

Shammah Thao

EEE 174 - CpE 185 Lab Section #2

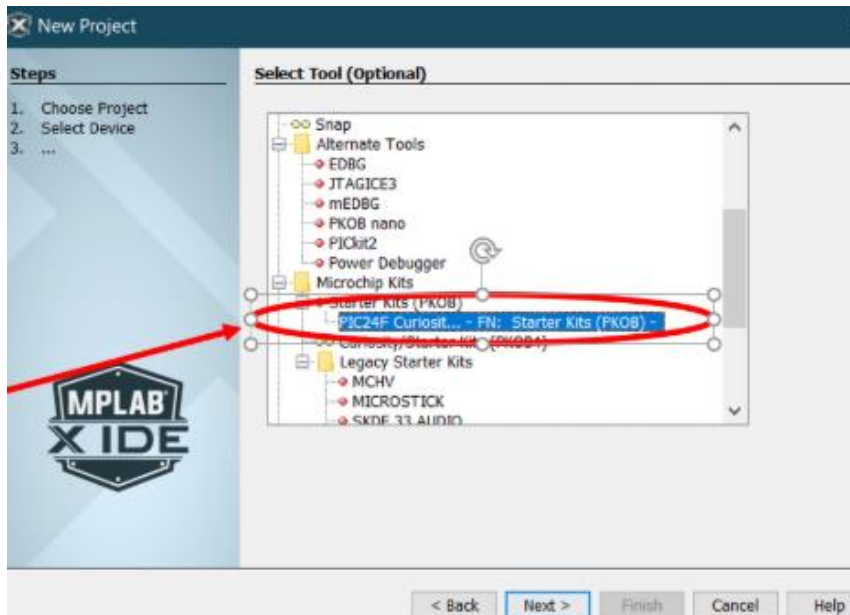
Monday & Wednesday

Lab 2: Microchip Development System

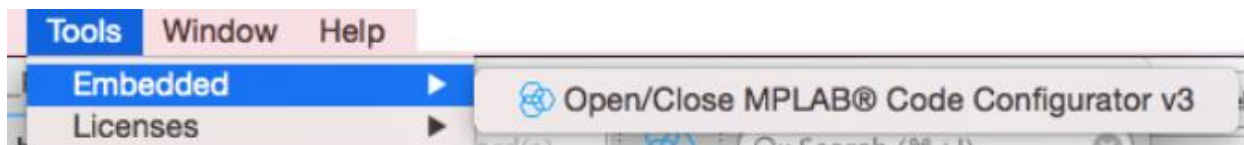
Dahlquist

## Part 1: 16-bit digital output Example

To start doing this lab we would have to start by downloading the software MPLAB X IDE, MPLAB XC 16 and install MPLAB Code Configurator. After that we would then have to open up MPLAB X IDE and start a new project.



After going through the process of creating the project, we then must open up the code configurator from the tools bar



Then after that we would have to set the pin module to be like the one from the example.

**Pin Module**

Easy Setup Registers  
Selected Package: TQFP44

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA9	Pin Module	GPIO	IO_RA9	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RA10	Pin Module	GPIO	IO_RA10	<input type="checkbox"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB0	ICD	PG01		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB1	ICD	PGC1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none

**(2) IO\_RA9 and IO\_RA10 will appear in list**

**(1) Click on the 'output' row under RA9 and RA10**

Output - MPLAB X IDE Code Configurator    Notifications [MCC]    Pin Manager: Grid View

Package: TQFP44    Pin No: 19 20 30 31 34 13 32 35 12 21 22 23 24 33 41 42 43 44 1 8 9 10 11 14 15 25 26 27 36 37 38 2 3 4 5

Function	Direction	Port A																Port B																Port C																					
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9																		
CLIO	input																																																						
CLIO	output																																																						
CLIO	input																																																						
OSCO	output																																																						
REFO	input																																																						
SOSC	input																																																						
SOSC	output																																																						
PGC	input																																																						
PGC	input																																																						
GPIO	input																																																						
GPIO	output																																																						

When the pin module matches to what was needed, we then generated the code from the left-hand side.

