An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Computer Engineering Department
Microcontroller Lab (10636496)
Report Grading Sheet

Instructor Name: Hikmat Drawsheh	**Experiment: 2**
Academic Year: 2024/2025	**Performed on:**
Semester:	**Submitted on:**
Student Names:	
1- Sadeen Hawash	2- Walaa Ashqar
3-	4-
5-	6-

Evaluation Criterion	CLO	Grade	Points
Abstract and Aims Aims and idea of the experiment are clearly stated in simple words		10	
Introduction, Apparatus and Procedures Introduction is complete and well-written, all grammar/spelling correct, Appropriate background information related to the principles of the experiment is provided. The list of apparatus and procedures are also provided		15	
Experimental Results, Calculations and Discussion Results analyzed correctly. Experimental findings adequately and specifically summarized, in graphical, tabular, and/or written form. Comparison of theoretical predictions to experimental results, including discussion of accuracy and error analysis as needed.		50	
Conclusions Conclusions summarize the major findings from the experimental results with adequate specificity. Highlighting the most important results		15	
Appearance Title page is complete, page numbers applied, content is well organized, correct spelling, fonts are consistent, good visual appeal. You have also to use reference for the information you provide		10	
Total		100	

Table of Contents

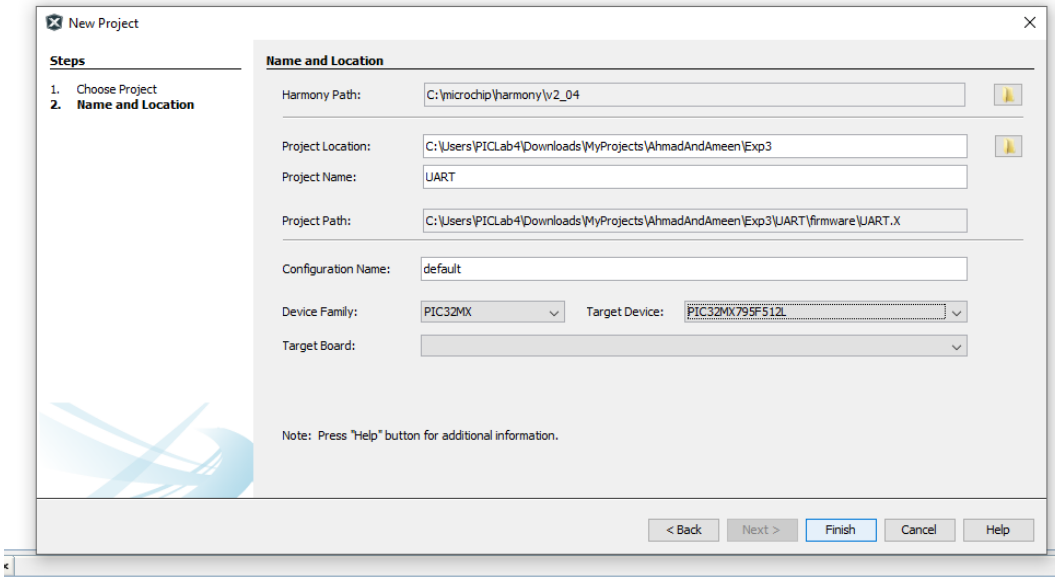
Contents

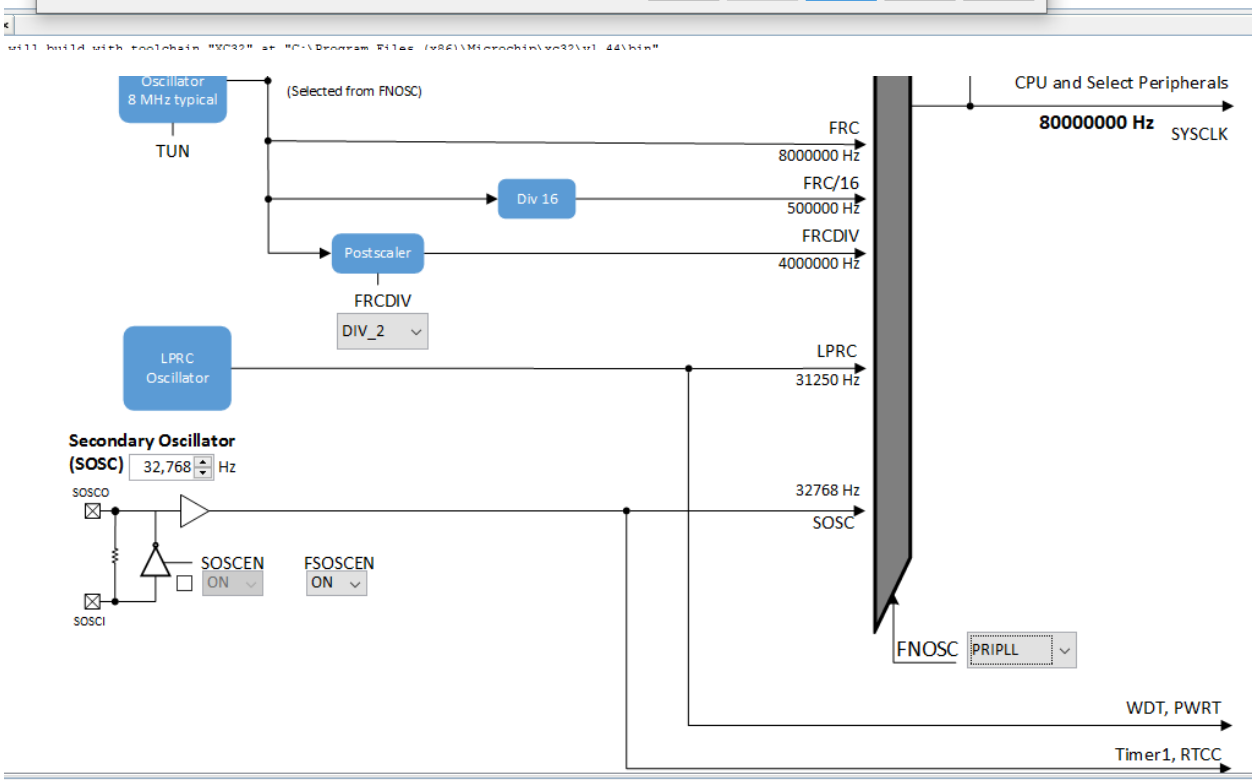
Abstract

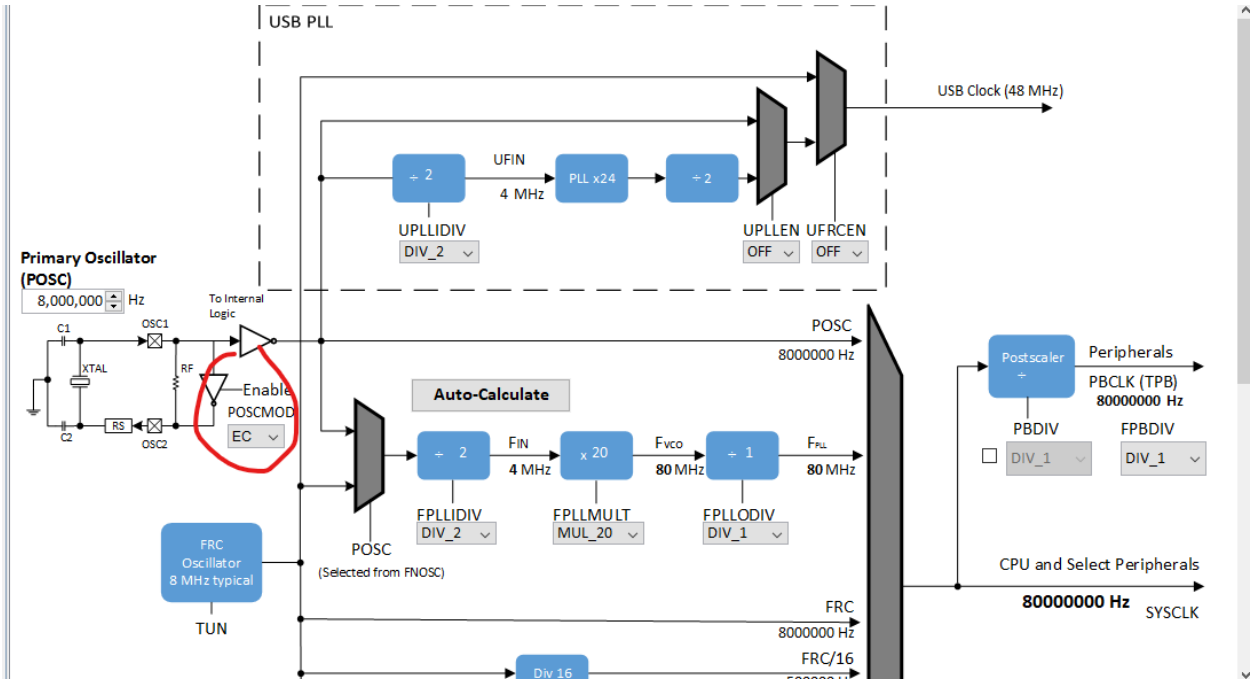
The purpose of this experiment is to familiarize the students with the methods of Input/Output (I/O) control of PICTM32 microcontroller. This includes reading data from input pins and writing to output.

