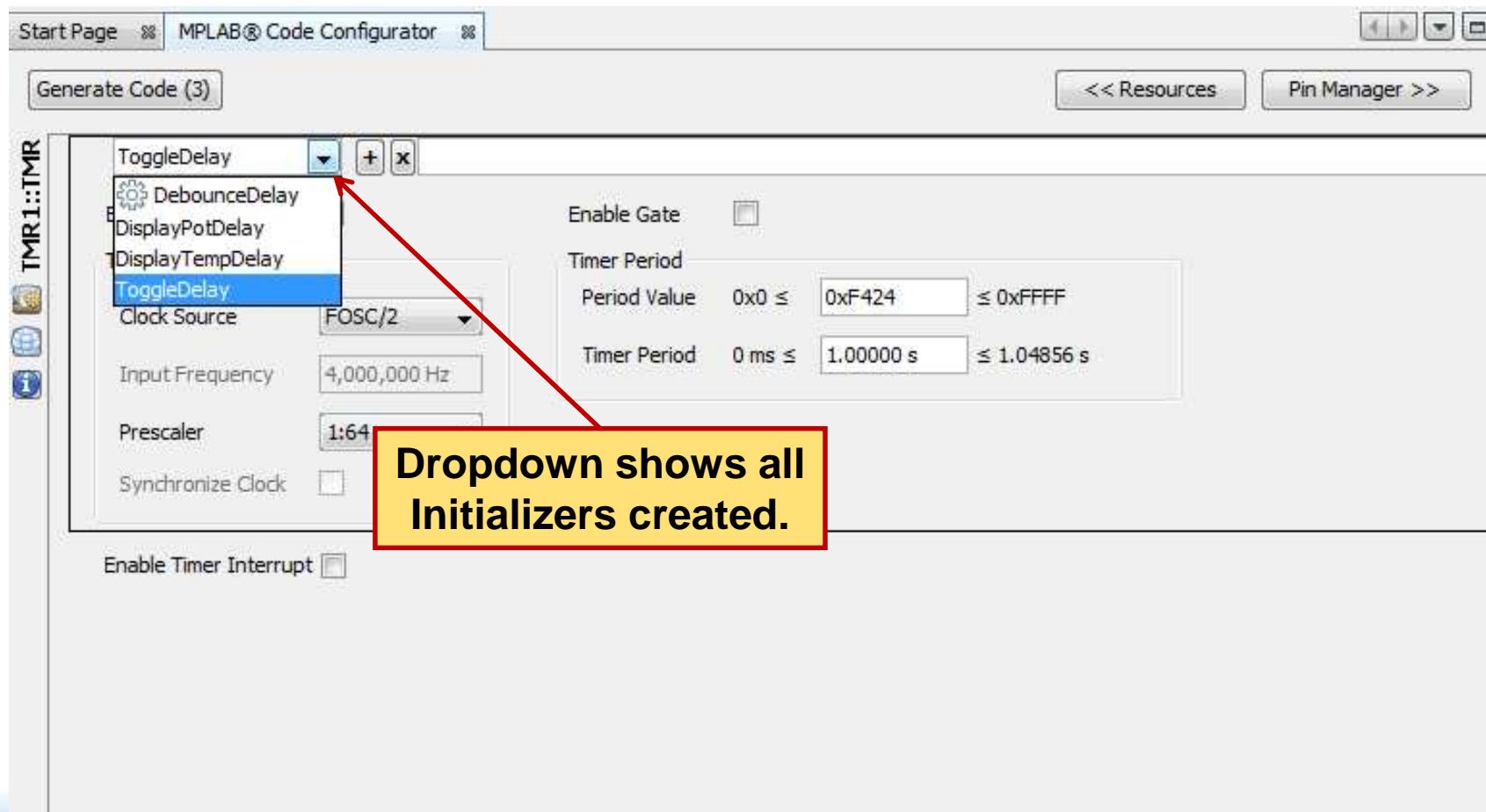
PIC24 Lab 4

- Add ToggleDelay Initializer



PIC24 Lab 4

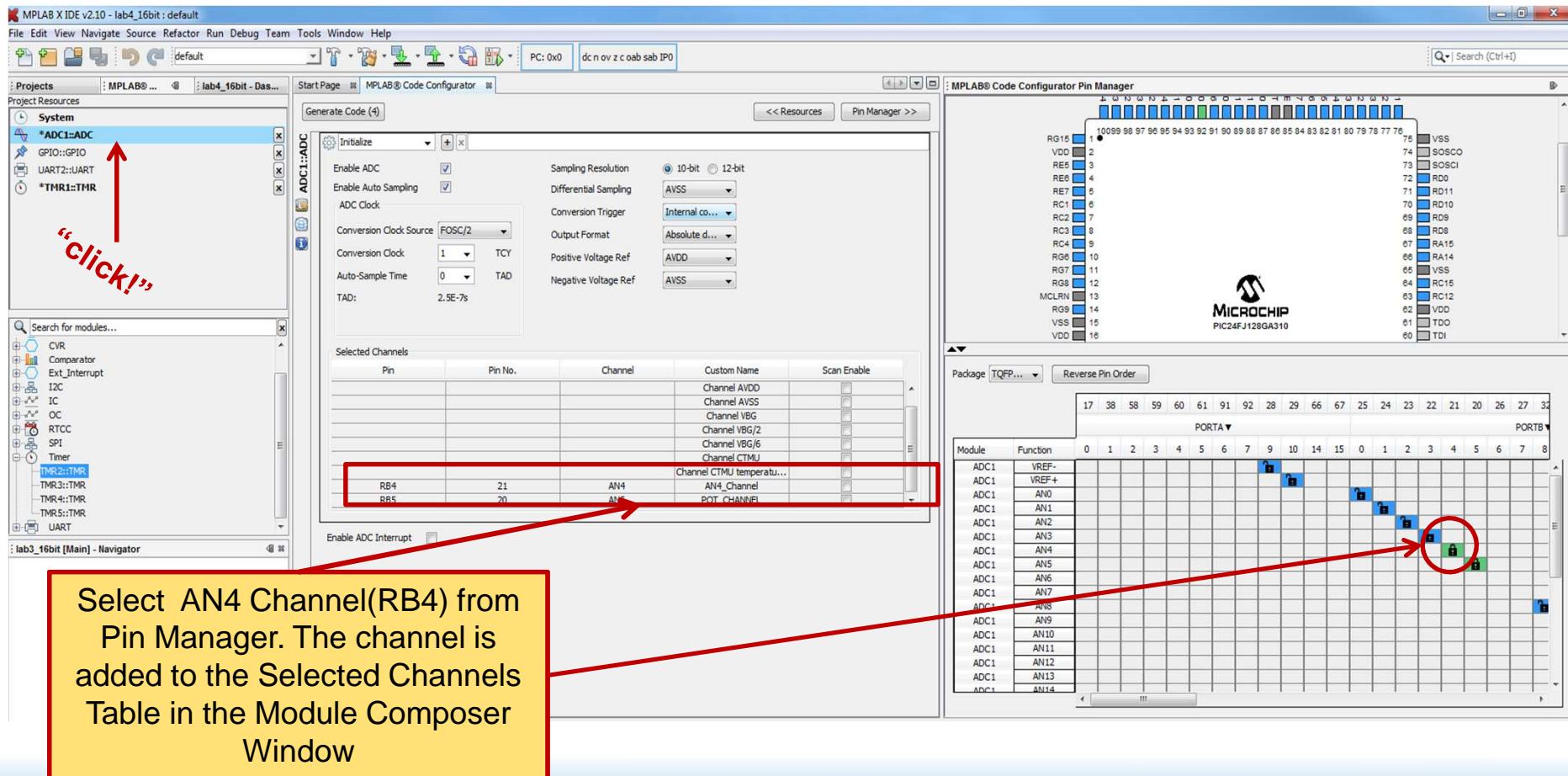
- Use dropdown to see all initializers added



Dropdown shows all
Initializers created.

PIC24 Lab 4

- Add AN4 Channel to ADC Configuration



PIC24 Lab 4

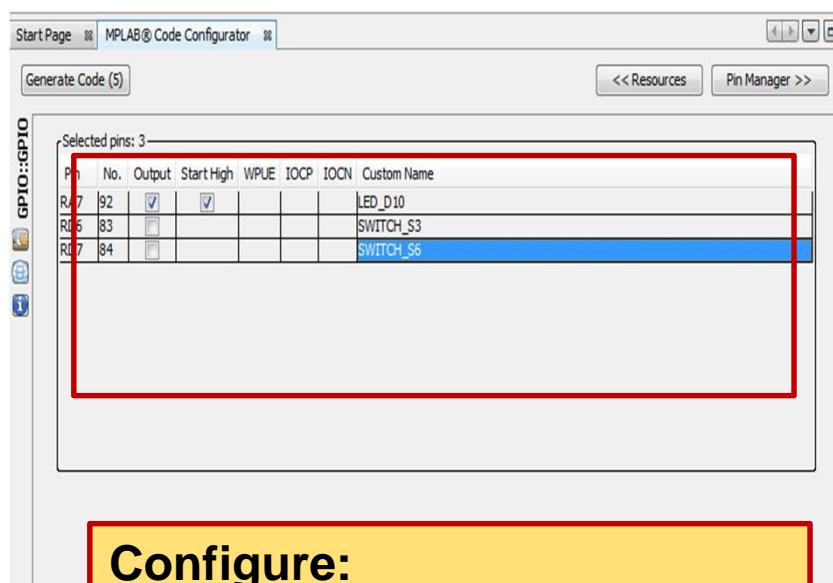
- Rename AN4_Channel as TEMP_CHANNEL

Selected Channels				
Pin	Pin No.	Channel	Custom Name	Scan Enal
			Channel AVDD	<input type="checkbox"/>
			Channel AVSS	<input type="checkbox"/>
			Channel VBG	<input type="checkbox"/>
			Channel VBG/2	<input type="checkbox"/>
			Channel VBG/6	<input type="checkbox"/>
			Channel CTMU	<input type="checkbox"/>
			Channel CTMU temperatu...	<input type="checkbox"/>
RB4	21	AN4	TEMP_CHANNEL	<input type="checkbox"/>
RB5	20	AN5	PC1_CHANNEL	<input type="checkbox"/>

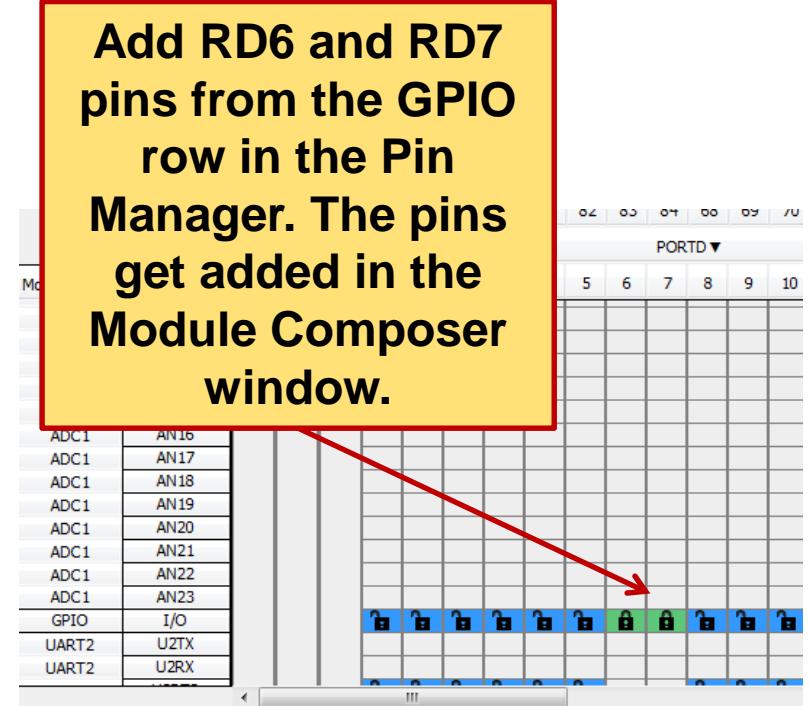
Rename AN4_Channel to
TEMP_CHANNEL

PIC24 Lab 4

- Configure the added GPIO Pins

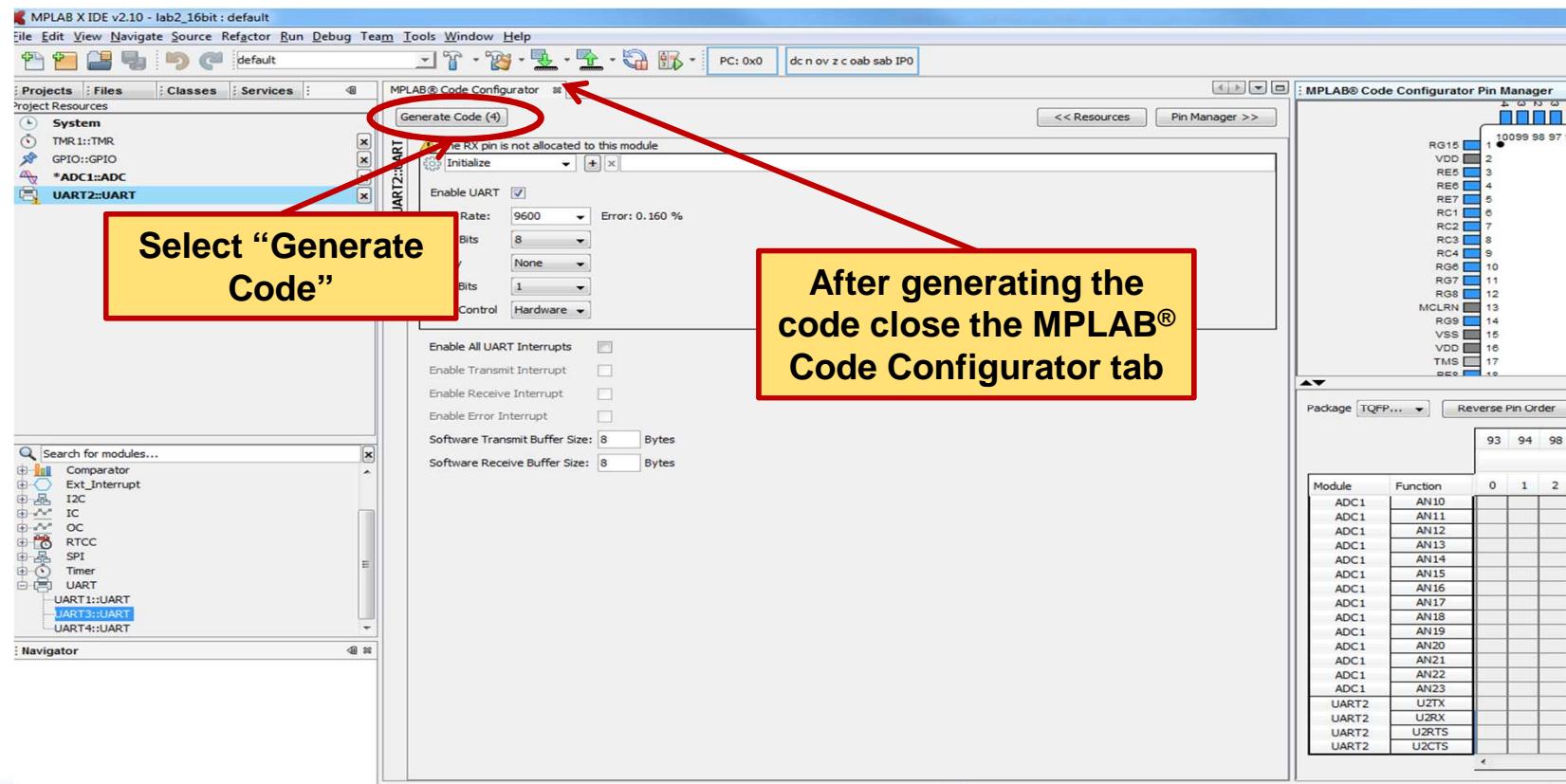


Configure:
→RD6 as input. Rename it as SWITCH_S3
→RD7 as input. Rename it as SWITCH_S6