MPLAB X IDE v5.15 - MyProject: default

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

Project Resources    Generate    Import...    Export

**Generate**

Click on the **Generate** button

System

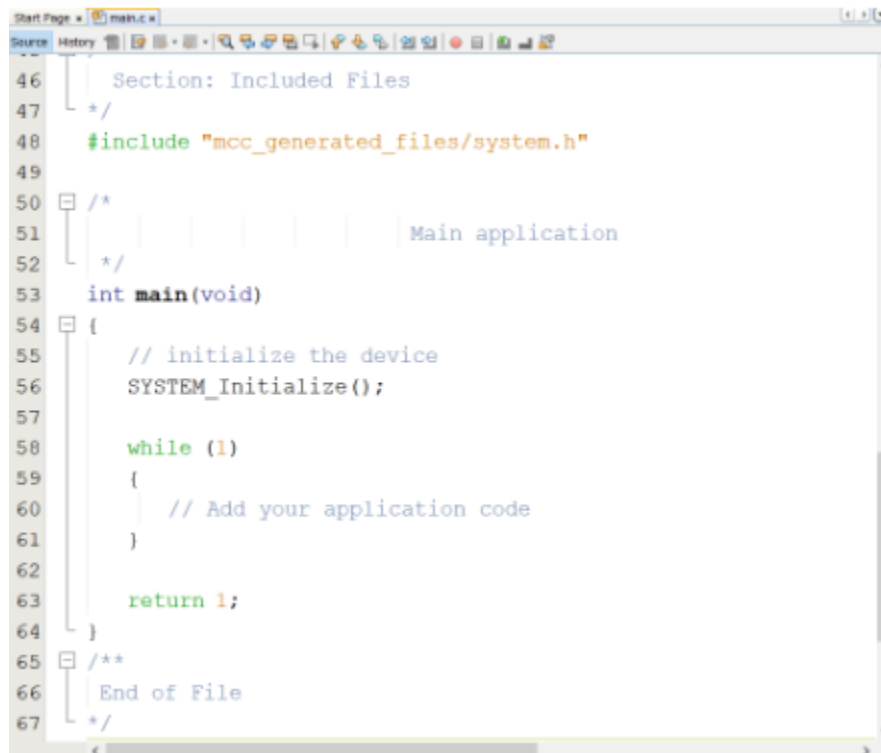
- Interrupt Module
- Pin Module
- System Module

Device Resources

PIC24F128GA204 Product Page

PIC24F128GA204

We then got the main file called MCC.h from the generated files, which we then added a given line of code to run.



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

After putting in the code and building the program, we should be able to see the led on the board change. The led 1 should be lit up constantly without touching it and led 2 should be blank with no light on.



Output			Notifications [MCC]			Pin Manager: Grid View ×																																		
Package: TQFP44			Pin No:			19	20	30	31	34	13	32	35	12	21	22	23	24	33	41	42	43	44	1	8	9	10	11	14	15	25	26	27	36	37	38	2	3	4	5
			Port A ▼										Port B ▼										Port C ▼																	
Module	Function	Direction	0	1	2	3	4	7	8	9	10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9			
Clock ▼	CLKI	input			🔒																																			
	CLKO	output				🔒																																		
	OSCI	input			🔒																																			
	OSCO	output				🔒																																		
	REFO	output																							🔒															
	SOSCI	input																																						
SOSCO	output																																							
ICD ▼	PGCx	input																																						
	PGDx	input																																						
Pin Module ▼	GPIO	input	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		
	GPIO	output	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒	🔒		

After we fix the pin module, we then generated the code again to get the mcc.h file and we modify that too with the code given to use in the instruction.

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/mcc.h"
/*
    Main application
*/
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    while (1)
    {
        if (S1_GetValue())
            LED1_SetLow();
        else
            LED1_SetHigh();
    }