Introduction

First, we need to create a new harmony project and choose the kit on it, then we need to setup the external clock and select input clock frequency with 80 MHz by select PRIPLL from input of multiplexer.


New Project
Steps
1. Choose Project
2. Name and Location
Name and Location
Harmony Path:
C:\microchip\harmony\v2_04
Project Location:
C:\Users\PICLab4\Downloads\MyProjects\AhmadAndAmeen\Exp3
Project Name:
UART
Project Path:
C:\Users\PICLab4\Downloads\MyProjects\AhmadAndAmeen\Exp3\UART\firmware\UART.X
Configuration Name:
default
Device Family:
PIC32MX
Target Device:
PIC32MX795F512L
Target Board:
Note: Press "Help" button for additional information.
< Back
Next >
Finish
Cancel
Help


Oscillator
8 MHz typical
TUN
(Selected from FNOSC)
FRC
8000000 Hz
Div 16
FRC/16
500000 Hz
Postscaler
FRCDIV
4000000 Hz
FRCDIV
DIV_2
LPRC
Oscillator
LPRC
31250 Hz
Secondary Oscillator
(SOSC)
32,768
Hz
SOSCO
SOSCI
SOSCEN
ON
FSOSCEN
ON
32768 Hz
SOSC
CPU and Select Peripherals
80000000 Hz
SYSCLK
FNOSC
PRIPLL
WDT, PWRT
Timer1, RTCC



The experiment is to define a counter and display its value on the Leds, and the second part is to display leds by pressing certain buttons.

Materials

- ChipKITTM Pro MX7 processor board with USB cable.
- Microchip MPLAB R X IDE.
- MPLAB R XC32++ Compiler.
- MPLAB Harmony Framework.

Methods

We define a variable called counter that have initial value of 0,
Then we increment its value in the state of the program.

And in the second part we checked that value of button 1 and 2 and depending at their values we can light certain Leds.

Experimental Results

First, we initiate the state for the parts (Counting and IO) in the app.h file as shown:

```
typedef enum {
	APP_STATE_INIT=0,
	APP_STATE_SERVICE_TASKS,
	APP_STATE_COUNTER,
} APP_STATES;
```

Then we initiate the counter in app.h and give its initial value in app.c in the init state as shown:

```
typedef struct {
	APP_STATES state;
	int counter;
} APP_DATA;
```

Then we write the code for the counter to display its value to leds in state counter and we shift it by 12 to display the values in RG12 – RG15 when btn1 clicked:

```
case APP_STATE_INIT:
{
        bool appInitialized = true;
        myCounter=0;
        bool btn1 = PLIB_PORTS_PinGet(PORTS_ID_0,
        PORT_CHANNEL_G, PORTS_BIT_POS_6 );
        if (appInitialized) {
                if(btn1) appData.state = APP_STATE_COUNTER;
        }
        break;
}
case APP_STATE_COUNTER:
{
        PORTG= appData.counter <<12;
        int i;
        for (i=0; i<50000000; i++);
        appData.counter++;
        break;
}
default:
{
        break;
}
```

We added an empty loop just to act as a delay to give it some time to appear in the leds.

The next part we wrote a code to flash the leds when btn2 pressed:

```
case APP_STATE_SERVICE_TASKS:
{
        bool btn1 = PLIB_PORTS_PinGet(PORTS_ID_0,
        PORT_CHANNEL_G, PORTS_BIT_POS_6 );
         bool btn2 = PLIB_PORTS_PinGet(PORTS_ID_0,
        PORT_CHANNEL_G, PORTS_BIT_POS_7 );
        if (btn1 && btn2) {
                LATG = 0x8000;
        } else if (btn2) {
                LATG = 0x4000;
        } else if (btn1) {
              LATG = 0x2000;
         } else {
             LATG = 0x1000;
         }
         break;
  }
```

Discussion

In this experiment we found we can read values from buttons and then depending on their values we can light certain leds and also, we can make a counter and display the values into leds.

Conclusion

We learned how to set up a new MPLAB project with a header and source file. Next, we learnt how to work with buttons and leds. We read button values and show certain leds based on those values, and we also displayed a 4-bit counter on output LEDs. Furthermore, we found that in two of the situations we executed, a delay was required to see the outcome accurately and clearly.