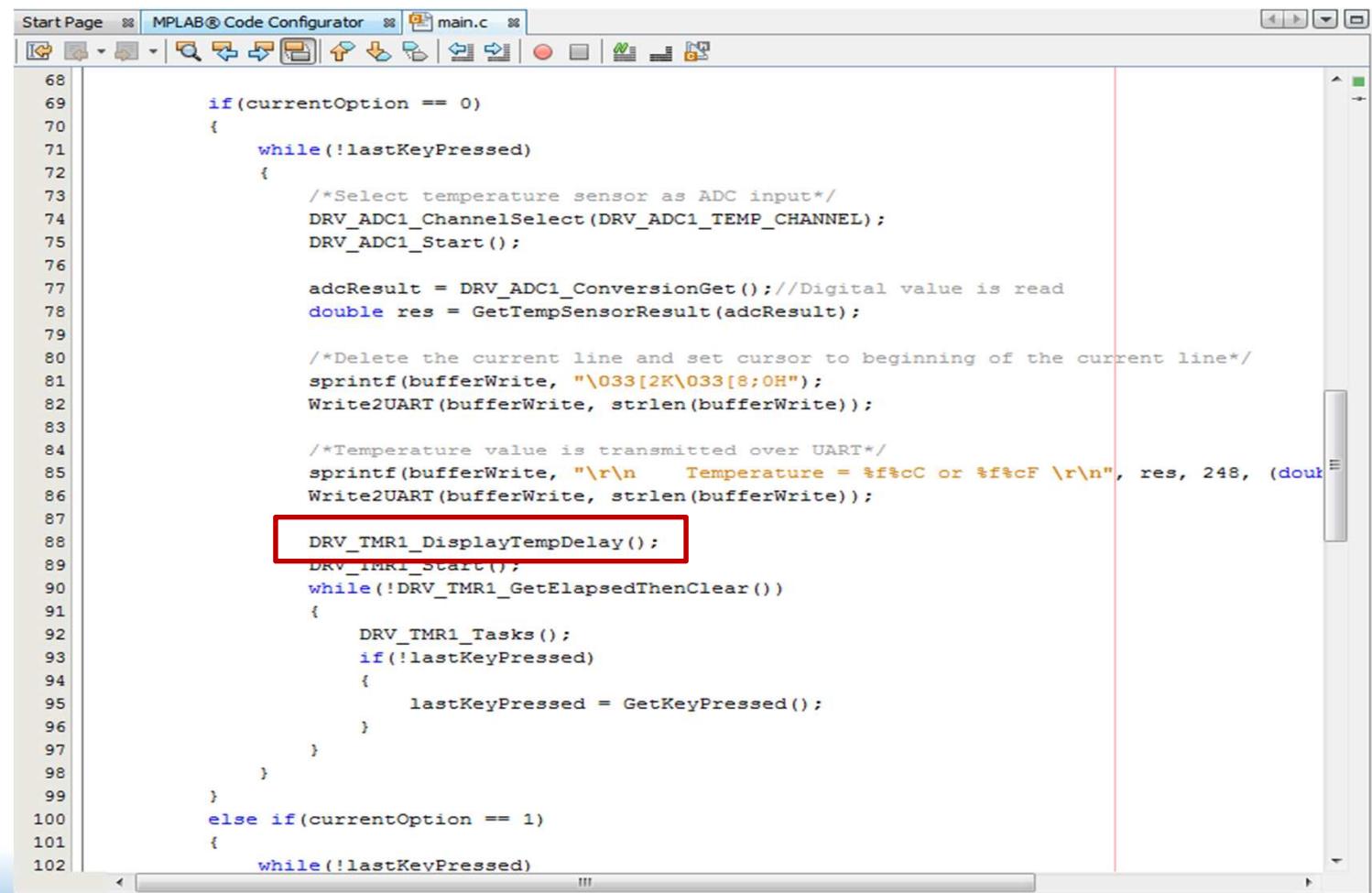
PIC24 Lab 4

- Generate the code and close MPLAB® Code Configurator



PIC24 Lab 4

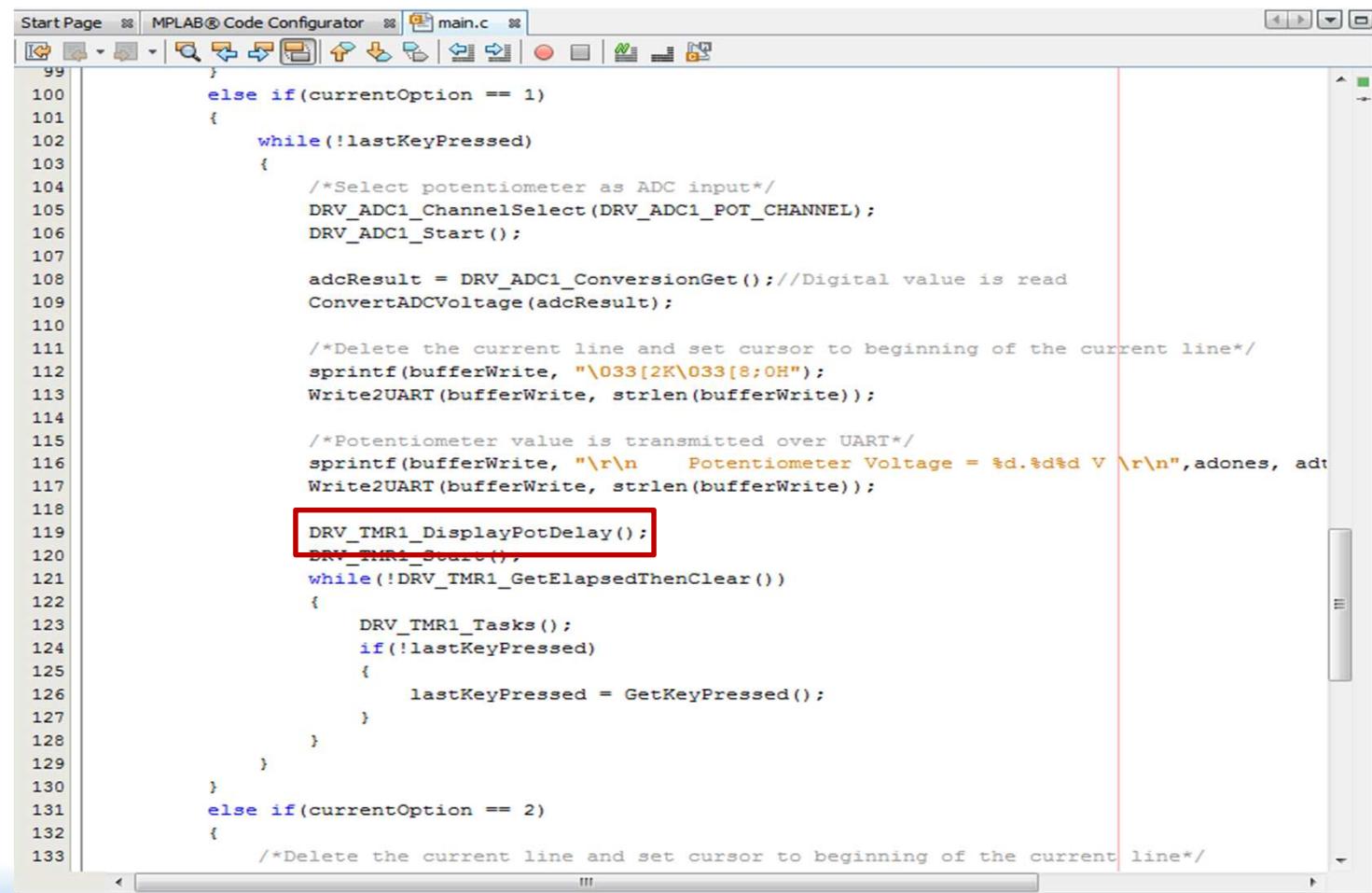
- Open main.c and add the following code



```
Start Page <> MPLAB® Code Configurator <> main.c <>
68     if(currentOption == 0)
69     {
70         while(!lastKeyPressed)
71         {
72             /*Select temperature sensor as ADC input*/
73             DRV_ADC1_ChannelSelect(DRV_ADC1_TEMP_CHANNEL);
74             DRV_ADC1_Start();
75
76             adcResult = DRV_ADC1_ConversionGet(); //Digital value is read
77             double res = GetTempSensorResult(adcResult);
78
79             /*Delete the current line and set cursor to beginning of the current line*/
80             sprintf(bufferWrite, "\033[2K\033[8;OH");
81             Write2UART(bufferWrite, strlen(bufferWrite));
82
83             /*Temperature value is transmitted over UART*/
84             sprintf(bufferWrite, "\r\n      Temperature = %f°C or %f°F \r\n", res, 248, (double)res);
85             Write2UART(bufferWrite, strlen(bufferWrite));
86
87             DRV_TMR1_DisplayTempDelay();
88             DRV_TMR1_Start();
89             while(!DRV_TMR1_GetElapsedThenClear())
90             {
91                 DRV_TMR1_Tasks();
92                 if(!lastKeyPressed)
93                 {
94                     lastKeyPressed = GetKeyPressed();
95                 }
96             }
97         }
98     }
99     else if(currentOption == 1)
100    {
101        while(!lastKeyPressed)
```

PIC24 Lab 4

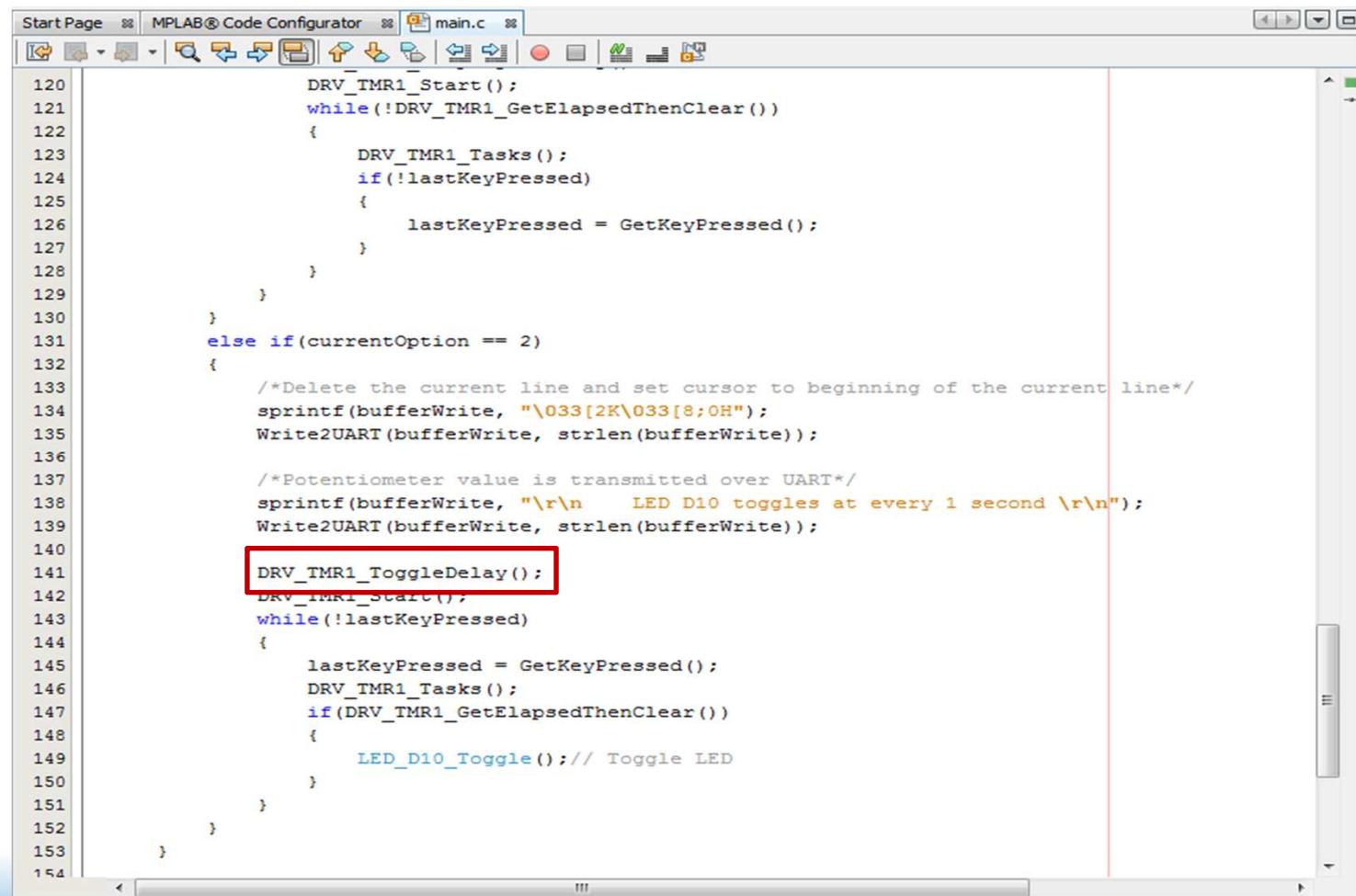
- Open main.c and add the following code



```
Start Page  MPLAB® Code Configurator  main.c  99
100         }
101     else if(currentOption == 1)
102     {
103         while(!lastKeyPressed)
104         {
105             /*Select potentiometer as ADC input*/
106             DRV_ADC1_ChannelSelect(DRV_ADC1_POT_CHANNEL);
107             DRV_ADC1_Start();
108
109             adcResult = DRV_ADC1_ConversionGet(); //Digital value is read
110             ConvertADCVoltage(adcResult);
111
112             /*Delete the current line and set cursor to beginning of the current line*/
113             sprintf(bufferWrite, "\033[2K\033[8;OH");
114             Write2UART(bufferWrite, strlen(bufferWrite));
115
116             /*Potentiometer value is transmitted over UART*/
117             sprintf(bufferWrite, "\r\n      Potentiometer Voltage = %d.%d%d V \r\n",adones, adtens, adones);
118             Write2UART(bufferWrite, strlen(bufferWrite));
119
120             DRV_TMR1_DisplayPotDelay();
121             DRV_TMR1_Start();
122             while(!DRV_TMR1_GetElapsedThenClear())
123             {
124                 DRV_TMR1_Tasks();
125                 if(!lastKeyPressed)
126                 {
127                     lastKeyPressed = GetKeyPressed();
128                 }
129             }
130         }
131     else if(currentOption == 2)
132     {
133         /*Delete the current line and set cursor to beginning of the current line*/
```

PIC24 Lab 4

- Open main.c and add the following code



```
Start Page  MPLAB® Code Configurator  main.c
120     DRV_TMR1_Start();
121     while(!DRV_TMR1_GetElapsedThenClear())
122     {
123         DRV_TMR1_Tasks();
124         if(!lastKeyPressed)
125         {
126             lastKeyPressed = GetKeyPressed();
127         }
128     }
129 }
130 }
131 else if(currentOption == 2)
132 {
133     /*Delete the current line and set cursor to beginning of the current line*/
134     sprintf(bufferWrite, "\033[2K\033[8;OH");
135     Write2UART(bufferWrite, strlen(bufferWrite));
136
137     /*Potentiometer value is transmitted over UART*/
138     sprintf(bufferWrite, "\r\n      LED D10 toggles at every 1 second \r\n");
139     Write2UART(bufferWrite, strlen(bufferWrite));
140
141     DRV_TMR1_ToggleDelay(); // Add this line
142     DRV_TMR1_Start();
143     while(!lastKeyPressed)
144     {
145         lastKeyPressed = GetKeyPressed();
146         DRV_TMR1_Tasks();
147         if(DRV_TMR1_GetElapsedThenClear())
148         {
149             LED_D10_Toggle(); // Toggle LED
150         }
151     }
152 }
153 }
154 }
```