    return 1;
}
```

After putting the code in, we then build the code and ran it, which gave us the following result. The led1 should be on when the button is pushed and led should be off no matter what. Led 1 only turn on when the button is pressed.





### Part 3 A: Programming the PIC24/dsPIC33 Timer

For this part we are implementing a timer into our code which would make the led flashes over time. We first build a new project then modify the pin modules again.

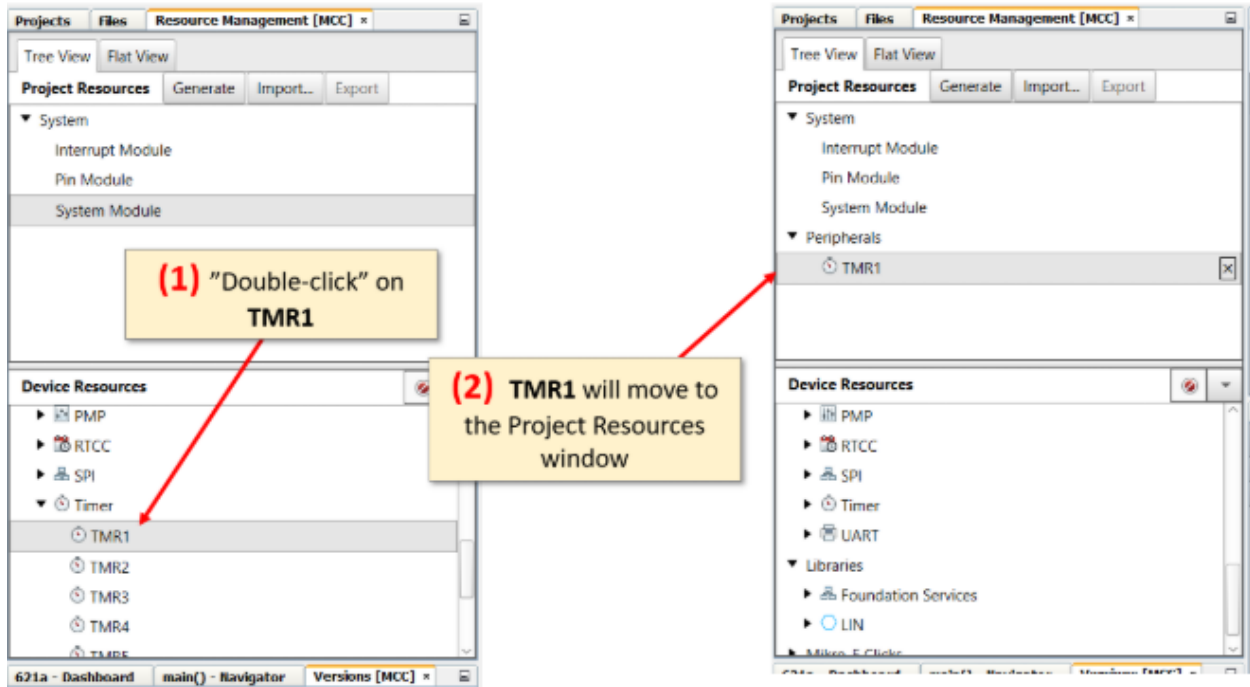
(1) Click on the 'output' row under RA9

(2) IO\_RA9 will appear in list

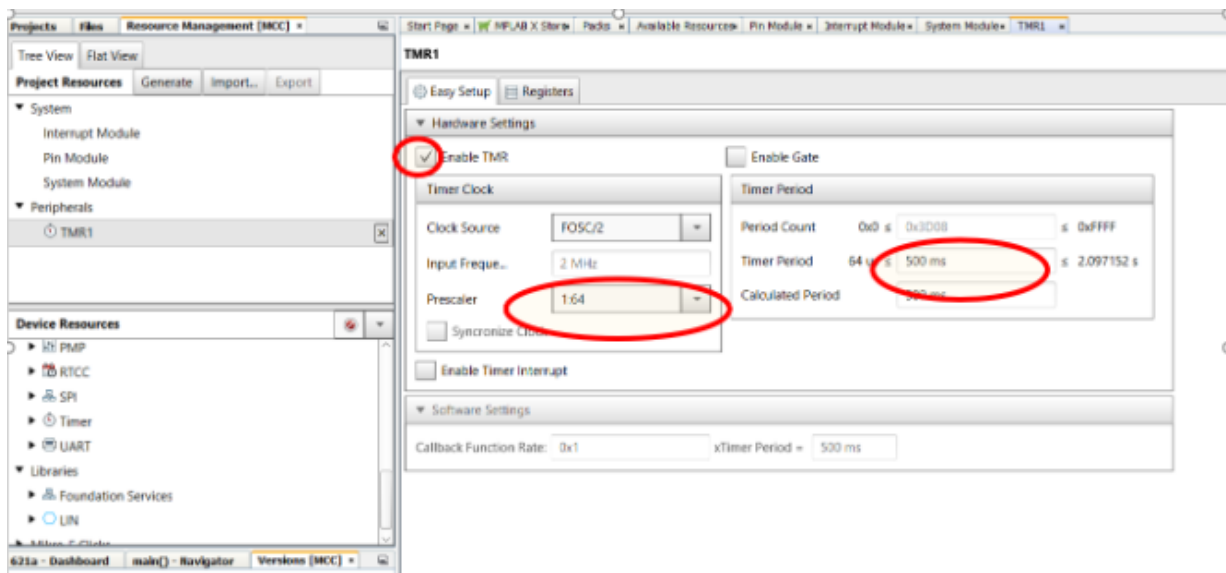
(3) Rename IO\_RA9 to LED1

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA9	Pin Module	GPIO	LED1			<input checked="" type="checkbox"/>				none
RB0	ICD	PGD1								none
RB1	ICD	PGC1								none

We then added the timer into the system and started setting it up.



We then set the timer to operate a certain interval and how flash it should work.



After this, we get the timer down we generate the code for the mcc.h file. Which added the following code to the main method.

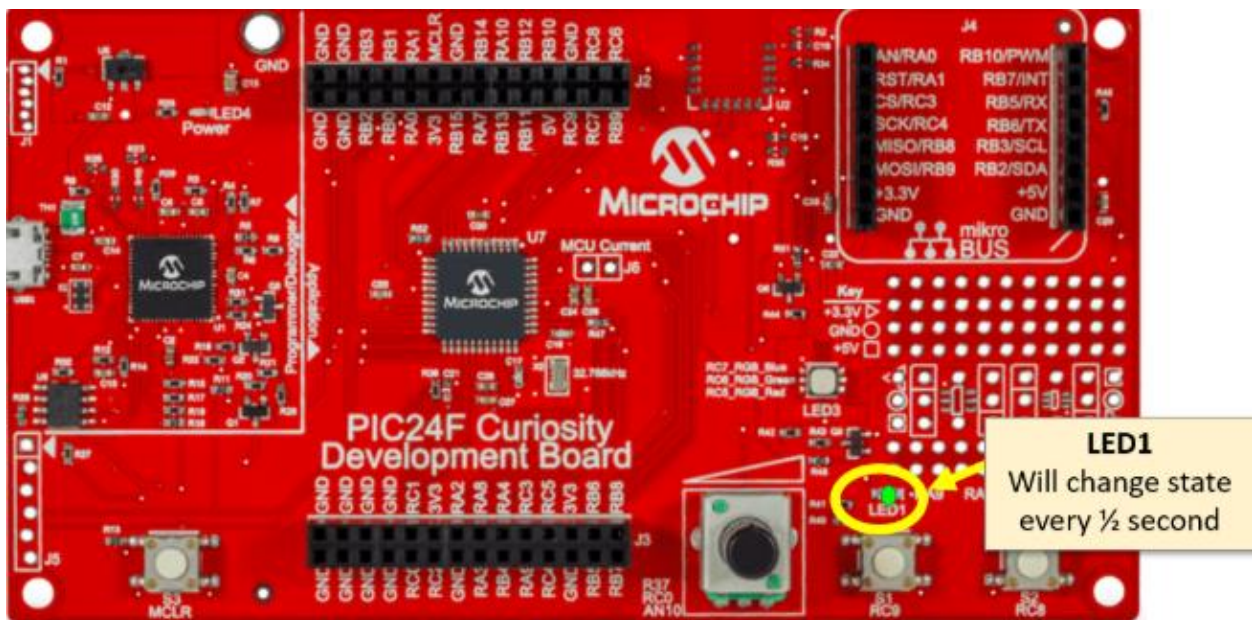