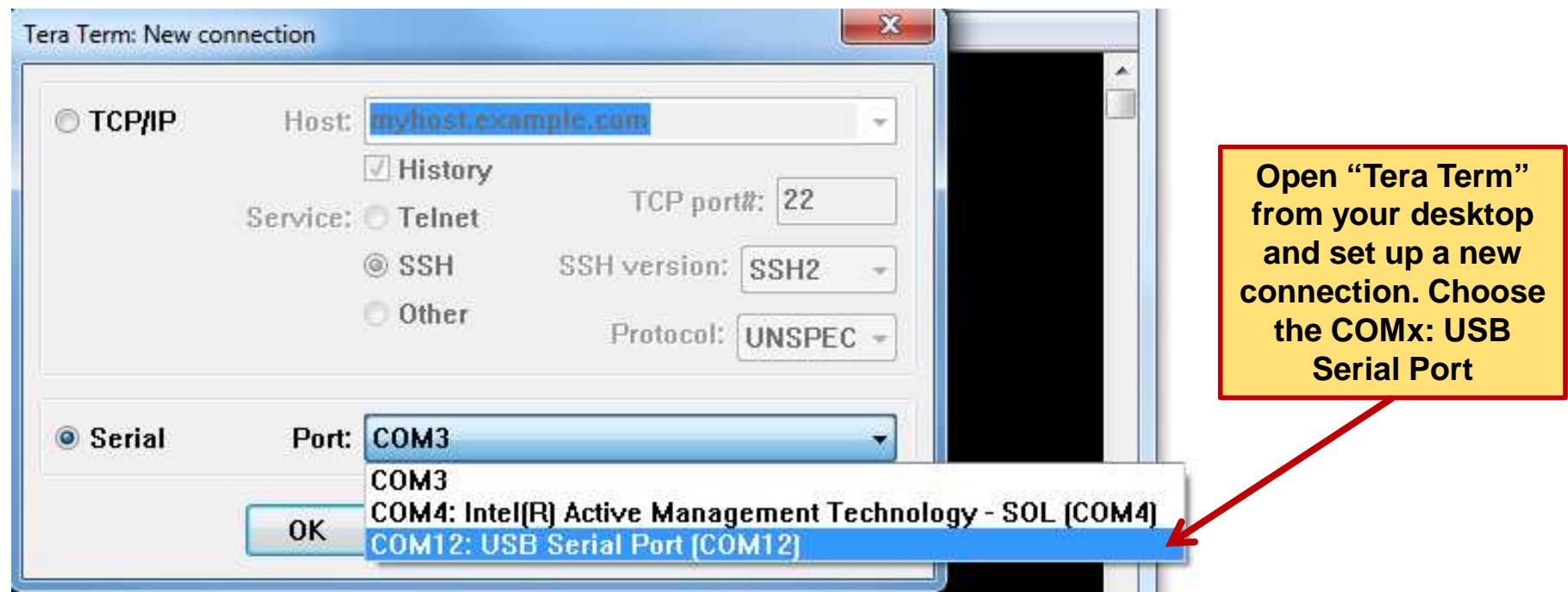
PIC24 Lab 4

- “Make and Program” the project



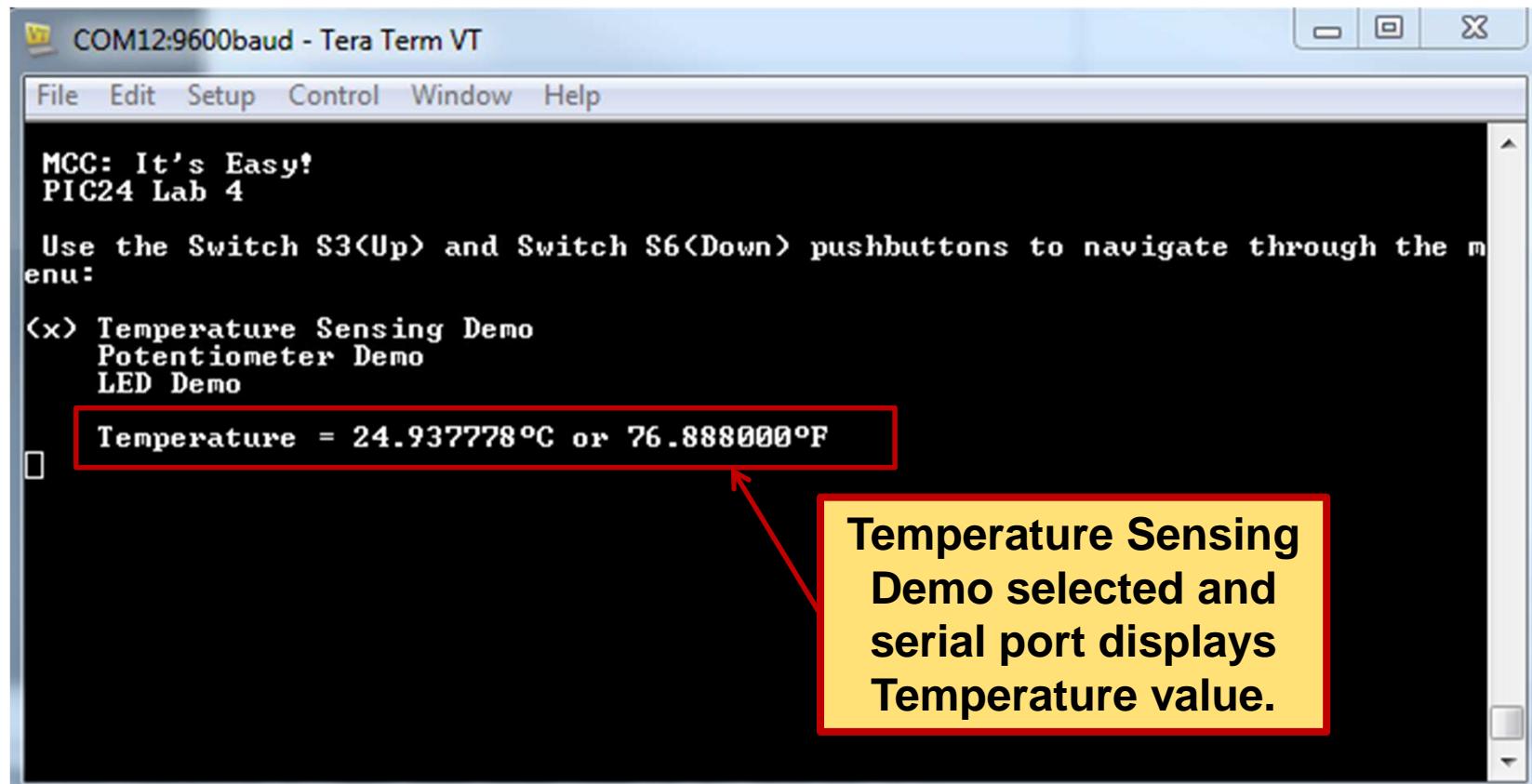
PIC24 Lab 4

- Start Tera Term
- Select the USB Serial Port



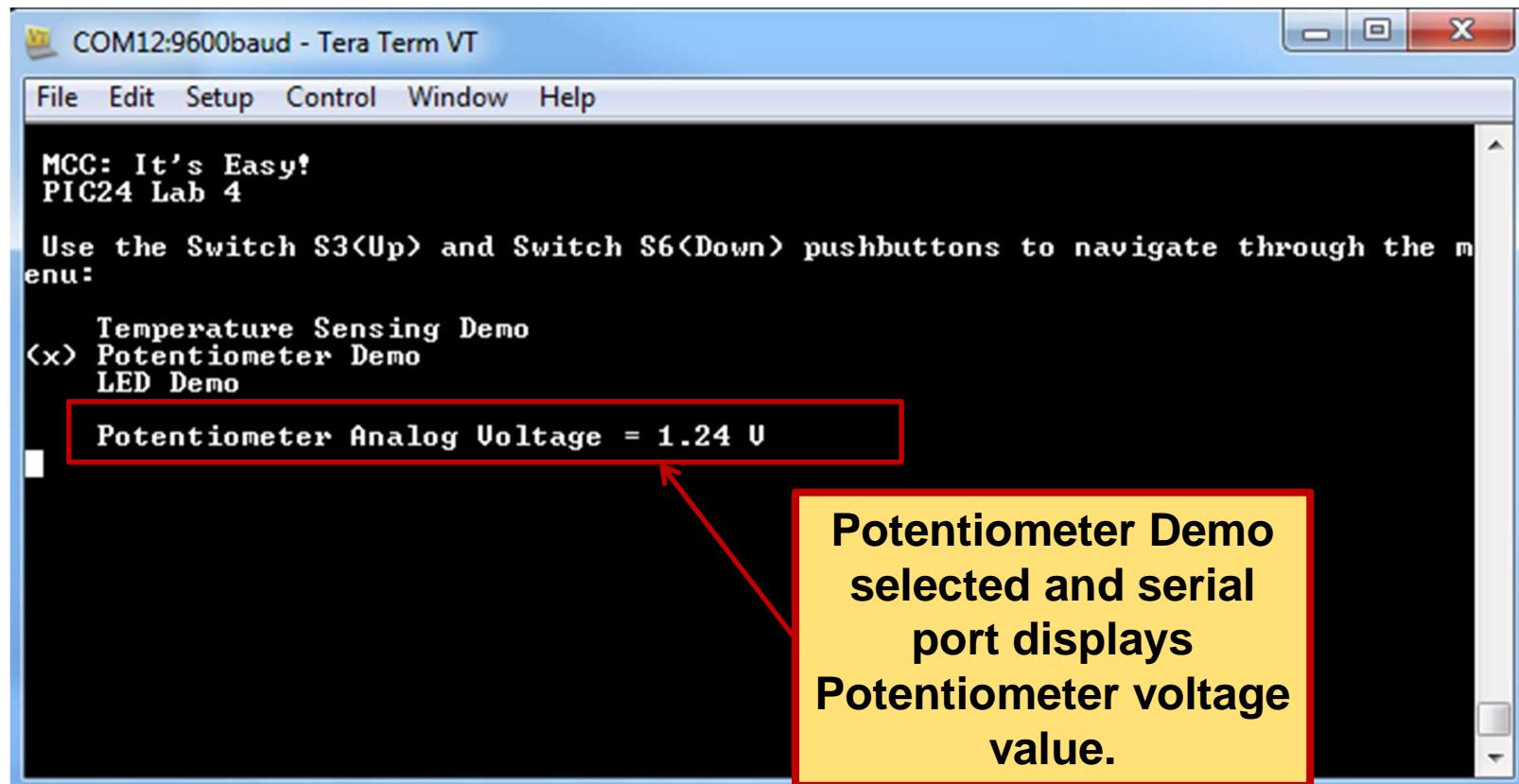
PIC24 Lab4

Scroll through the Menu using Switch S3 and Switch S6



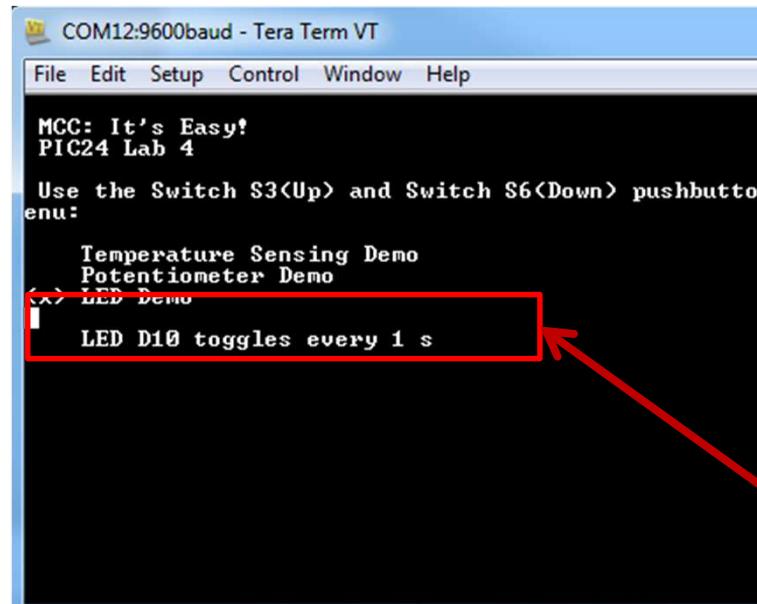
PIC24 Lab 4

Use Switch S6 to choose Potentiometer Demo



PIC24 Lab 4

Use Switch S6 to choose LED Demo



COM12:9600baud - Tera Term VT

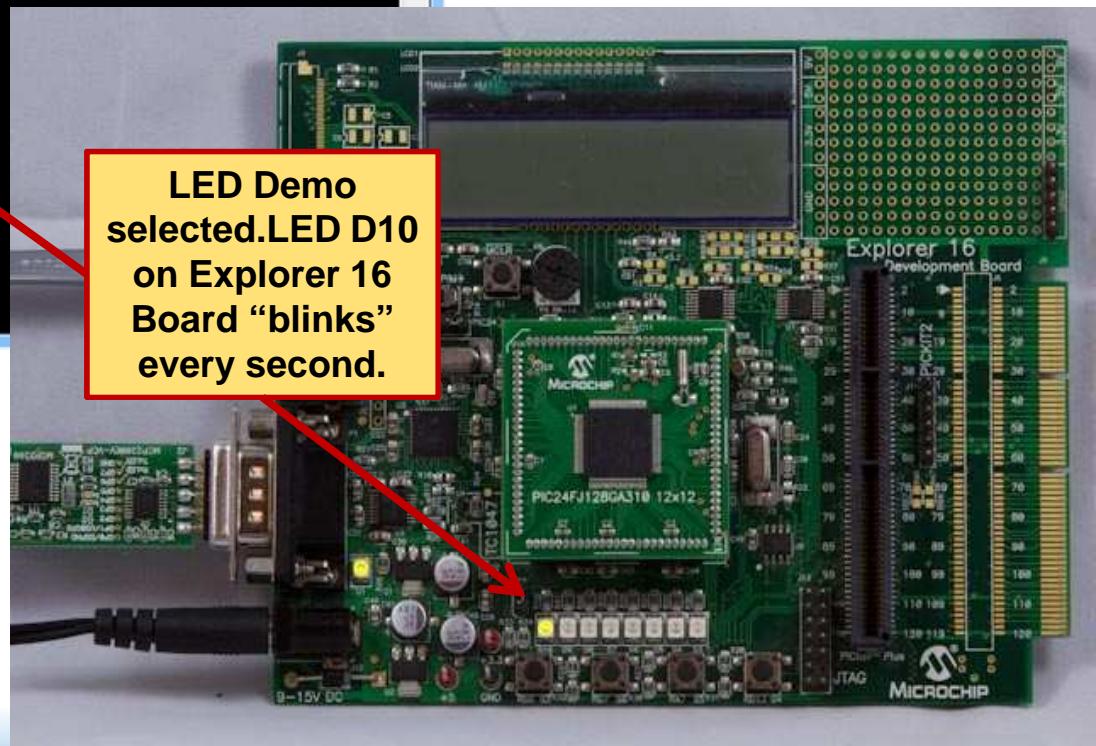
File Edit Setup Control Window Help

MCC: It's Easy!
PIC24 Lab 4

Use the Switch S3<Up> and Switch S6<Down> pushbuttons to navigate through the menu:

Temperature Sensing Demo
Potentiometer Demo
(x) LED Demo

LED D10 toggles every 1 s



LED Demo selected. LED D10 on Explorer 16 Board “blinks” every second.

PIC24 Lab 4 Summary

- We configured multiple initializers in the Timer module
- We configured pushbuttons to navigate through a menu of commands and perform different tasks

PIC24 Lab 5: I²C Communication

PIC24 Lab 5 Objectives

- **Communication using I²C**
- **Configure and use I²C to read and write data into I²C EEPROM Slave**
- **Configure UART to receive feedback messages**

PIC24 Lab 5 Overview

- Close any files open in the editor, and close all MPLAB® X IDE Projects
- Open lab5_16bit.X
- Install the I²C EEPROM PIM
- Setup I²C1 Master
 - Setup Baud Rate Generator Value as 5

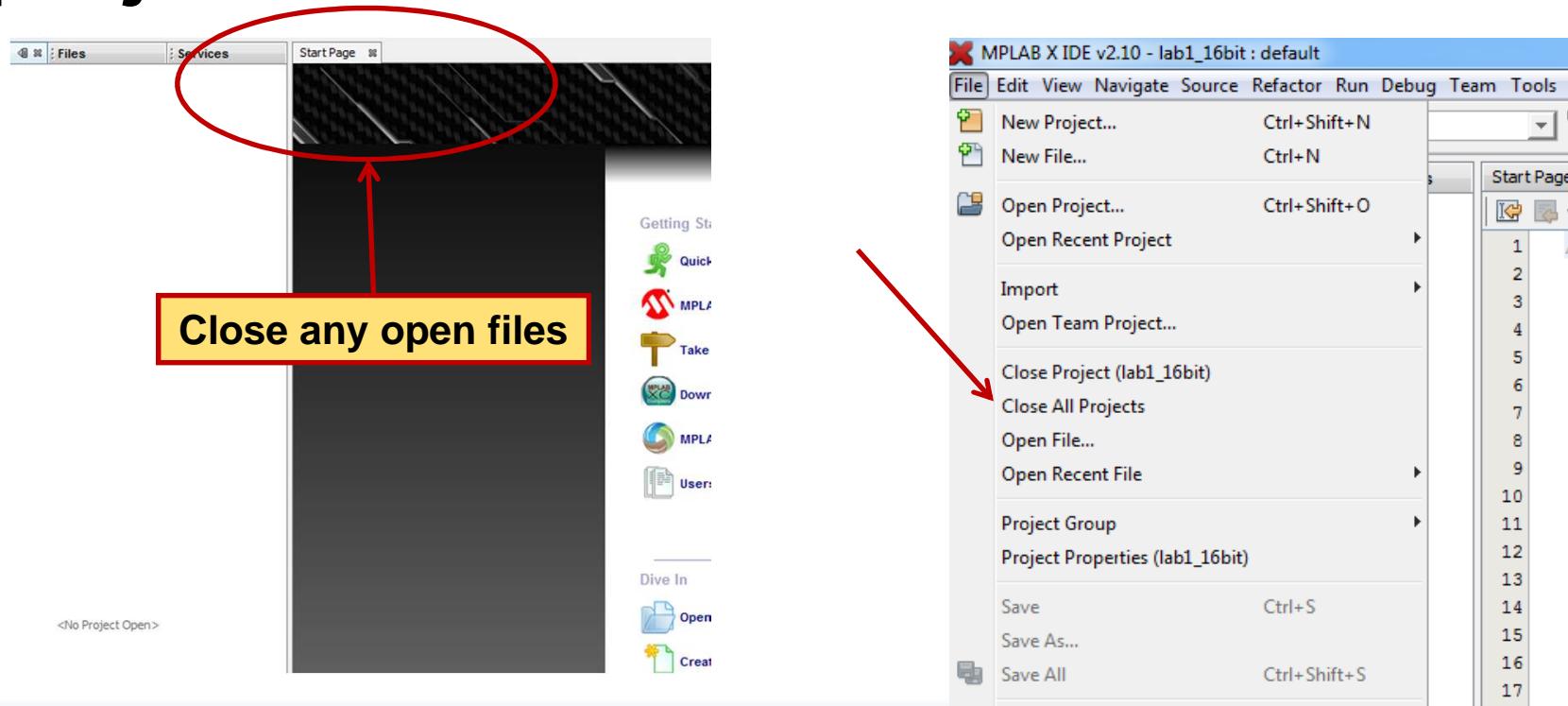
PIC24 Lab 5

- Remove power from the Explorer 16 Board
- Install the I²C EEPROM
- Reconnect the Power to the board



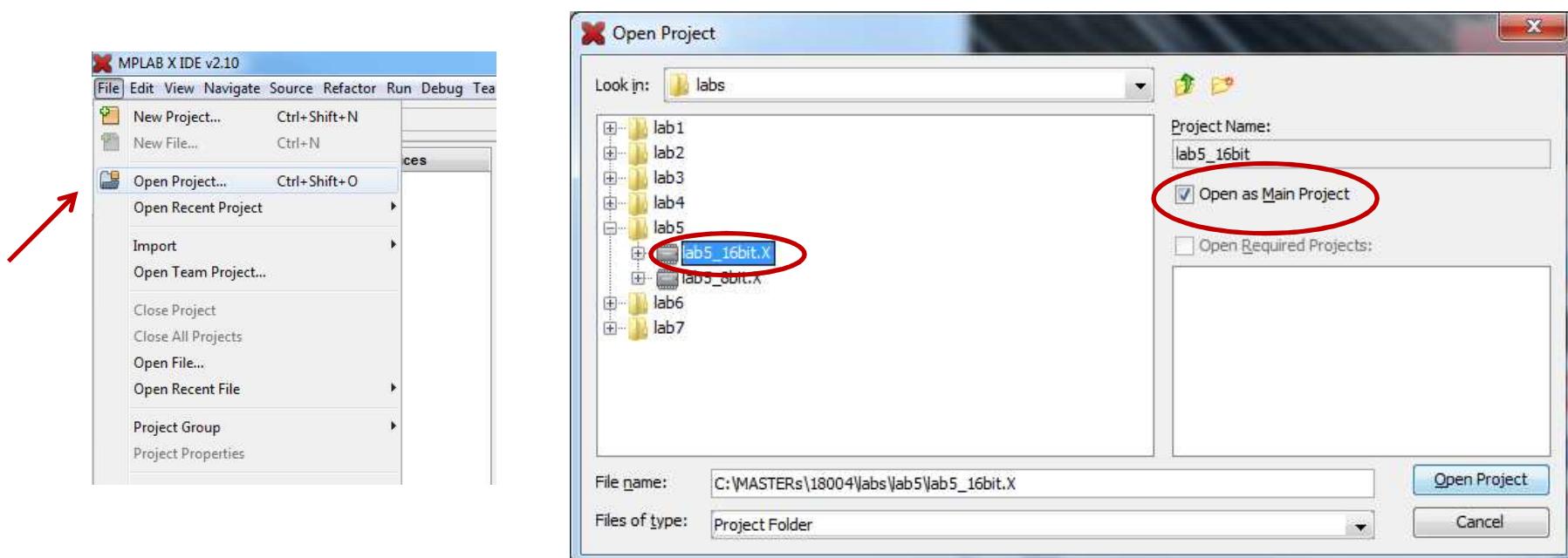
PIC24 Lab 5

- Close any open files in the editor window, and all open MPLAB® X IDE projects



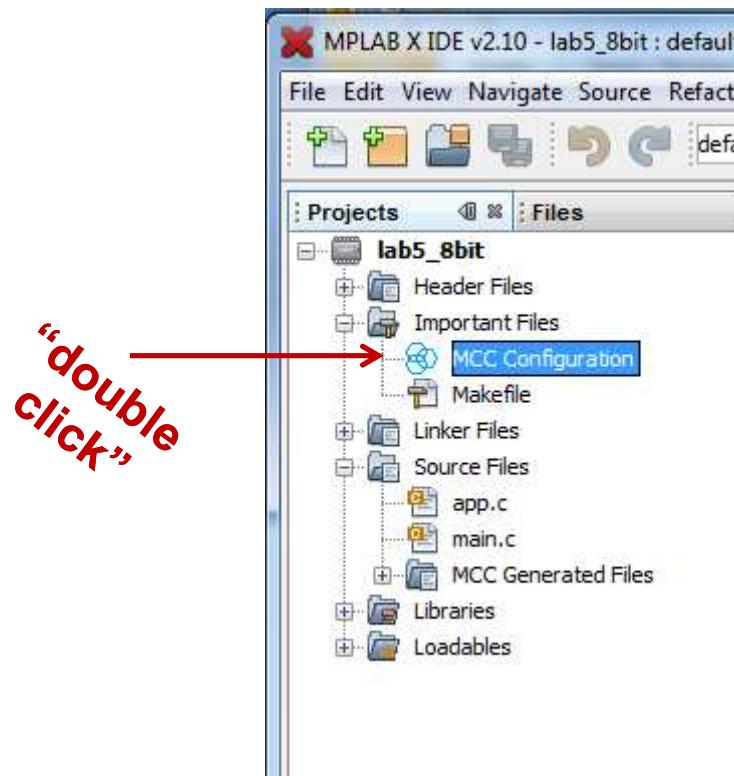
PIC24 Lab 5

- Open lab5_16bit.X



PIC24 Lab 5

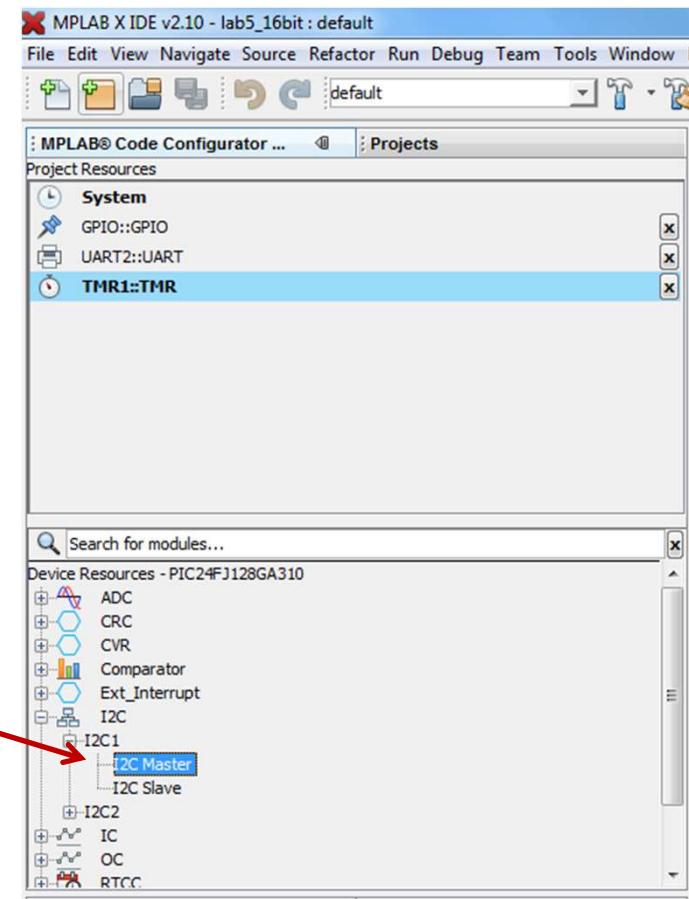
- Start MPLAB® Code Configurator



PIC24 Lab 5

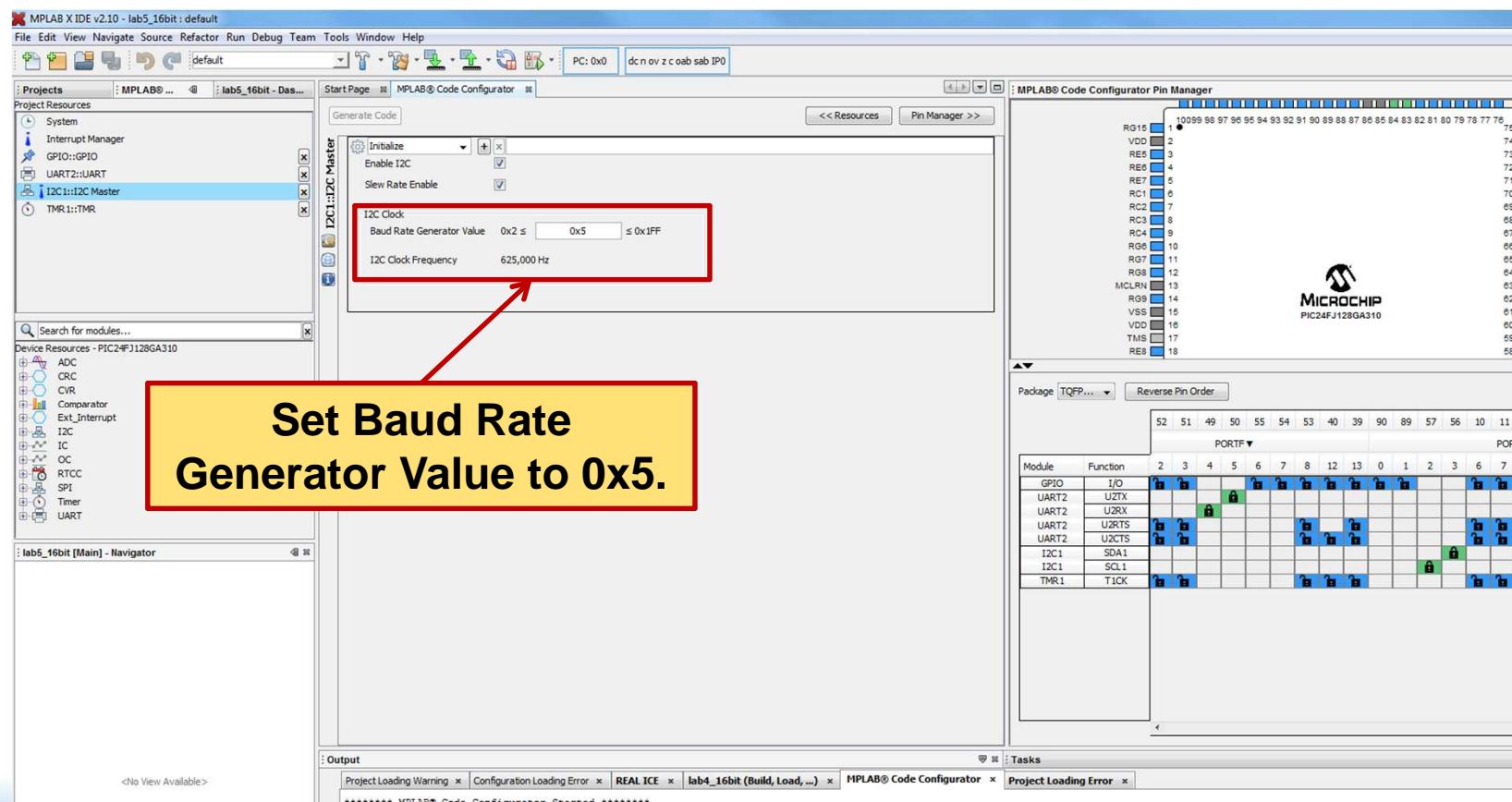
- Select I²C1 Master from Device Resources