```

#include "mcc_generated_files/system.h"
#include "mcc_generated_files/mcc.h" //##### must be added #####
/*
    Main application
*/
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    TMR1_Start(); // ##### Start the timer ###
    while (1)
    {
        if (IFS0bits.T1IF)
        {
            LED1_Toggle();
            IFS0bits.T1IF = 0 ;
        }
    }
    return 1;
}

```

After this we build the program and ran it to show the result onto the board. It would show the LED light blinking at a constant rate without you needing to hold the button



### Part 3 B: Programming a PIC24/dsPIC33 Timer Using Interrupts

We had to do the same thing as the other part, where we make a new project, and modify the pin module to match the example.

Pin Module

Easy Setup

Selected Package: TQFP44

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA9	Pin Module	GPIO	LED1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB0	ICD	PGD0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none
RB1	ICD	PGC1		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	none

(2) IO\_RA9 will appear in list

(3) Rename IO\_RA9 to LED1

Click on the 'output' v under RA9

We then added the interrupts and also a timer into the program.

Projects Files Resource Management [MCC]

Tree View Flat View

Project Resources Generate Import... Export

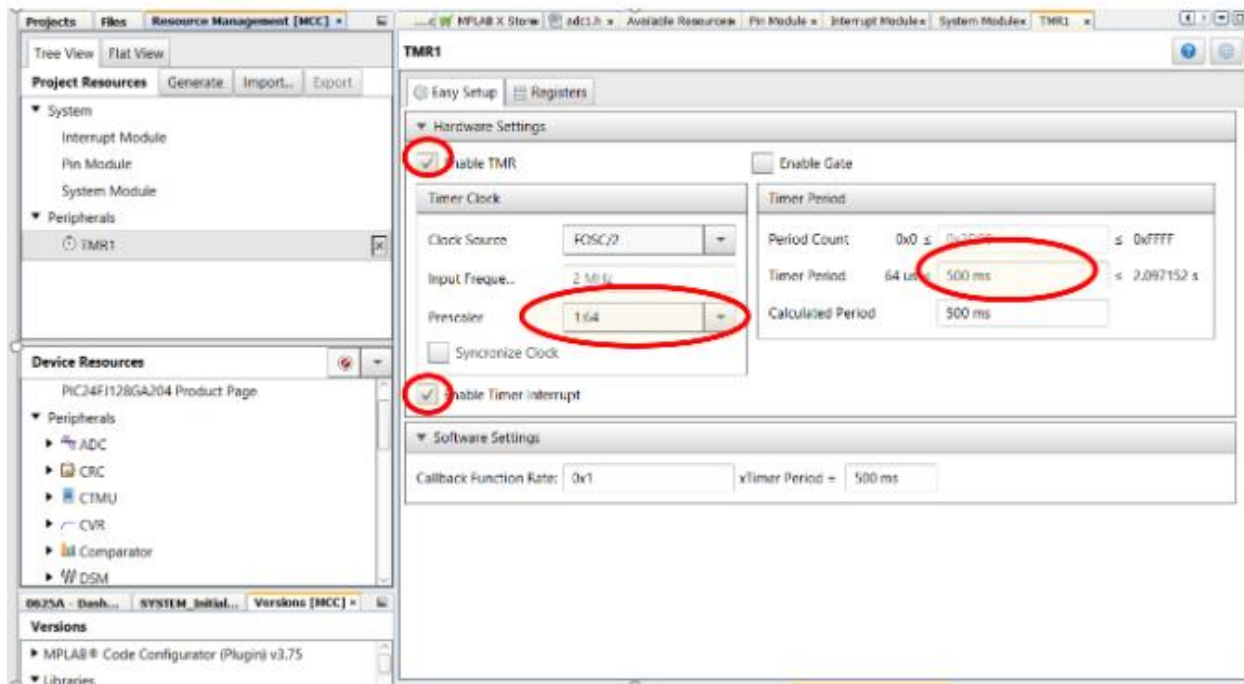
- System
  - Interrupt Module
  - Pin Module
  - System Module
- Peripherals
  - TMR1

Interrupt Module

Easy Setup

Interrupt Manager

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notificati...	19	<input type="checkbox"/>	1
TMR1	T1	T1 - Timer1	3	<input checked="" type="checkbox"/>	1



After which we then generated the code and added the give code into the main method of the mcc.h file

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/mcc.h" //##### must be added #####

void My_ISR(void) //#####
{
    LED1_Toggle();
}

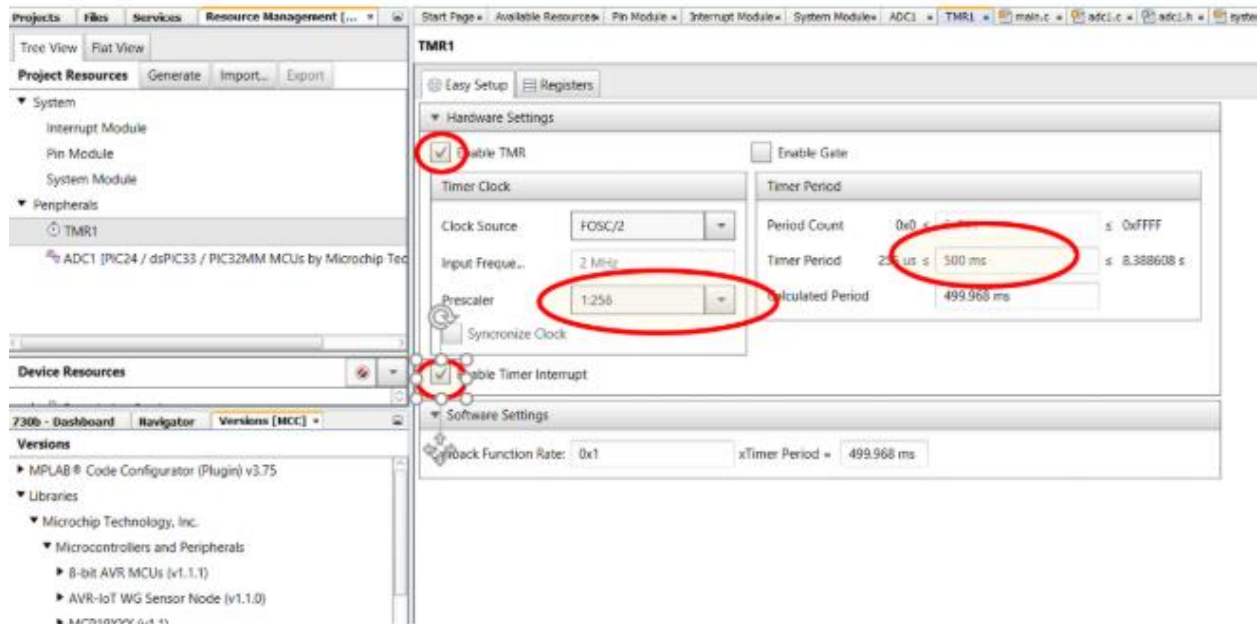
int main(void)
{
    SYSTEM_Initialize();
    TMR1_SetInterruptHandler(My_ISR) ; //#####
    TMR1_Start(); //#####
    while (1)
    {
    }
    return 1;
}
```