Double Click on I²C1 and Double click on I²C Master



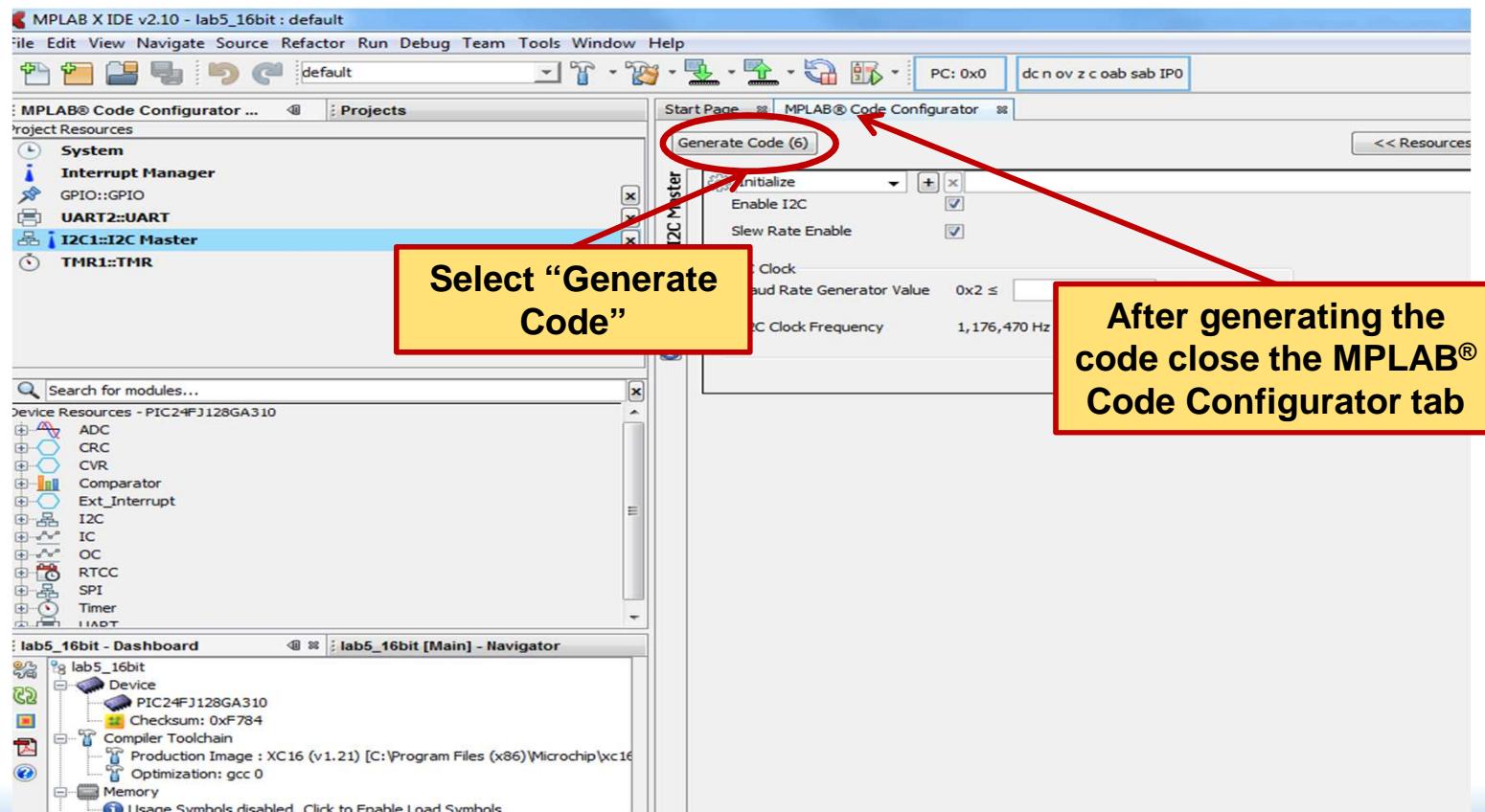
PIC24 Lab 5

- Configure the I²C module



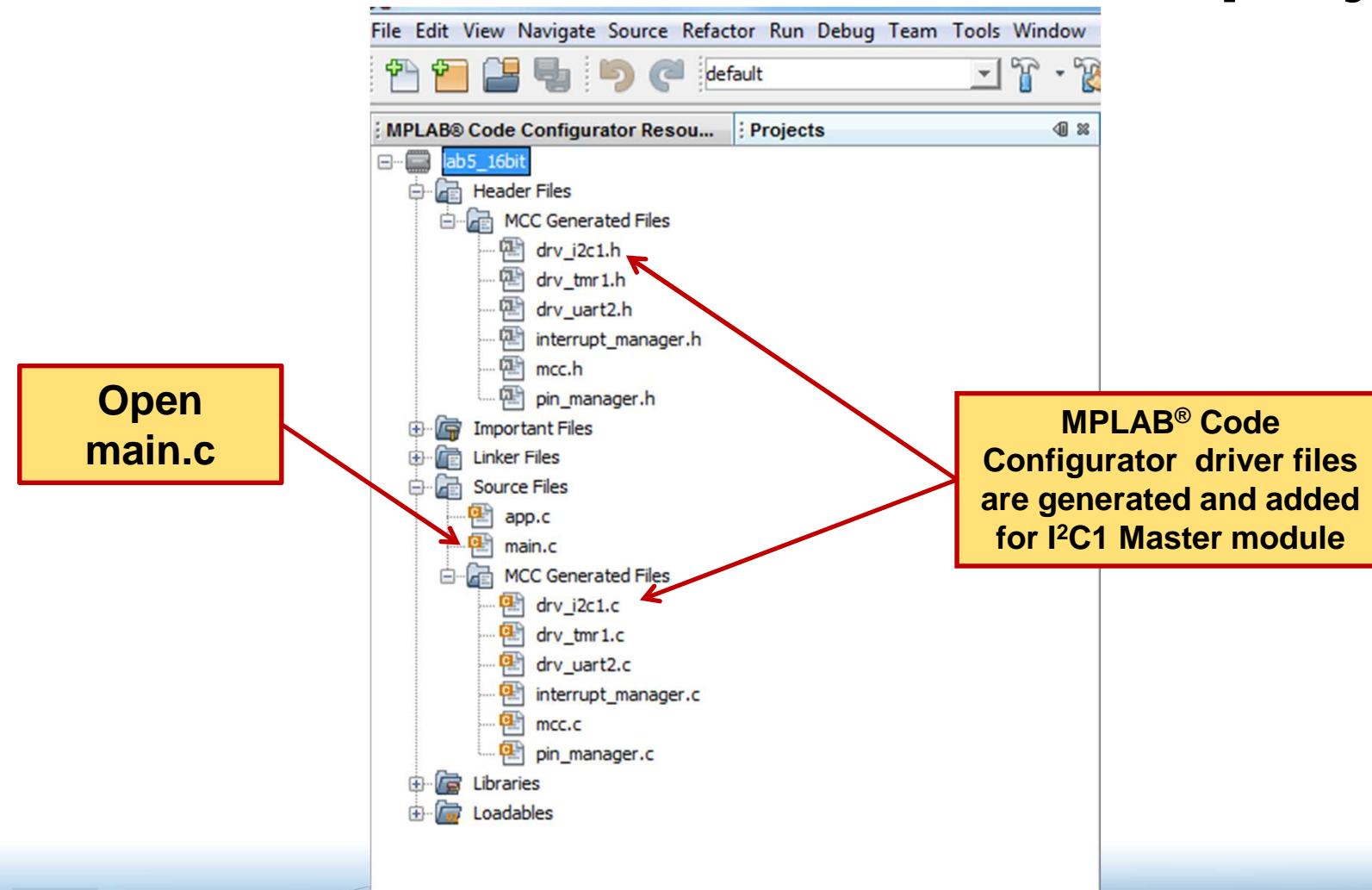
PIC24 Lab 5

- Generate the code and close MPLAB® Code Configurator



PIC24 Lab 5

- Generated files added to the project



PIC24 Lab 5

- Open main.c and fill in code

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

    if(1 == GetKeyPressed())
    {
        pressedKey = 0;
        //Write Data to EEPROM
        DRV_I2C1_MasterWrite(eepromWriteBuffer,sizeof(eepromWriteBuffer),I2C_SLAVE_ADDRESS,&i2cStatus);
        //delay to complete I2C Write
        DRV_TMR1_EepromDelay();
        while(!DRV_TMR1_GetElapsedThenClear())
        {
            DRV_TMR1_Tasks();
        }
        DRV_TMR1_Stop();

        sprintf(bufferWrite, "\r\nEEPROM Write was executed\r\n");
        Write2UART(bufferWrite, strlen(bufferWrite));
    }
    else if(2 == GetKeyPressed())
}
```

PIC24 Lab 5

- Open main.c and fill in code

```
    Write2UART(DUPLICATEDEVICE, strlen(DUPLICATEDEVICE));
}

else if(2 == GetKeyPressed())
{
    pressedKey = 0;
    //Put PC back to EEPROM Start Address Location
    DRV_I2C1_MasterWrite(eepromWriteBuffer,NUM_ADDRESS_BYTES,I2C_SLAVE_ADDRESS,&i2cStatus);
    //I2C Communication delay
    DRV_TMR1_EepromDelay();
    while(!DRV_TMR1_GetElapsedThenClear())
    {
        DRV_TMR1_Tasks();
    }
    DRV_TMR1_Stop();

    //Read data from EEPROM
    DRV_I2C1_MasterRead(eepromReadBuffer,sizeof(eepromReadBuffer),I2C_SLAVE_ADDRESS, &i2cStatus);
    sprintf(bufferWrite, "\r\nEEPROM Read was executed\r\n");
    Write2UART(bufferWrite, strlen(bufferWrite));

    match = true;
    for(i = 0; i < (sizeof(eepromReadBuffer)); i++)
    {
```

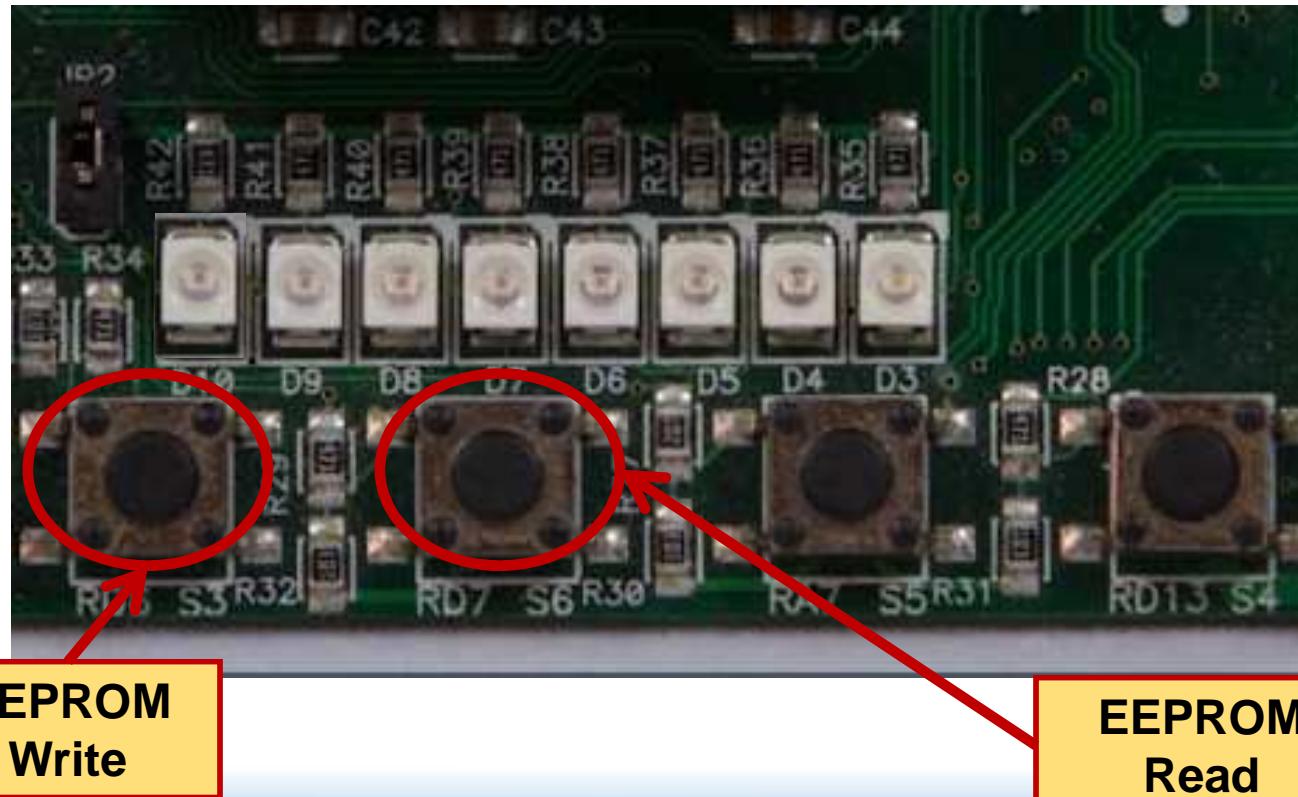
PIC24 Lab 5

- “Make and Program” the project



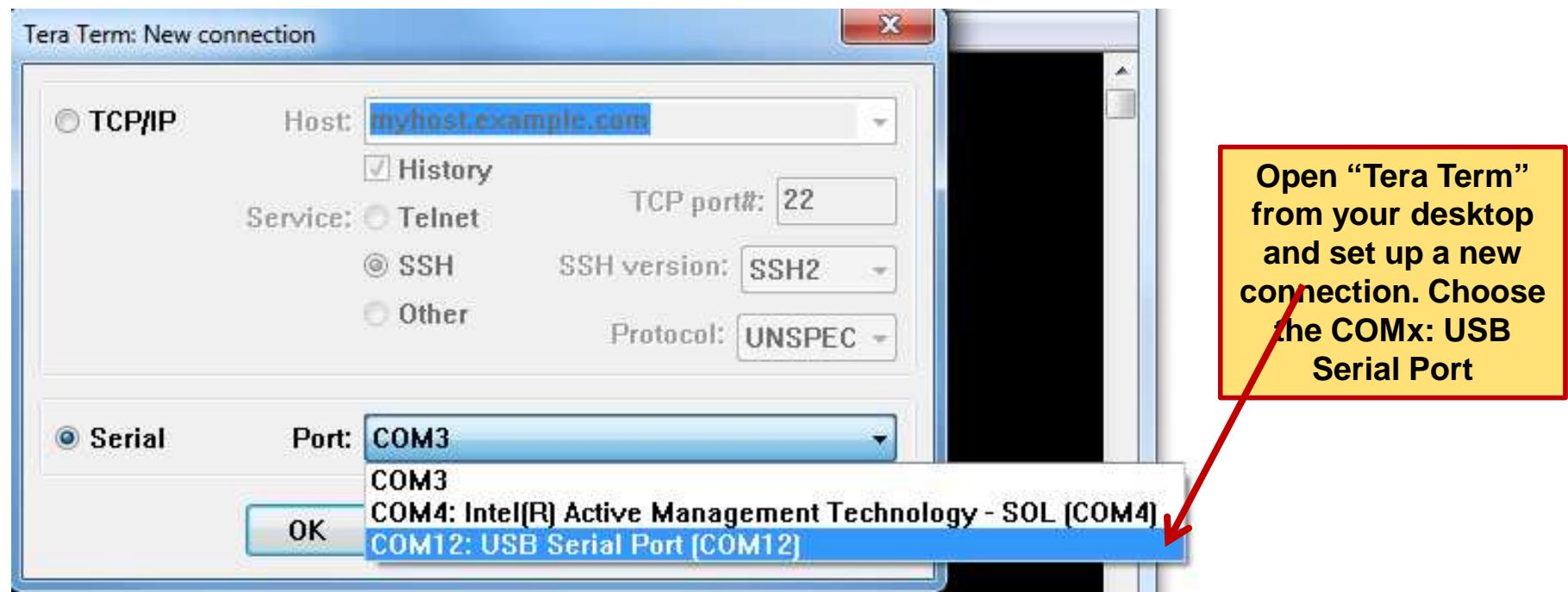
PIC24 Lab 5

- Press S3 to execute EEPROM Write
- Press S6 to execute EEPROM Read



PIC24 Lab 5

- Start Tera Term
- Select the USB Serial Port



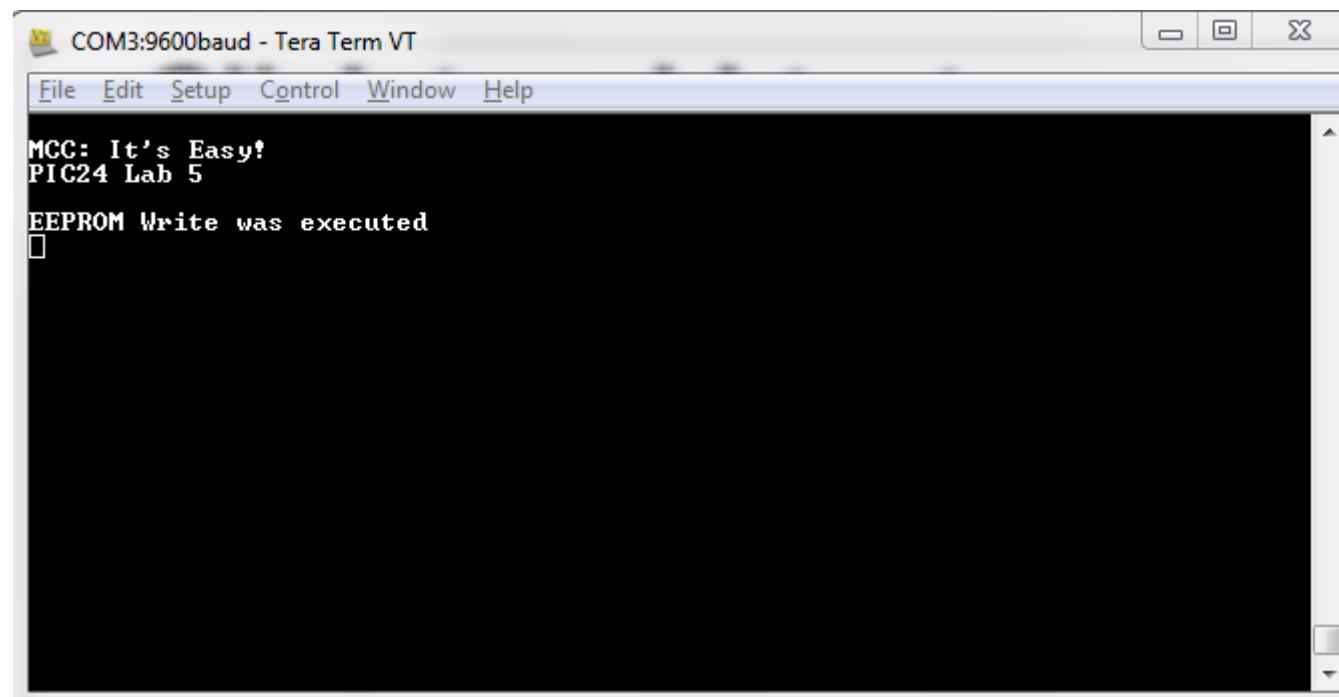
PIC24 Lab 5

Tera Term should display Lab 5 welcome message as shown



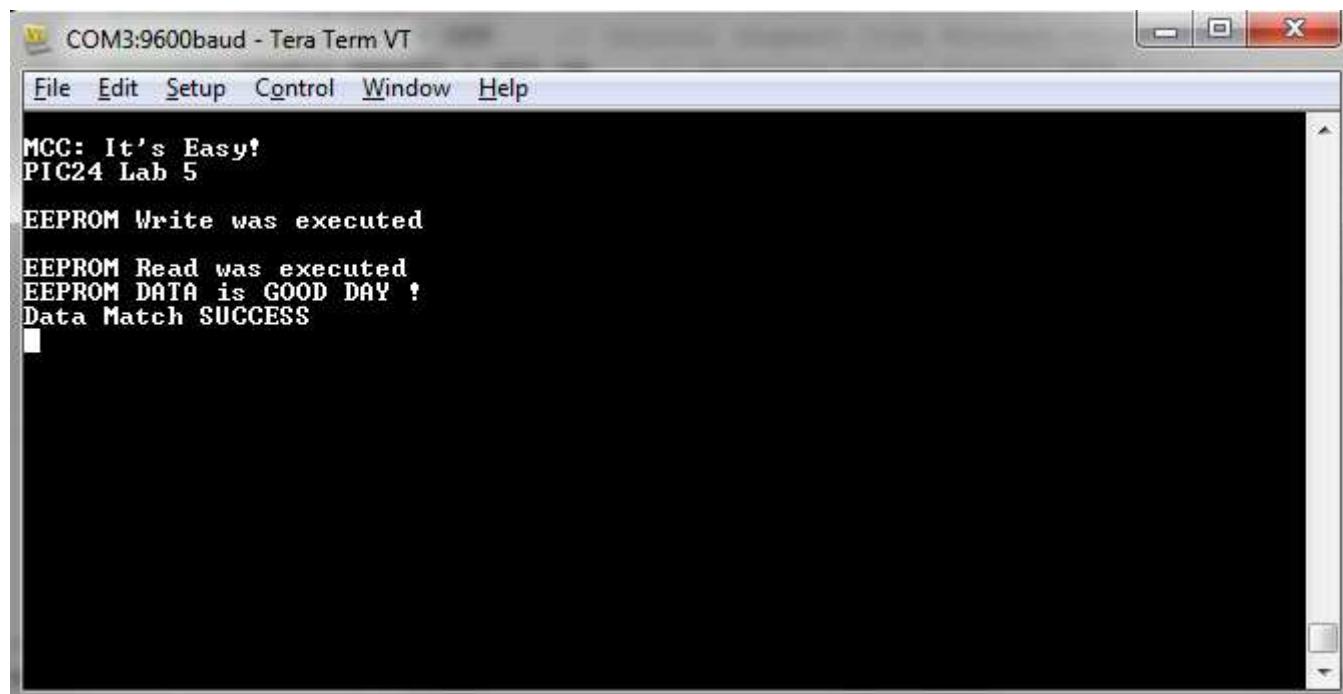
PIC24 Lab 5

**Press Switch S3 to execute EEPROM Write
Tera Term should display execution message**



PIC24 Lab 5

**Press Switch S6 to execute EEPROM Read
Tera Term should display execution and data
match success messages**



PIC24 Lab 5 Summary

- We learned how to write and read data from the I2C EEPROM board with the Explorer 16 Board
- We displayed the message read back from the EEPROM on Tera Term and verified if it was a success or a failure

PIC24 Lab 6:

Let's Play some music using SPI + PWM (Bonus Lab)

PIC24 Lab 6 Objectives

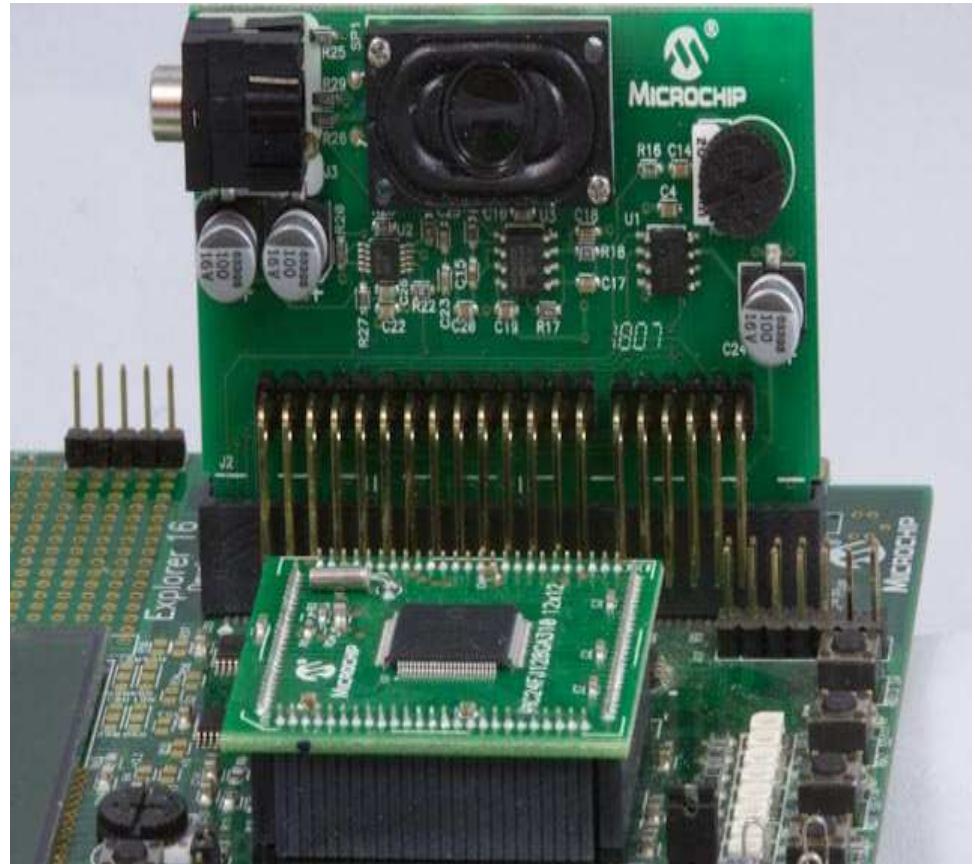
- Set up SPI to read music stored on EEPROM
- Utilize PWM to play music on the Speech Playback PICtail Plus board
- Enable the ADC to read the potentiometer connected to input RA5 to alter the speed the music is played
- Integrate MCC generated drivers with existing application

PIC24 Lab 6 Overview

- **Close any files open in the editor, and close all MPLAB® X IDE Projects**
- **Open lab6_16bit.X**
- **Install the Speech Playback PICTail Board**
- **Set up SPI to communicate with EEPROM on the Explorer 16 Board**
- **Setup OC1 to play music**

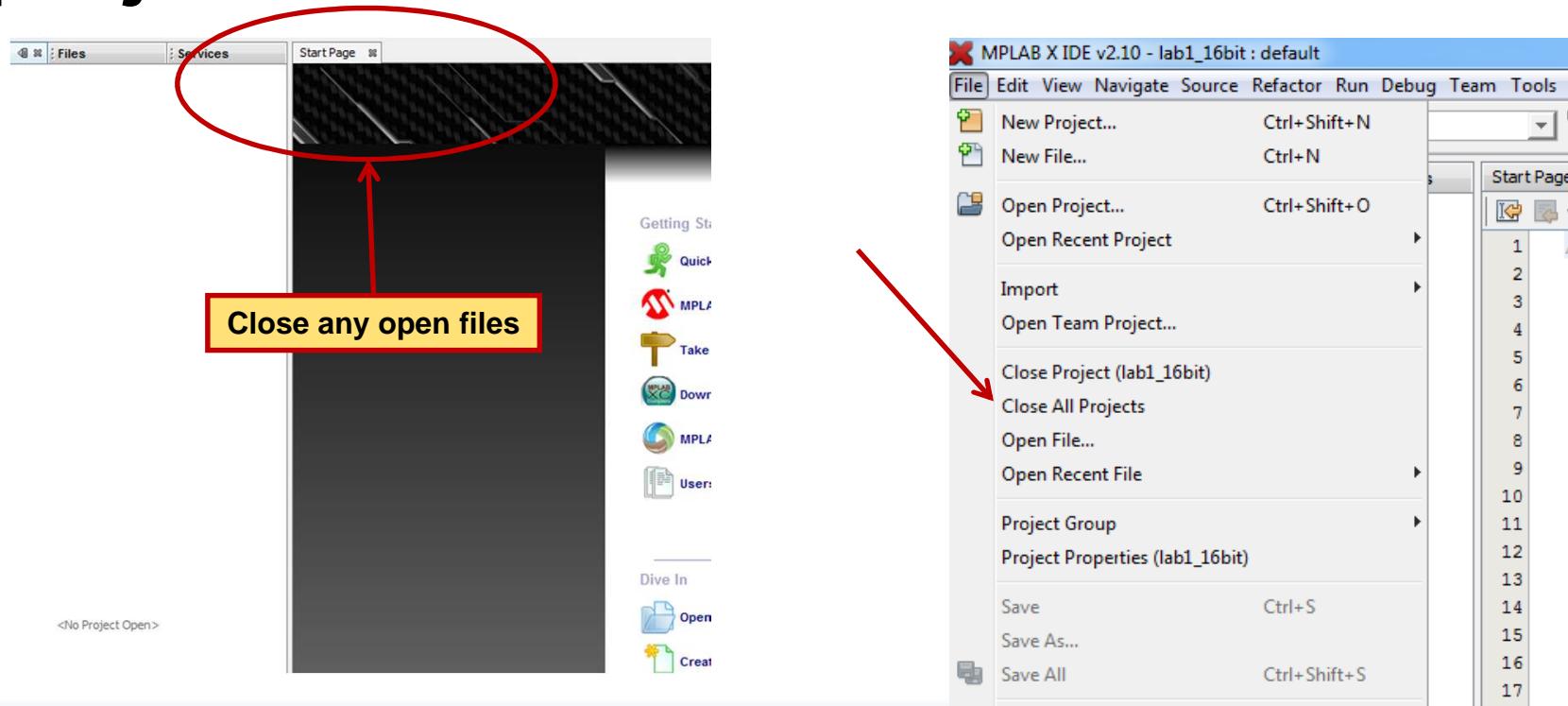
PIC24 Lab 6

- Remove power from the Explorer 16 Board
- Install the Speech Playback PICtail board
- Reconnect the Power to the board



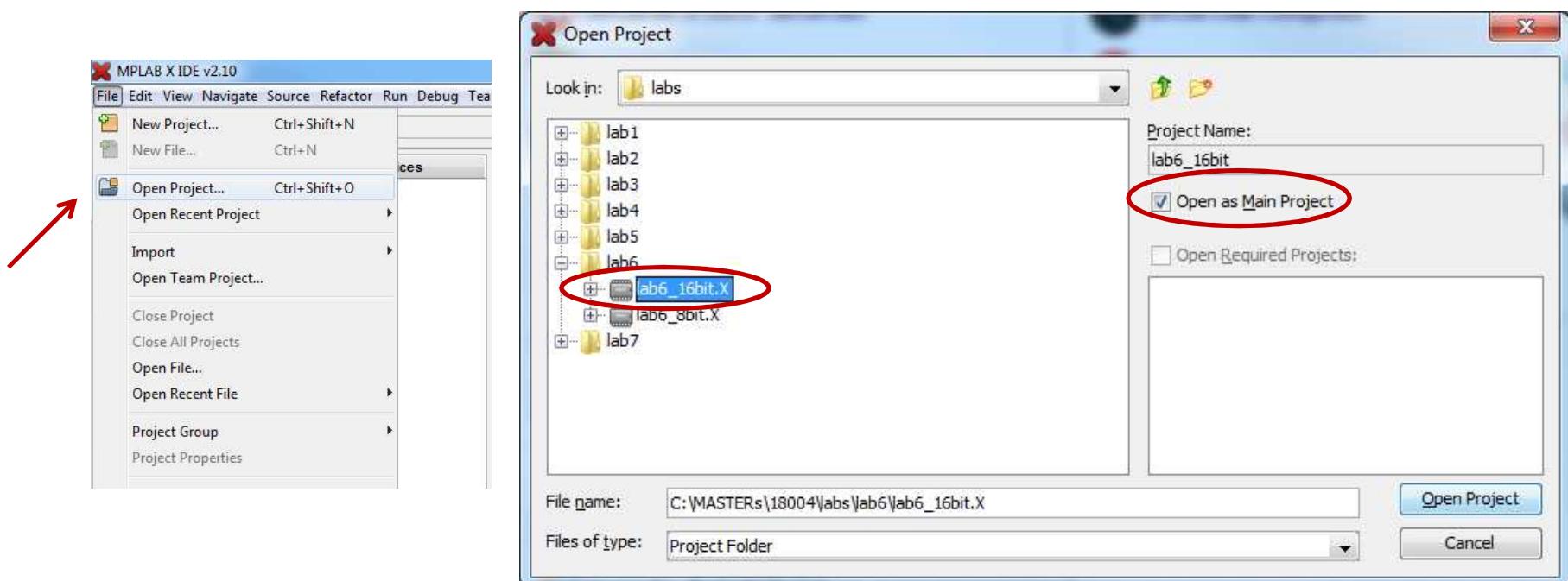
PIC24 Lab 6

- Close any open files in the editor window, and all open MPLAB® X IDE projects



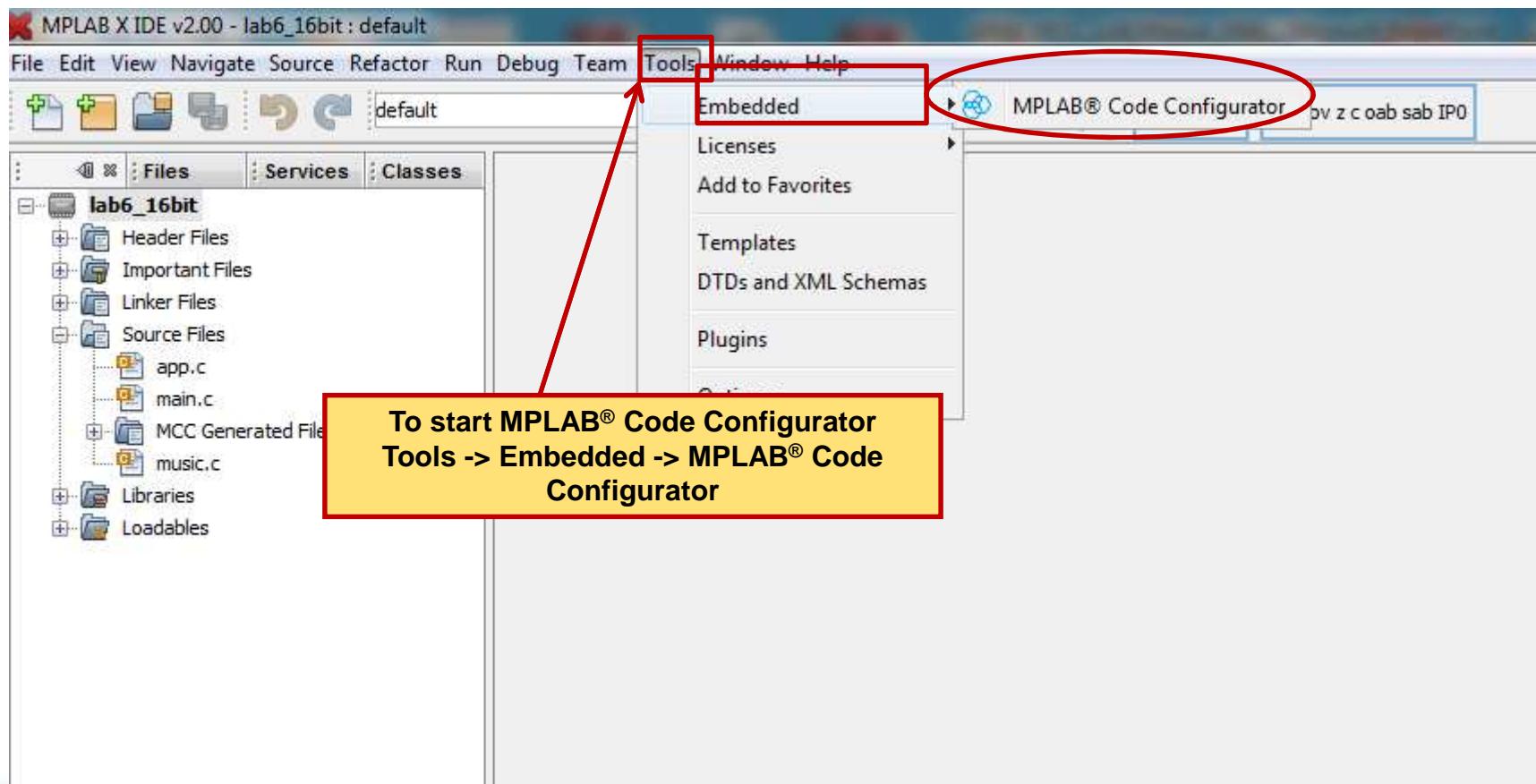
Lab 6

- Open lab6_16bit.X



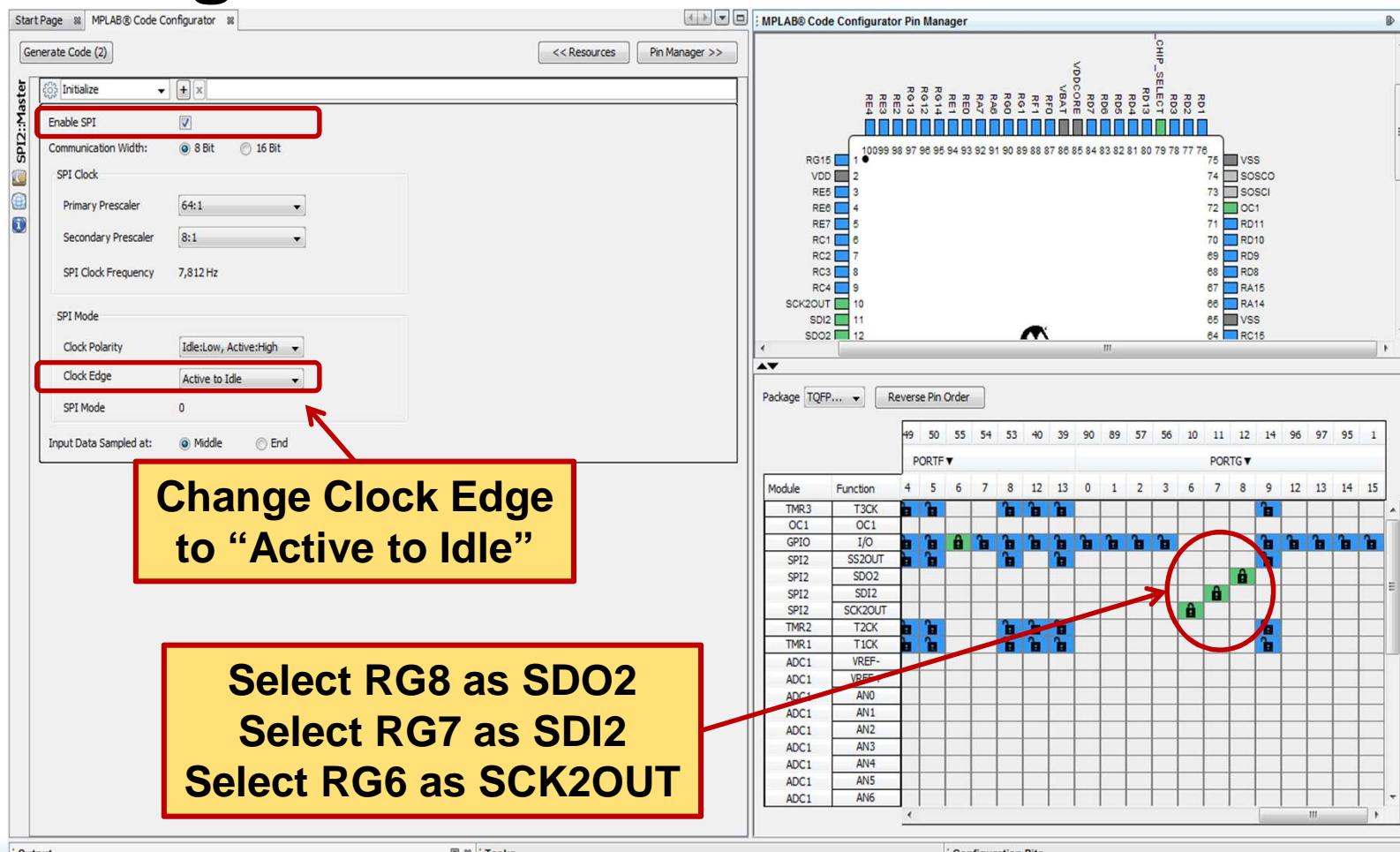
PIC24 Lab 6

- Start MPLAB® Code Configurator



PIC24 Lab 6

- Configure the SPI2 as SPI Master



The screenshot shows the MPLAB® Code Configurator interface with two main windows:

- SPI2 Master Configuration Window:**
 - Enable SPI:** Checked.
 - Communication Width:** 8 Bit.
 - SPI Clock:**
 - Primary Prescaler: 64:1
 - Secondary Prescaler: 8:1
 - SPI Clock Frequency: 7,812 Hz
 - SPI Mode:**