After building and running the code we would get the same result as Part 3 A, a constant blinking led that involves intervals.

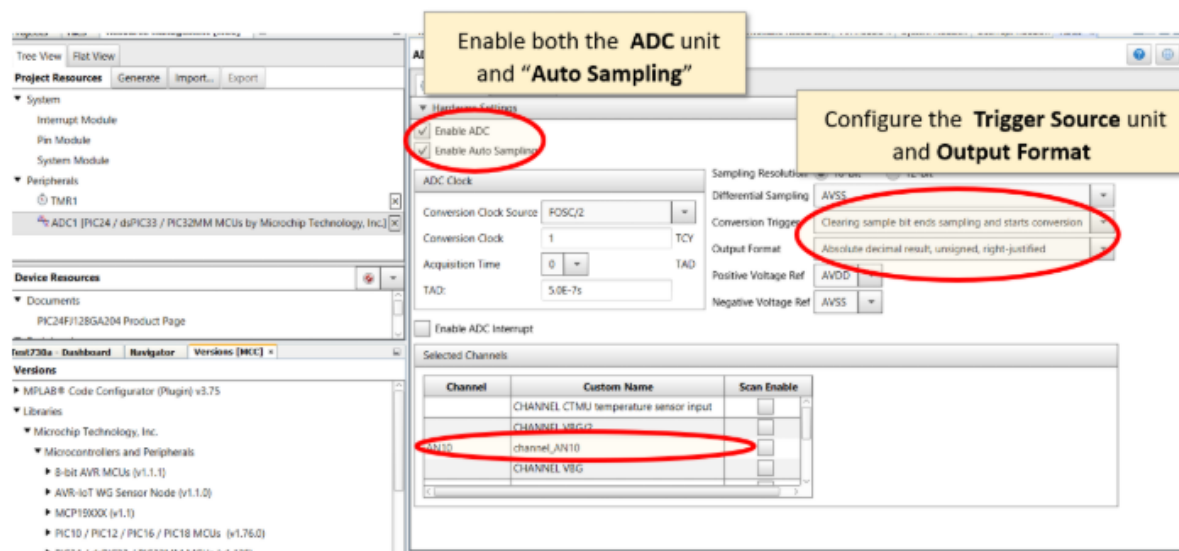




We then added the timer in.