 - Clock Polarity: Idle:Low, Active:High
 - Clock Edge:** Active to Idle (highlighted with a red box)
 - SPI Mode: 0
 - Input Data Sampled at: Middle
- MPLAB® Code Configurator Pin Manager:**
 - Shows the pin map for the PIC24 device, mapping pins to functions like VDDCORE, VBAT, and various digital pins (RD1, RD2, RD3, RD4, RD5, RD6, RD7, RD8, RD9, RD10, RD11, RD12, RD13, RD14, RD15, RD16, RD17, RD18, RD19, RD20, RD21, RD22, RD23, RD24, RD25, RD26, RD27, RD28, RD29, RD30, RD31, RD32, RD33, RD34, RD35, RD36, RD37, RD38, RD39, RD40, RD41, RD42, RD43, RD44, RD45, RD46, RD47, RD48, RD49, RD50, RD51, RD52, RD53, RD54, RD55, RD56, RD57, RD58, RD59, RD60, RD61, RD62, RD63, RD64, RD65, RD66, RD67, RD68, RD69, RD70, RD71, RD72, RD73, RD74, RD75, RD76, RD77, RD78, RD79, RD80, RD81, RD82, RD83, RD84, RD85, RD86, RD87, RD88, RD89, RD90, RD91, RD92, RD93, RD94, RD95, RD96, RD97, RD98, RD99, RD100).
 - Shows the package pinout for TQFP...
 - Shows the configuration bit matrix for PORTF and PORTG.

Annotations:

- A yellow box highlights the "Clock Edge" dropdown in the SPI2 Master window with the text: "Change Clock Edge to ‘Active to Idle’".
- A yellow box highlights the "SDO2", "SDI2", and "SCK2OUT" pins in the Pin Manager with the text: "Select RG8 as SDO2", "Select RG7 as SDI2", and "Select RG6 as SCK2OUT".

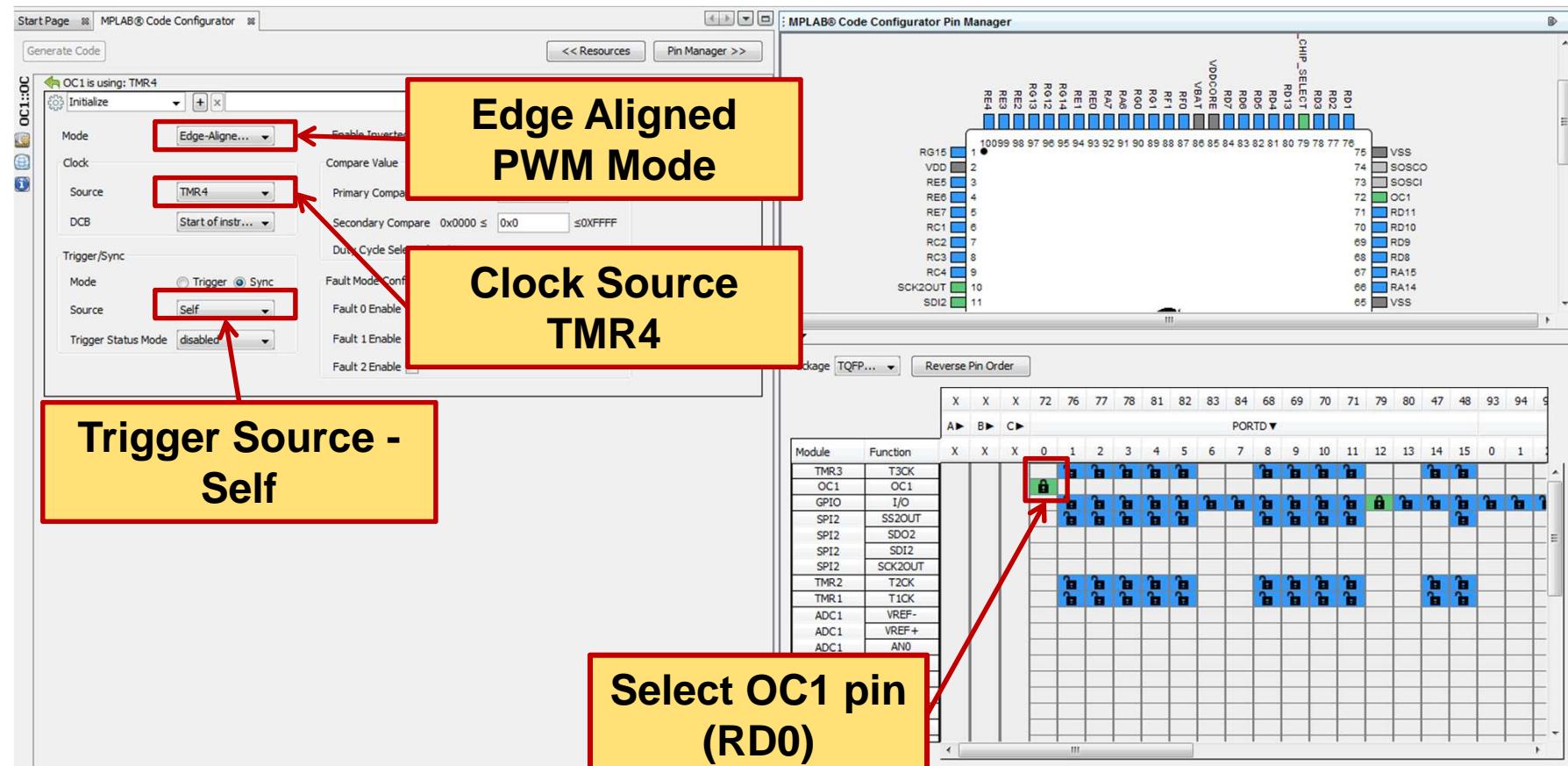
PIC24 Lab 6

- Did you know ...
- that the MPLAB X Output window displays information from MCC



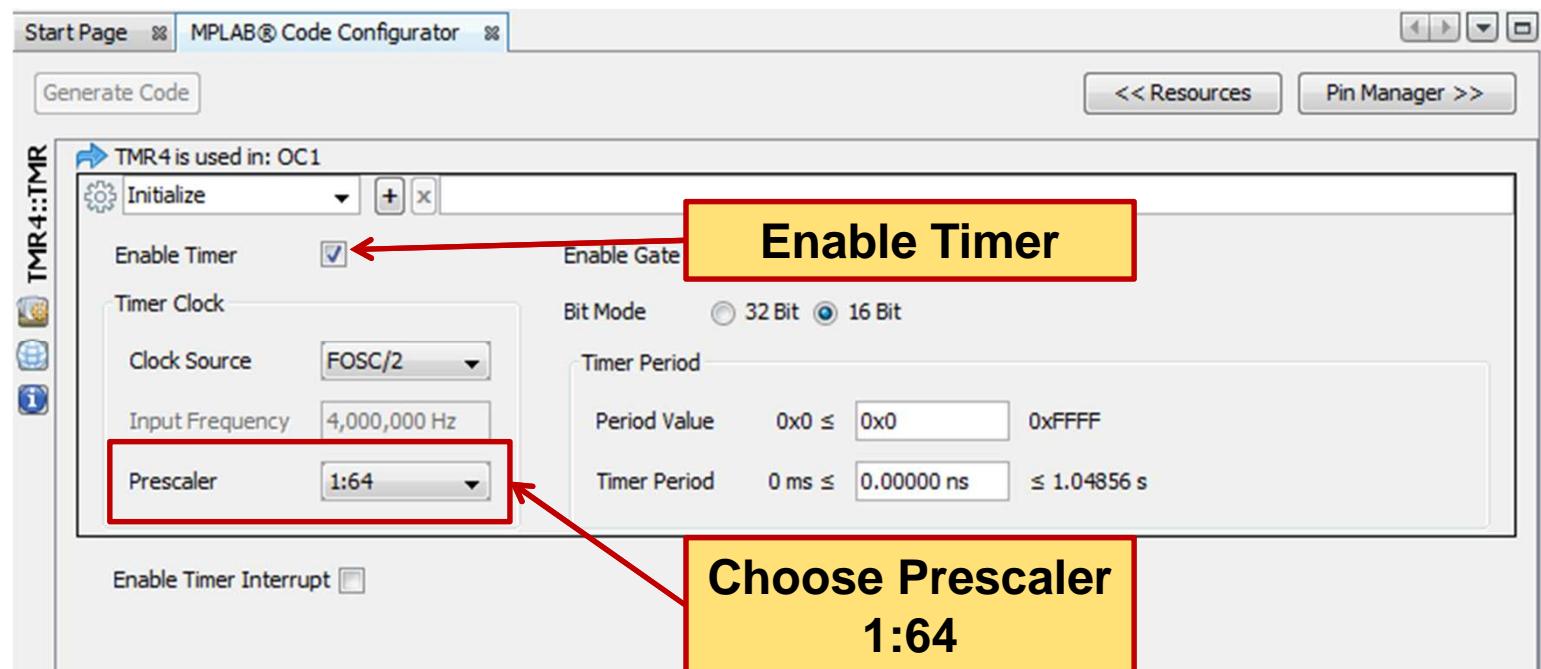
PIC24 Lab 6

- Set up OC1 in Edge Aligned PWM Mode



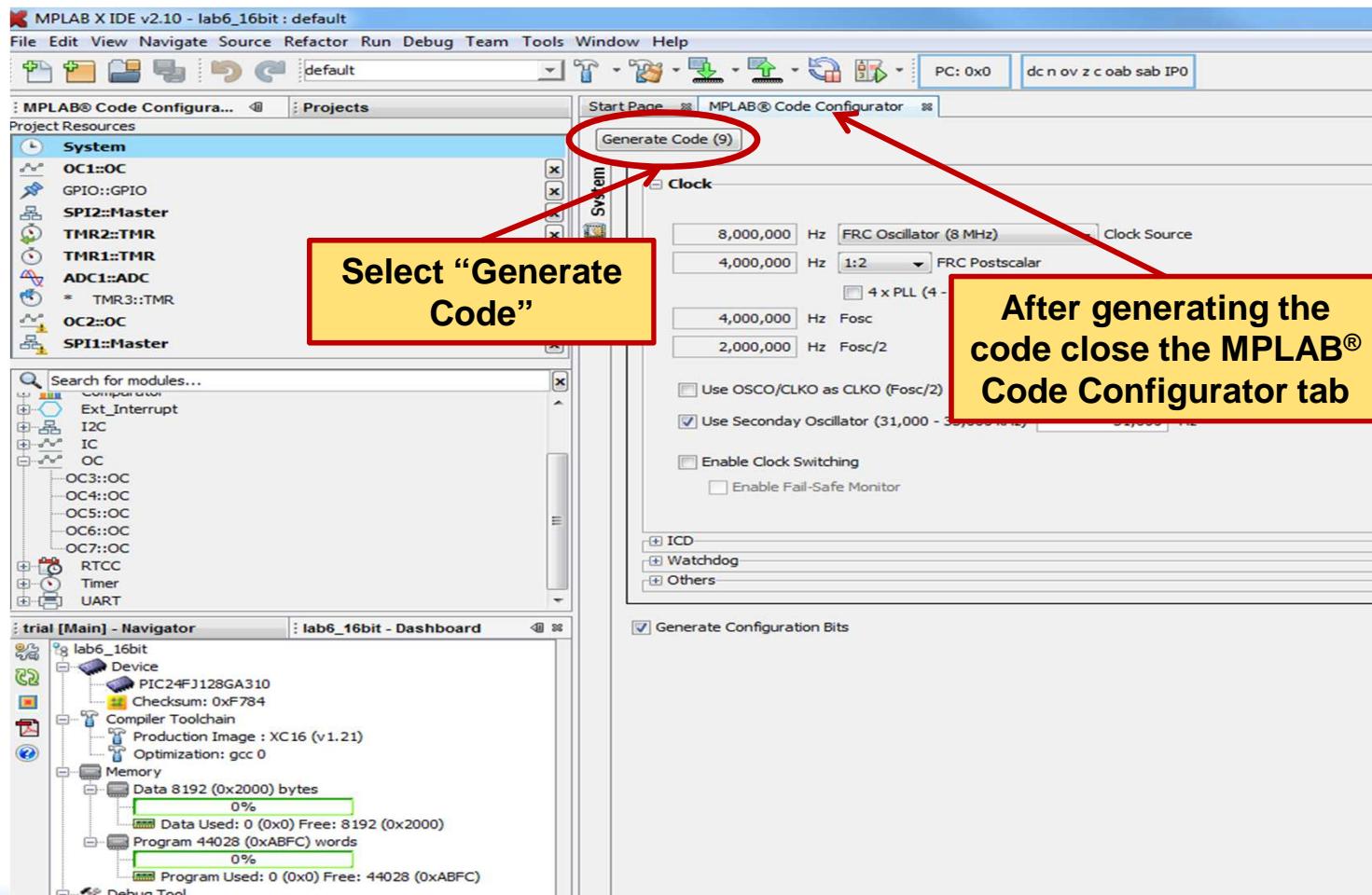
PIC24 Lab 6

- Enable TMR4 to use as Clock Source for OC1



PIC24 Lab 6

● Generate Code



PIC24 Lab 6

- “Make and Program” the project

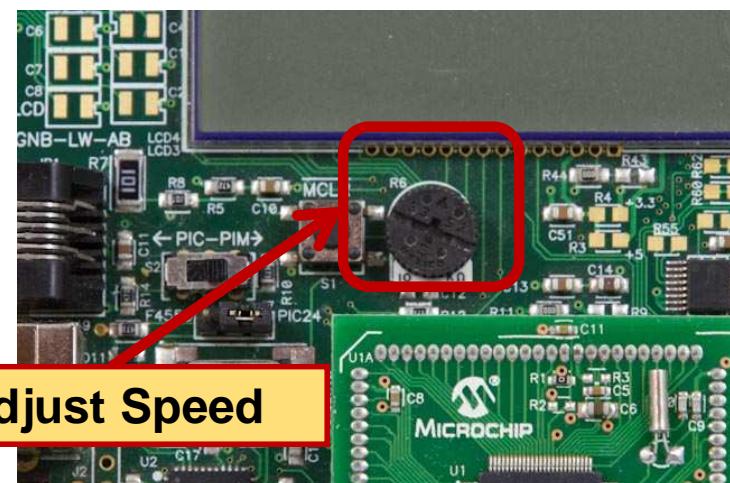
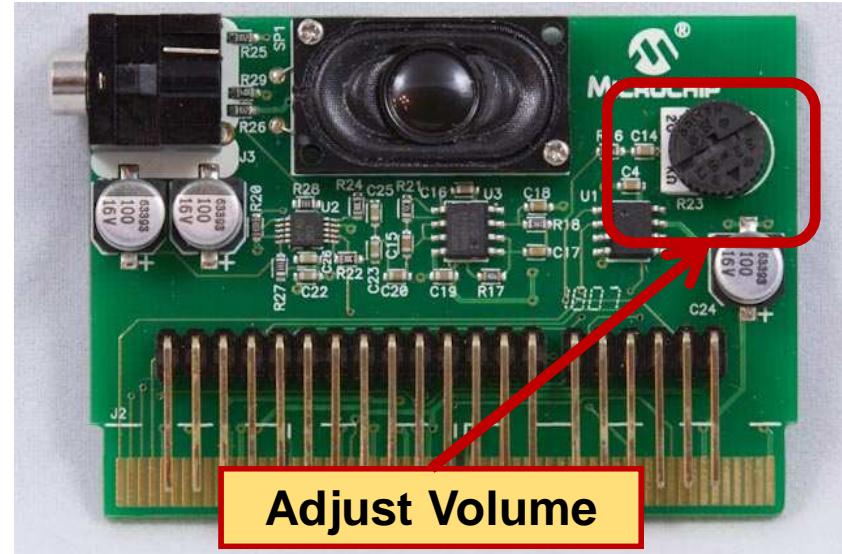


PIC24 Lab 6

- **Music playing**
- **Adjust Volume**

**On Speech
Playback board**

- **Adjust Speed
on Explorer 16
Board**



PIC24 Lab 6 Summary

- We used SPI to read music data from an EEPROM
- We use the OC module to play music with the Speech Playback PICtail board
- We used an ADC input to control the speed the song was played

PIC24 Lab 7: Use all the features of the Explorer 16 board (Bonus Lab)

PIC24 Lab 7 Objectives

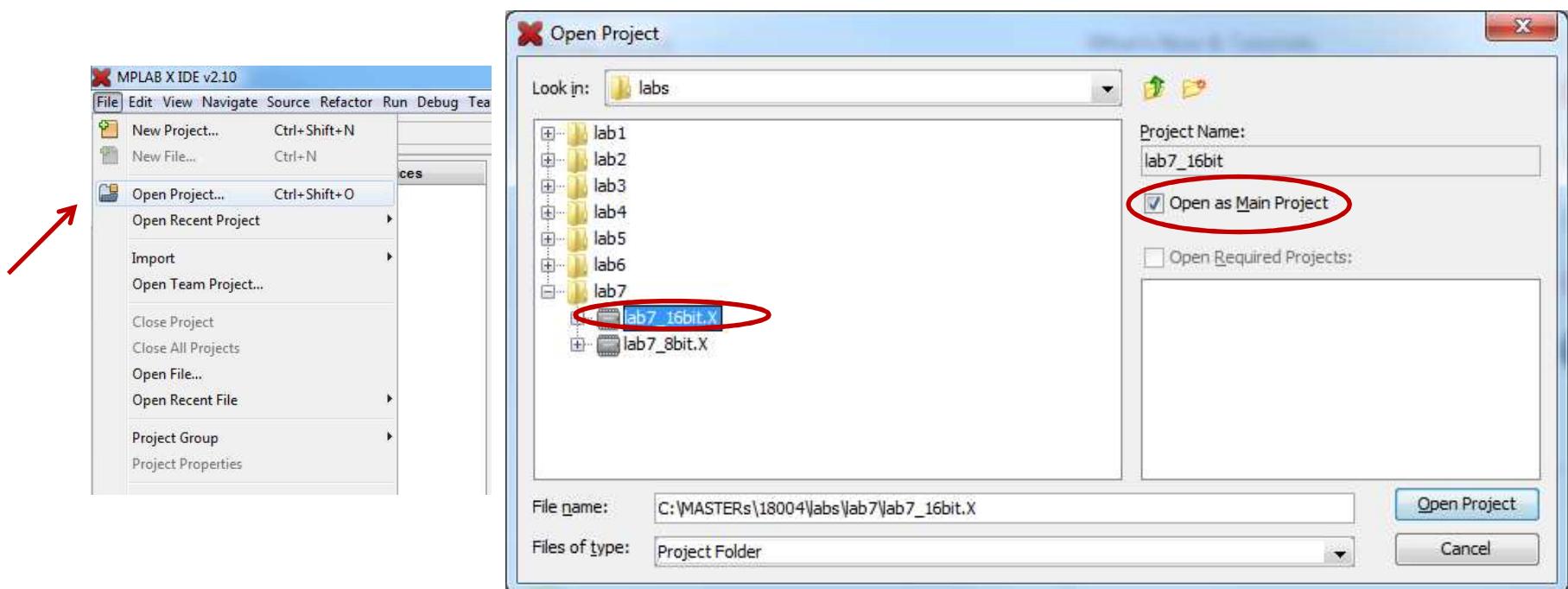
- To learn to use all features of Explorer 16 board and build an application
- To use the on-board LCD display.
- To use switches to select from a menu of commands to
 - display potentiometer voltage
 - display temperature
 - display date and time
 - toggle LED

PIC24 Lab 7 Overview

- **Close any files open in the editor, and close all MPLAB® X IDE Projects**
- **Open lab7_16bit.X**
- **Setup RTCC**
 - **Enable RTCC**
 - **Enable RTCC Output**
 - **Set RTCC Output to RTCC Clock**

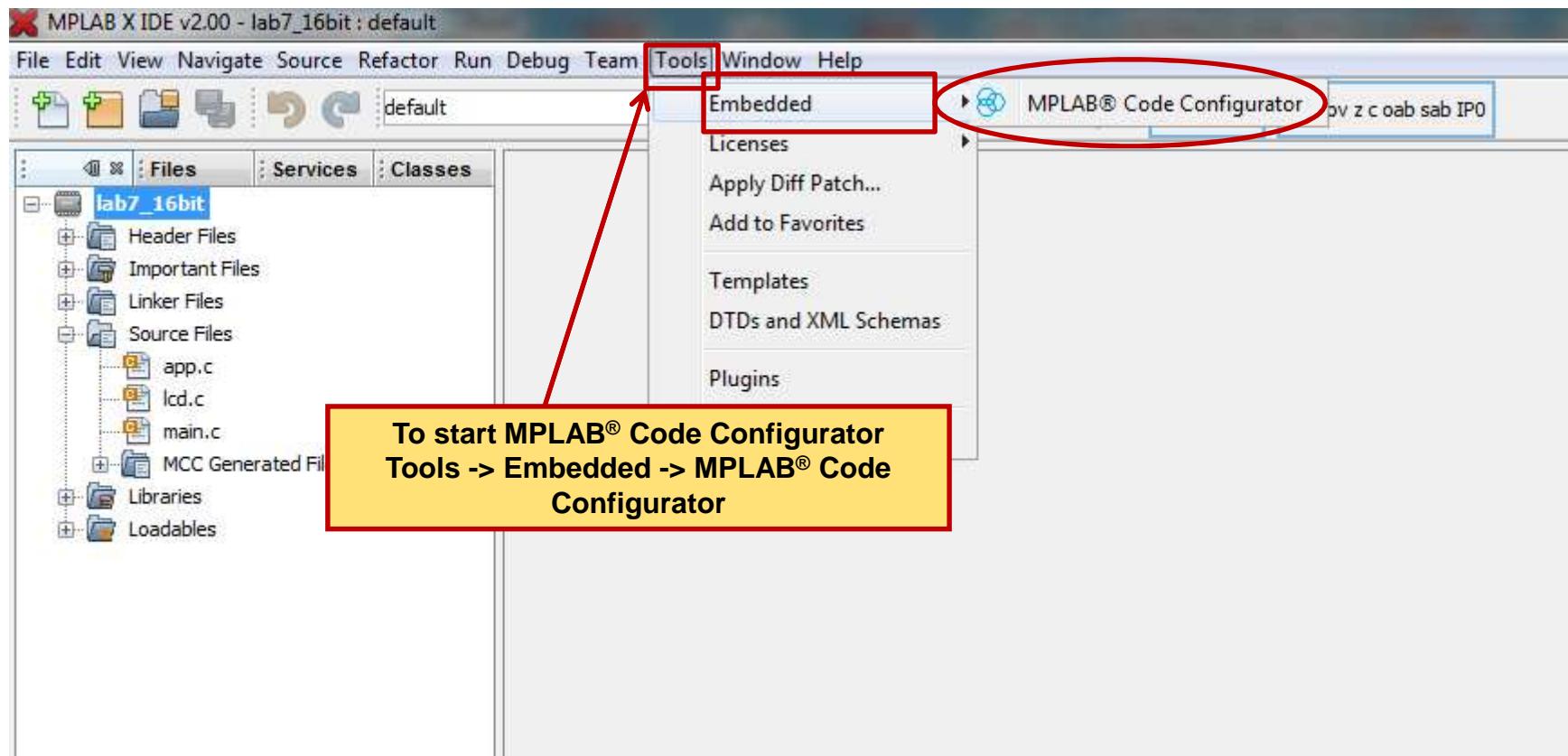
PIC24 Lab 7

- Open lab7_16bit.X



PIC24 Lab 7

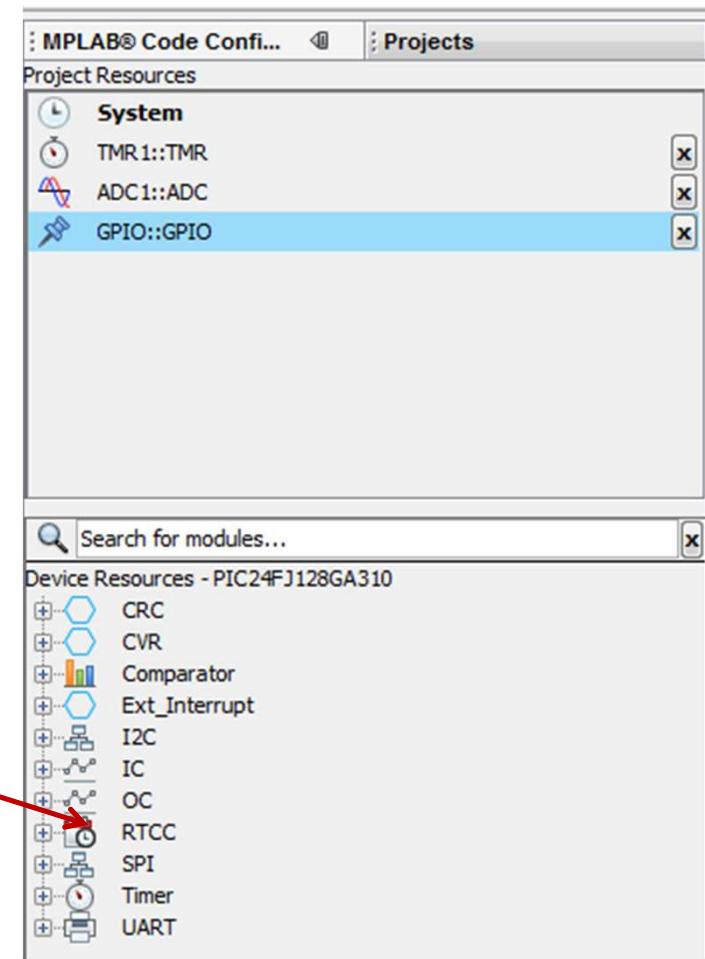
- Start MPLAB® Code Configurator



PIC24 Lab 7

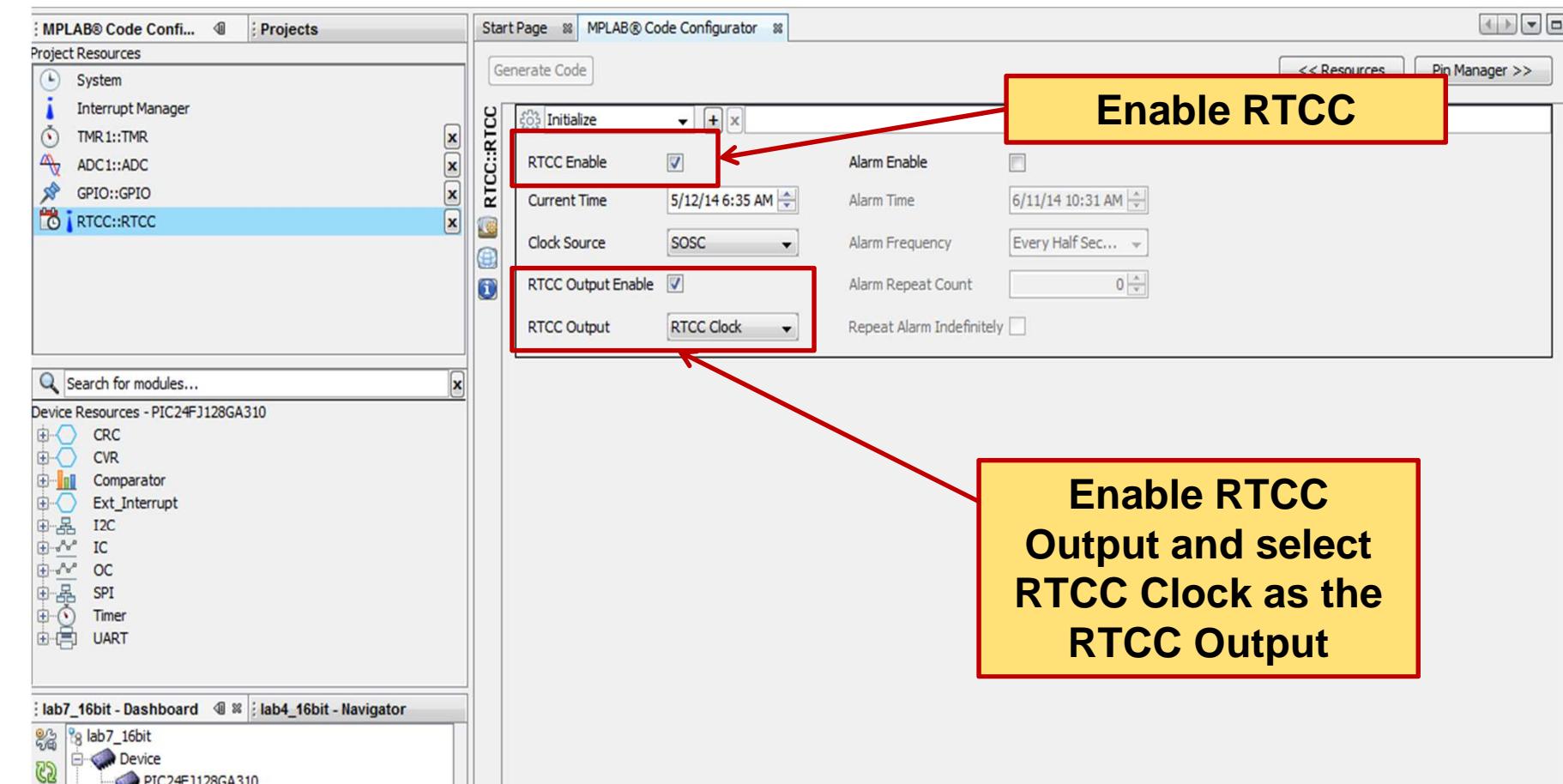
- Select RTCC::RTCC from Device Resources

Double Click on RTCC and Double click on RTCC::RTCC to configure input-output pins



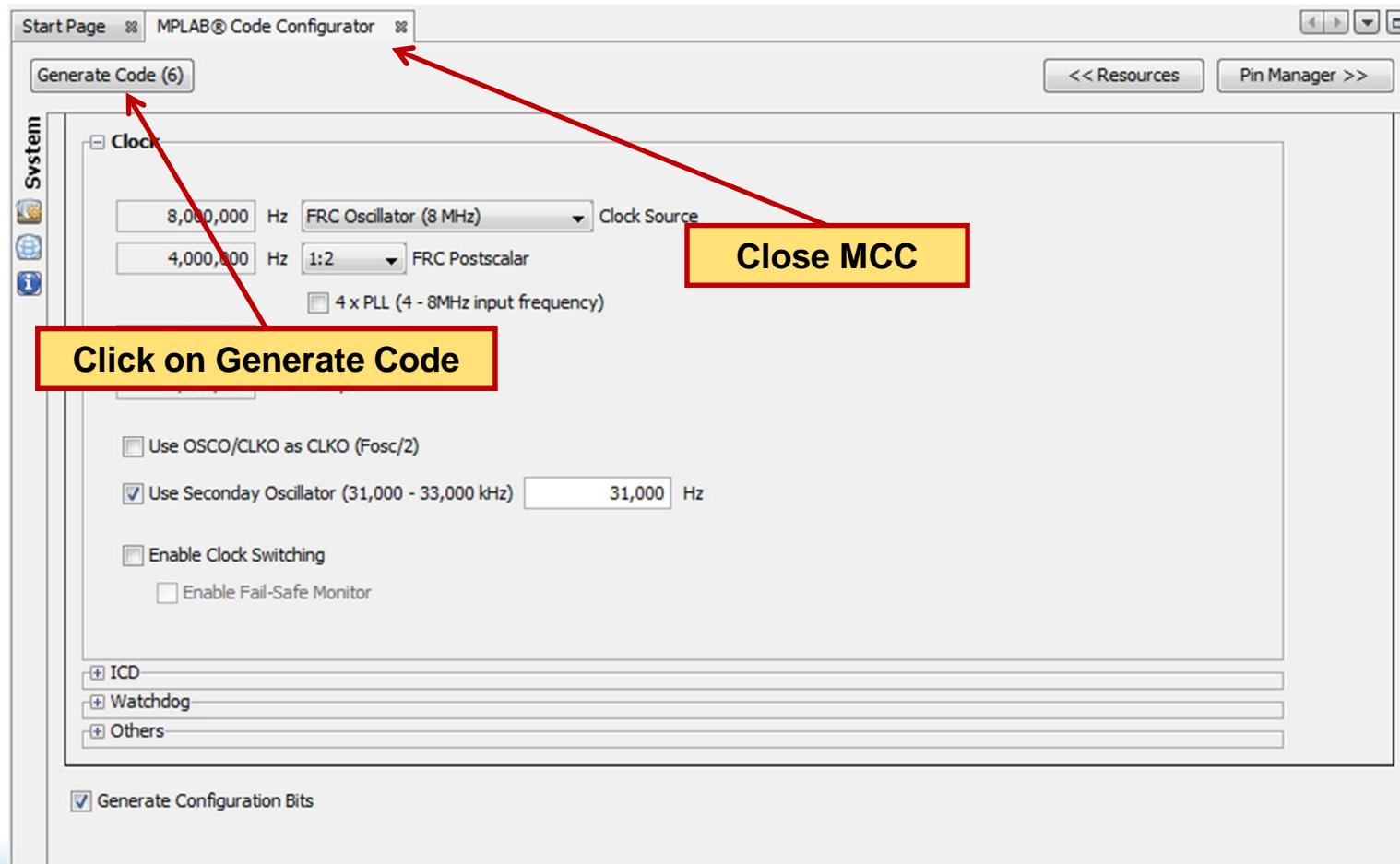
PIC24 Lab 7

• Configure RTCC module



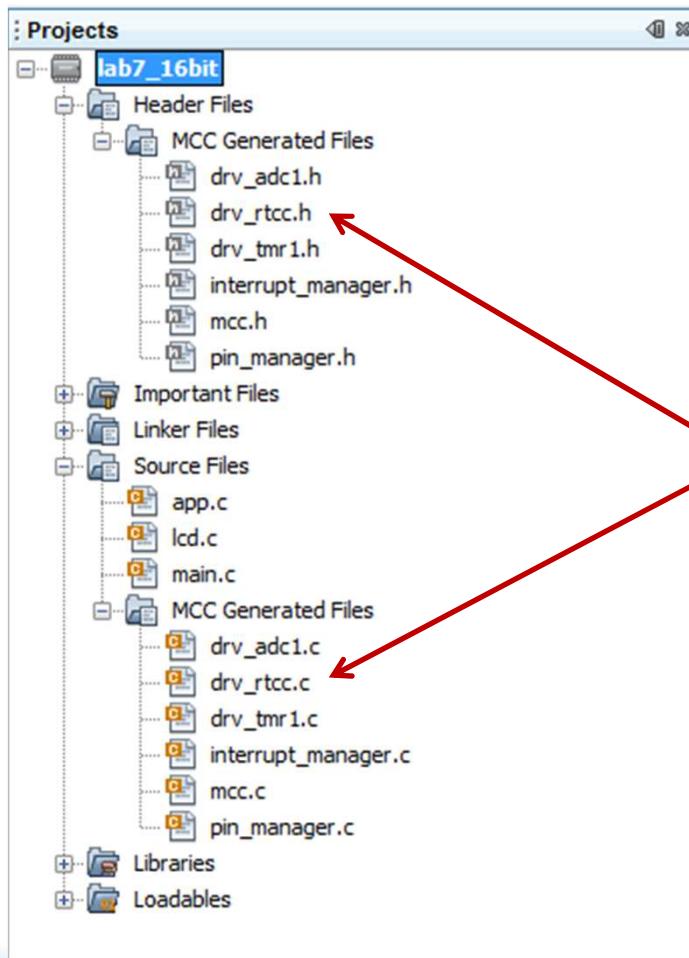
PIC24 Lab 7

- Generate Code



PIC24 Lab 7

- RTCC driver files added to the project



**MPLAB® Code
Configurator driver files
for RTCC module are
added into the project**

PIC24 Lab 7

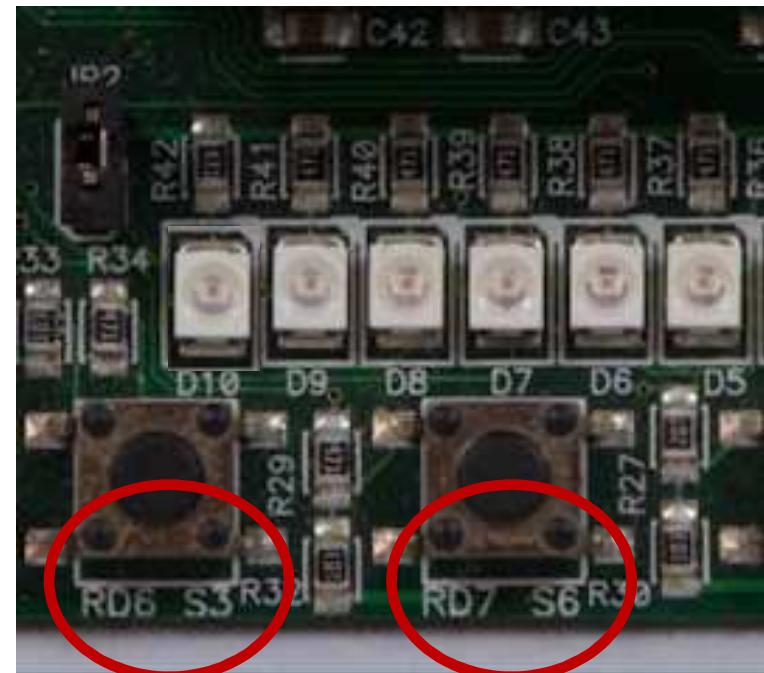
- “Make and Program” the project



PIC24 Lab 5

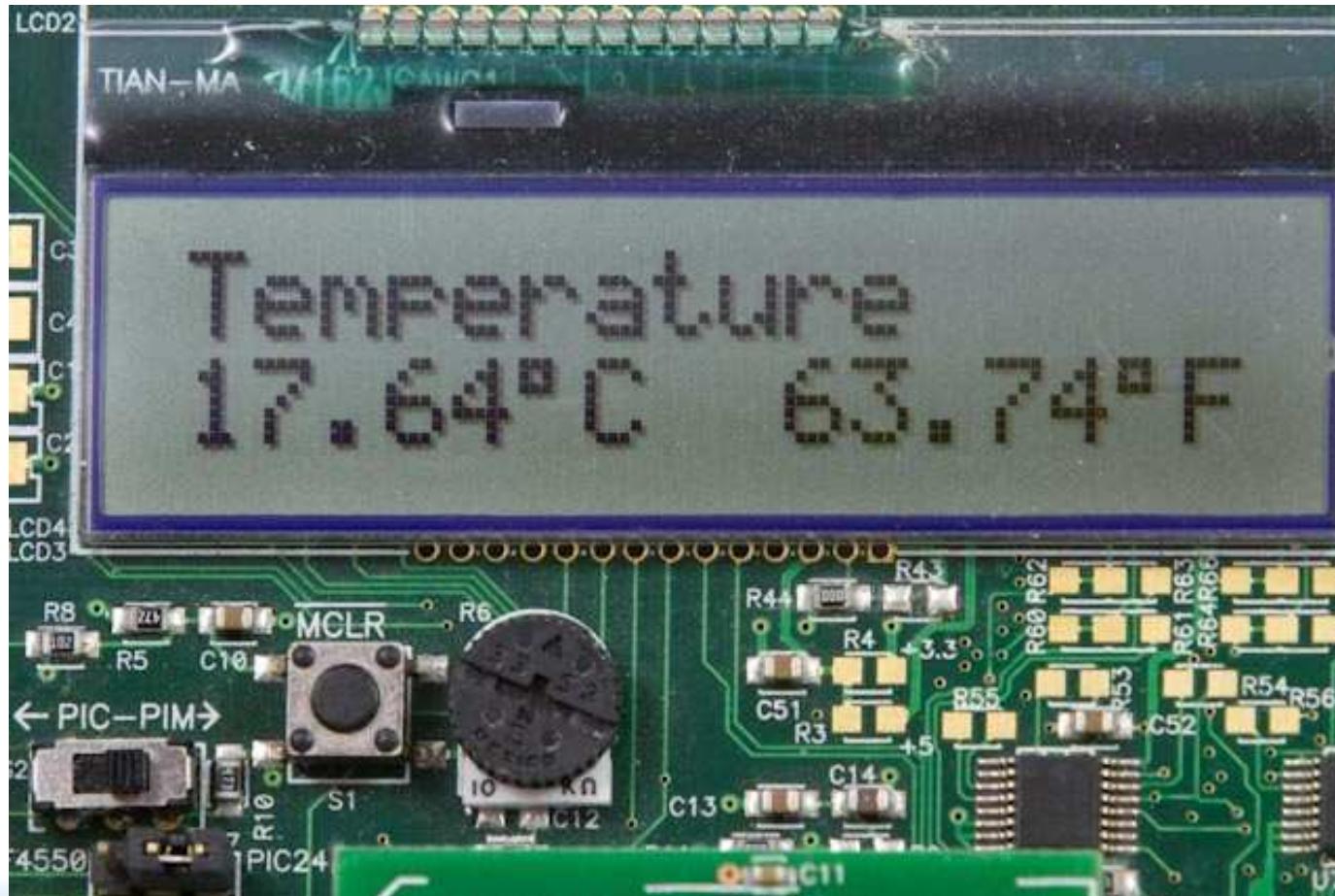
- Use the buttons S3 and S6 to navigate through the menu to display:

- Temperature
- Potentiometer voltage
- Date and Time
- LED blinking



PIC24 Lab 7

Temperature Sensor Demo



PIC24 Lab 7

Potentiometer Demo



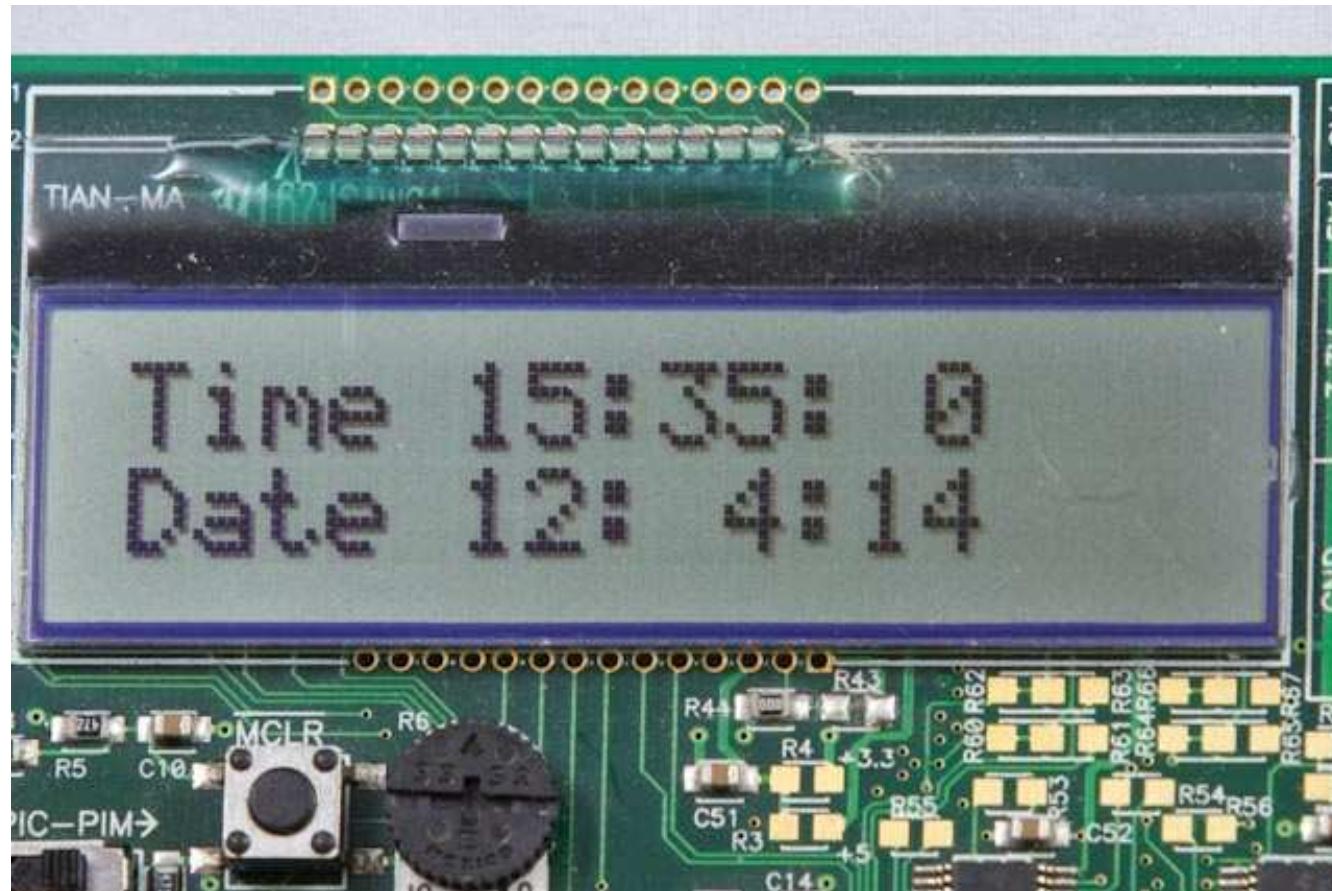
PIC24 Lab 7

LED Demo



PIC24 Lab 7

RTCC Demo



PIC24 Lab 7 Summary

- We built an application that uses all features on the Explorer 16 Board.
- We learned to display data on the LCD of the Explorer 16 Board

Summary

- **Today we covered:**
 - Installing and using MCC for generating code in your projects
 - Demonstrated how easily peripherals can be setup using MCC
 - Had fun using a very cool and very usable GUI

Summary

MCC v2.0 supports the following:

- **PIC12:**

- (L)F1501
- (L)F1822
- (L)F1840

- **PIC16:**

- (L)F15xx
- (L)F17xx
- (L)F18xx
- (L)F19xx

- **PIC18:**

- (L)F2xK20
- (L)F4xK20
- (L)F2xK22
- (L)F4xK22

- **PIC24:**

- FJxxGA310
- F(V)16KMxx

Summary

- **Future support**
 - Peripheral Support
 - Device Support
 - Enhancements

Questions?



Thank you!

Dev Tools For This Class

- **(AC002014) 9V 1.3A Universal Wall Mount Power Supply**
- **(DM240001) PICDEM PIC24/dsPIC33 Explorer 16 Kit**
- **(DM183032) PICDEM PIC18 Explorer Board**
- **(MCP2200EV-VCP) MCP2200 USB to RS232 Demo Board**
- **(MA180026) PIC18F45K22 44-PIN TQFP TO 84-PIN PIM**
- **(MA240029) PIC24FJ128GA310 General Purpose PIM**