We then configure the adc unit, adding it and enable it.



After this we then generate the code and then add the given code to our main method of the cc file.

```

#include "mcc_generated_files/system.h"
#include "mcc_generated_files/mcc.h" //##### must be added #####

void My_ISR(void)
{
    LED_Toggle();
    ADC1_SoftwareTriggerDisable();           // trigger next conversion
}

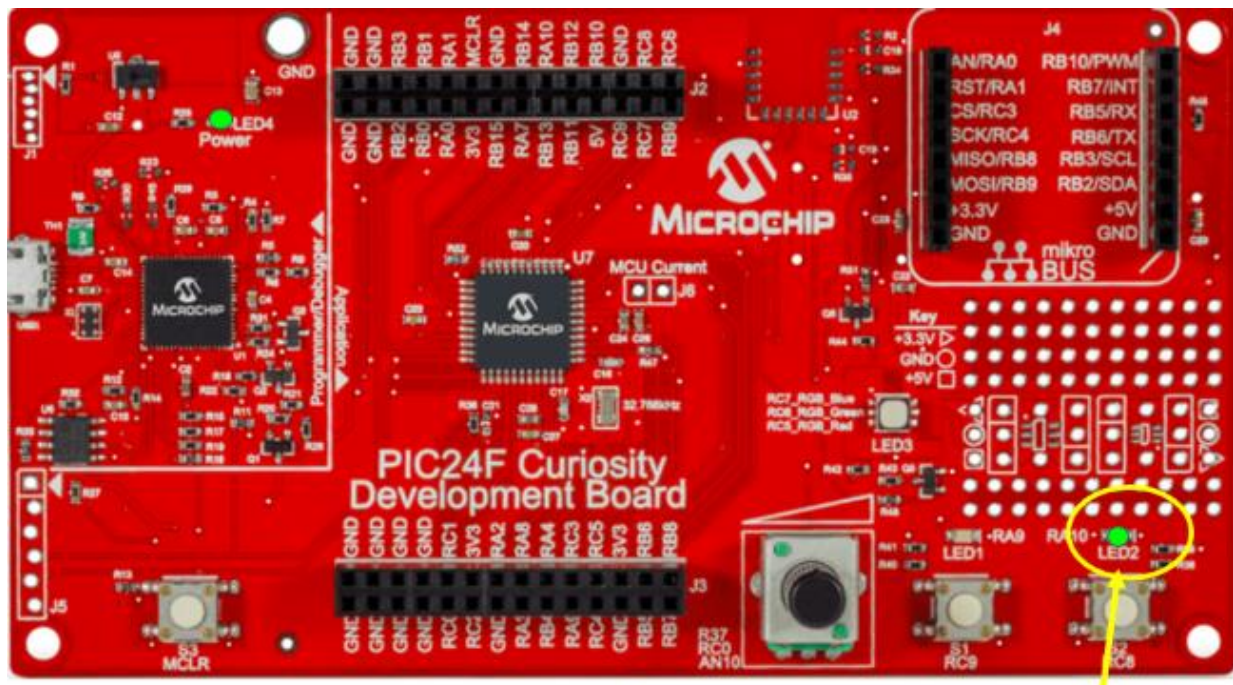
int main(void)
{
    SYSTEM_Initialize();
    ADC1_ChannelSelect(channel_AN10);
    ADC1_SoftwareTriggerDisable();           // begin first conversion
    TMR1_SetInterruptHandler(My_ISR);
    TMR1_Start();
    while (1)
    {
        if (ADC1_IsConversionComplete(channel_AN10)){
            if (ADC1_ConversionResultGet(channel_AN10)==0)
                TMR1_Period16BitSet(1);
            else
                TMR1_Period16BitSet(ADC1_ConversionResultGet(channel_AN10));
            // application code goes here
        }
    }
    return 1;
}

```

After building the program to check if there is any error, we then run the program to see the result.

We should be able to control how fast the led should be blinking based on the POT





**LED** blinks based at a frequency determined by the **POT**

Conclusion:

For this lab, it was straight forward and repetitive, but because of this repetitiveness I was able to learn more about the PIC24F board. It has interesting feature the we would be able to mess around with.