Dev Tools For This Class

- **(AC164125) Speech Playback Speech Playback PICtail™ Plus board Plus Daughter Board**
- **MPLAB® REAL ICE™ In-circuit emulator**
- **(AC243003) I²C EEPROM**

Appendix

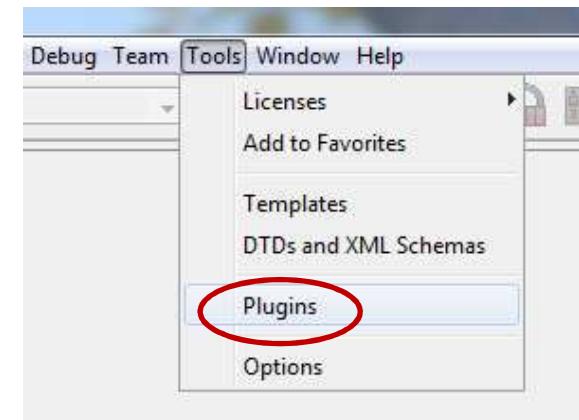
- **The MPLAB Code Configurator Plugin should already be installed for this MASTERs class**
- **The installation instructions are required only if the MCC Plugin needs to be installed**
- **Proceed with the Labs if the MCC Plugin is already installed**

Installing the MCC Plugin

- There are 2 methods to install MPLAB X Plugins
 - Off-line
 - Normal Plugin installation (use this during MASTERs)
- Use only one of these methods

Off-line Plugin installation

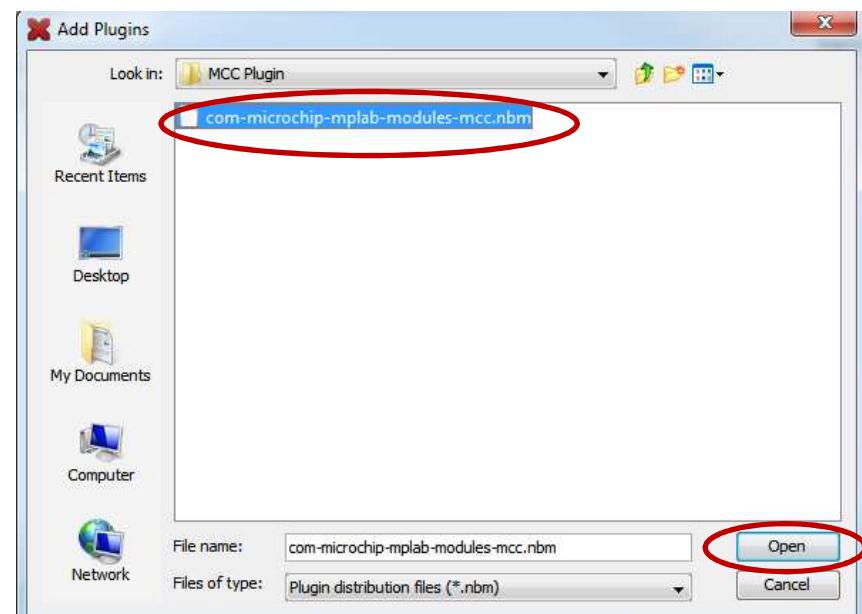
- Access Tools – Plugins
- Select the Downloaded tab and Click on Add Plugins



Off-line Plugin installation

Continued

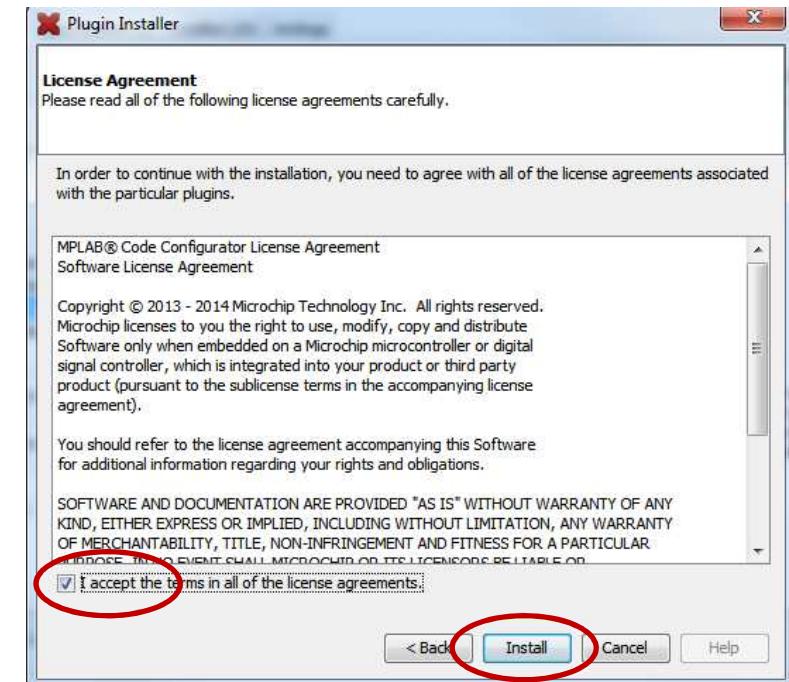
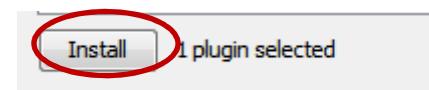
- Navigate to the folder and select the .nbm file and click on open



Off-line Plugin installation

Continued

- Click Install

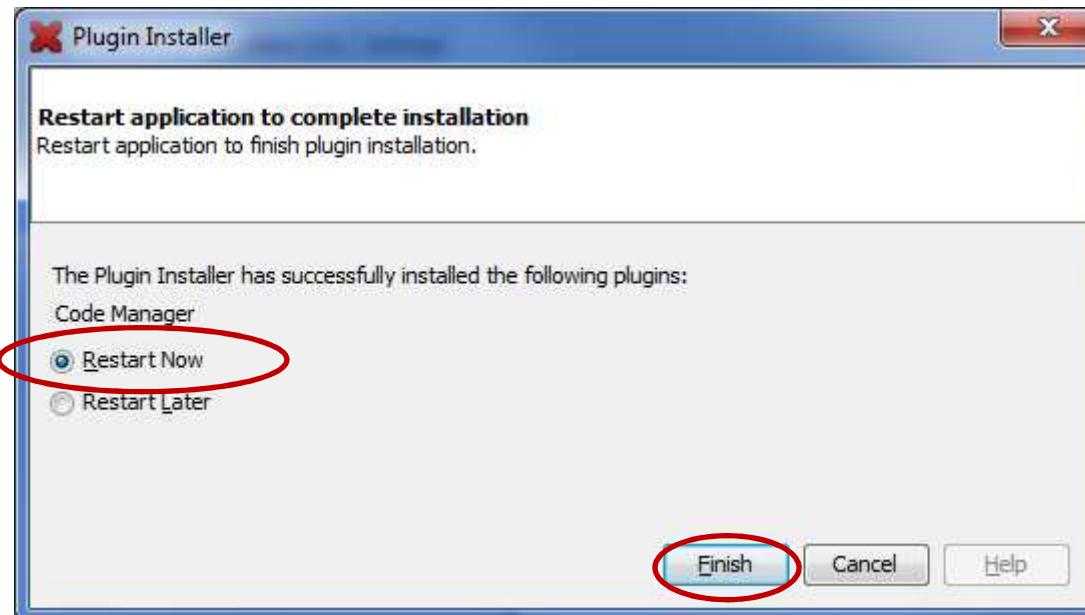


- Accept and Install License Agreement

Off-line Plugin installation

Continued

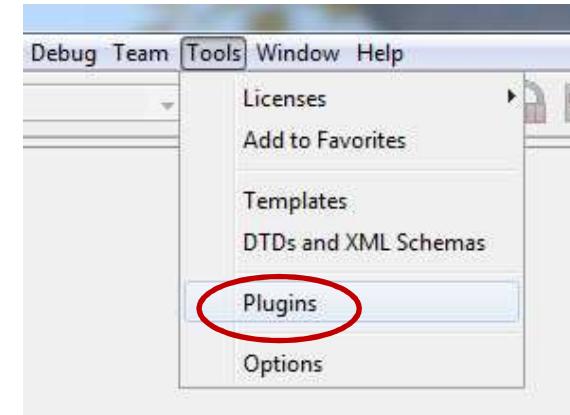
- Select “Restart Now”
- Click Finish



- Proceed to the Labs

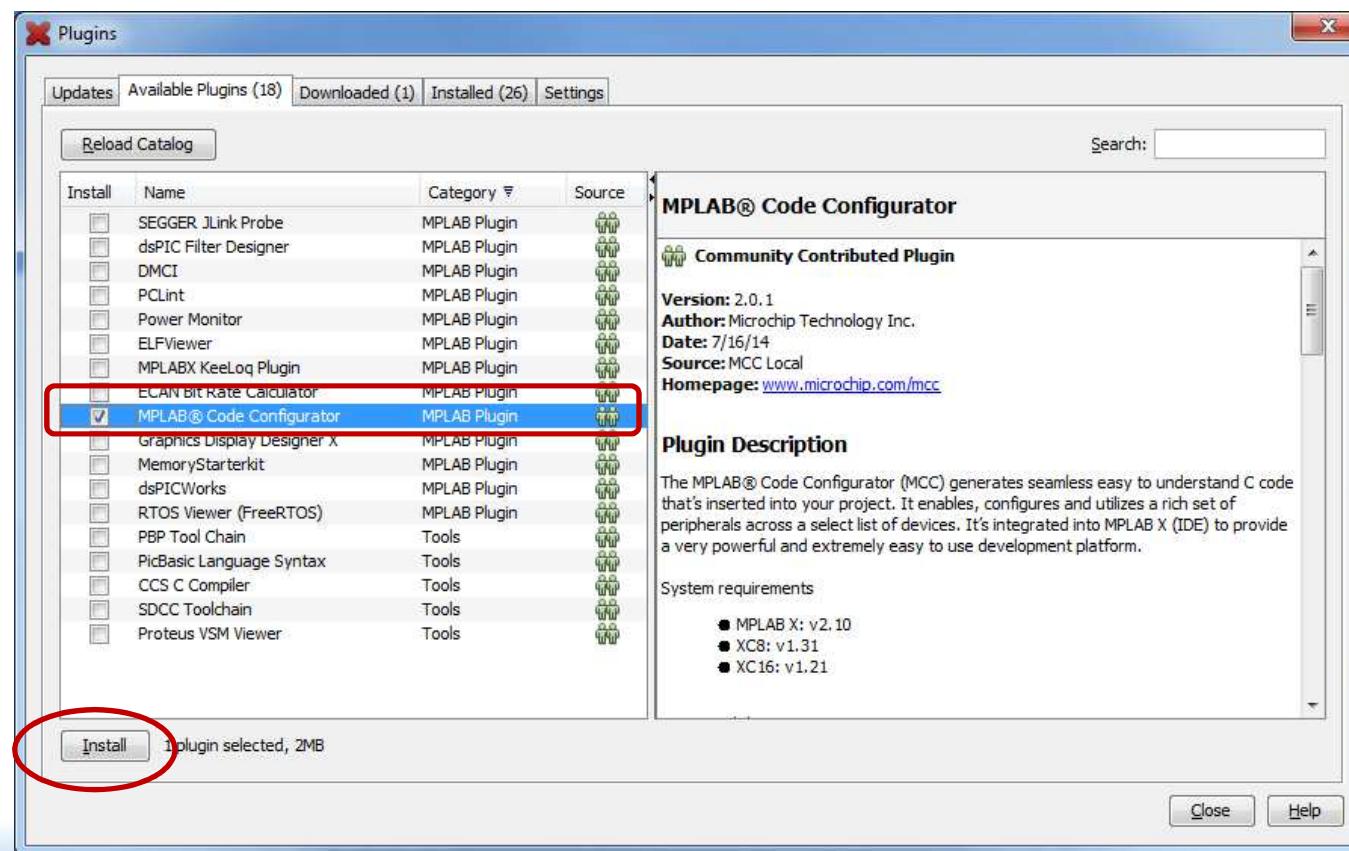
Normal Plugin Installation

- Access Tools – Plugins
- Select the Available Plugins Tab



Normal Plugin Installation continued

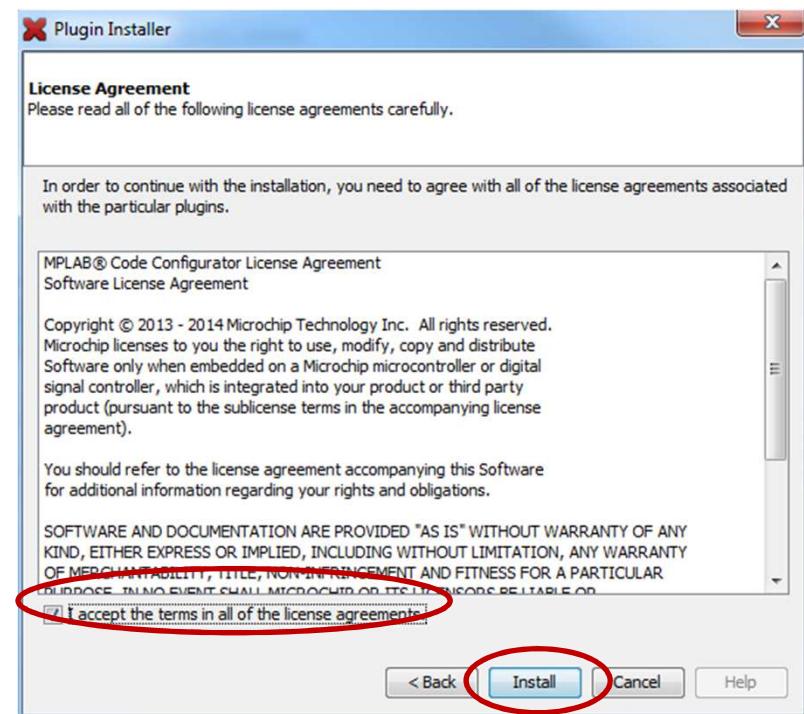
- Select MPLAB Code Configurator
- Click Install



Normal Plugin Installation

continued

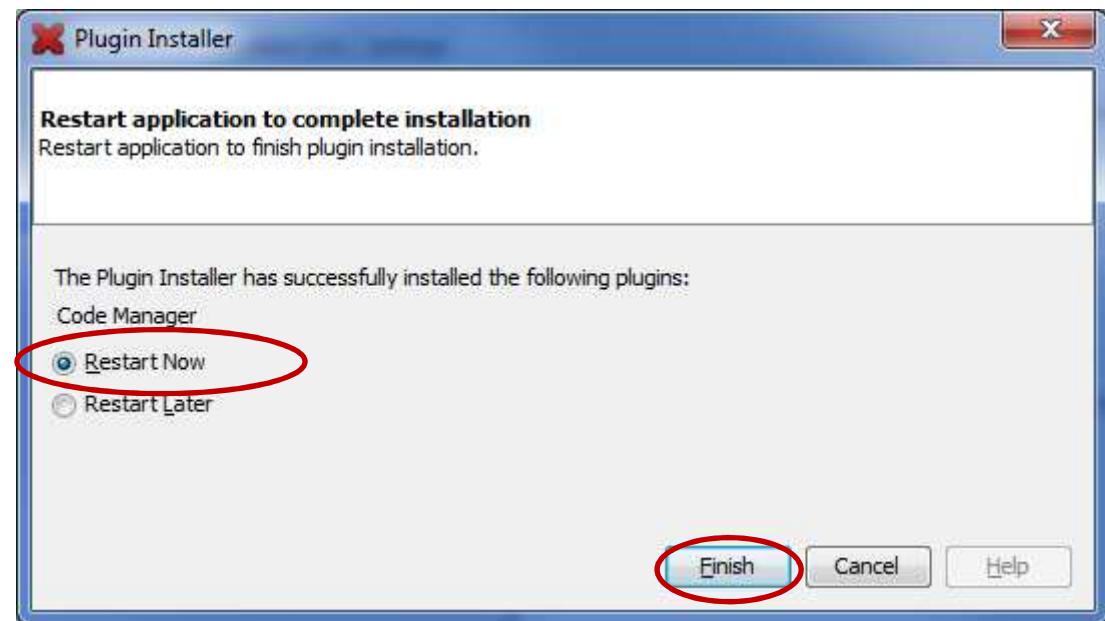
- Accept and Install License Agreement



Normal Plugin Installation

continued

- Select “Restart Now”
- Click Finish



- Proceed to the Labs

Trademarks

- The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC³² logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
- The Embedded Control Solutions Company is a registered trademark of Microchip Technology Incorporated in the U.S.A.
- Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KleerNet, KleerNet logo, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mTouch, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
- SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.
- GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.
- All other trademarks mentioned herein are property of their respective companies.
- © 2014, Microchip Technology Incorporated, All Rights Reserved.