An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا

**Computer Engineering Department**
**Microcontroller Lab (10636496)**
# Report Grading Sheet

| | |
|---|---|
| **Instructor Name: Hikmat Darawsheh** | **Experiment: 7** |
| **Academic Year: 2024** | **Performed on:** |
| **Semester:** | **Submitted on:** |
| **Student Names:** | |
| 1- Wala' Essam Ashqar | 2- Sadeen Hawash |
| 3- | 4- |
| 5- | 6- |

| Evaluation Criterion | CLO | Grade | Points |
|---|---|---|---|
| **Abstract and Aims**<br>Aims and idea of the experiment are clearly stated in simple words | | 10 | |
| **Introduction, Apparatus and Procedures**<br>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**<br>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**<br>Conclusions summarize the major findings from the experimental results with adequate specificity. Highlighting the most important results | | 15 | |
| **Appearance**<br>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 | |

# Contents



## Abstract

In this experiment we learn about interrupts and timers and we used them on Button 1 and Button 2 to switch between two modes: the first mode is a counter that counts 0 – 15 and the second mode is flashing mode that flashes all the Leds every while

## Introduction

By turning on the CN (change notice) for a particular pin, we can use interrupts in PIC32 code. This means that rather than writing a blocking code (for loop) the code can continue running and whenever this pin changes, it will trigger an interrupt, which will halt the code's execution and switch to the interrupt service routine (ISR), which we should implement to handle the interrupt.

## Materials

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

## Methods

To conduct this experiment, we needed to complete the following stages through many steps: We made a new project and adjusted the pin settings accordingly (RG6, RG7) to enable CN. After correctly setting the clock, we established an instance of the timers (using timer 0) and enables interrupts.

## Experimental Results

First, we defined the new states and define the new variables in the app.h as follows:

```
typedef enum
{       APP_STATE_INIT=0,
APP_STATE_SERVICE_TASKS,
   TIMER_INTERRUPT,
   CN_INTERRUPT,   /* TODO: Define states used by the application state machine. */
} APP_STATES;
typedef struct
{    APP_STATES state;
   int count;    int flag;
   /* TODO: Define any additional data used by the application. */
} APP_DATA;
```

Then in the initialize we set the counter to zero and start timer0 in app.c:

```
case APP_STATE_INIT:
    {
        appInitialized = true;
        if (appInitialized)
        {
            appData.count= 0;
            DRV_TMR0_Start();
            appData.state = APP_STATE_SERVICE_TASKS;
        }
        break;
    }
```

In the count states we increment the bit 12 of the G which is the start of the leds to have a counter and in the flashing state we toggle the values of the last 4 bits of G which are the Leds (12-15) and for each of the states when it does the operation it then transfers to main state.

```
case APP_STATE_SERVICE_TASKS:
    { break;    }
    case TIMER_INTERRUPT:
    {  if(appData.count==15) appData.count=0;
       appData.count++;
       LATG=appData.count<<12;
       appData.state = APP_STATE_SERVICE_TASKS;
       break;    }
    case CN_INTERRUPT:
    {  LATG=0XFFFF;
       for(i=0;i<5000000;i++);
       LATG=0X0000;
       appData.state = APP_STATE_SERVICE_TASKS;
       break;
    }      /* TODO: implement your application state machine.*/
     /* The default state should never be executed. */
    default:
    {        /* TODO: Handle error in application's state machine. */
       break;    }
```

In the system_interrupt.c we configure the code when an interrupt happens on the buttons and what to do every clock cycle as follows

```
APP_DATA appData;
void __ISR(_CHANGE_NOTICE_VECTOR, ipl1AUTO)_IntHandlerChangeNotification(void)
{
   /* TODO: Add code to process interrupt here */
   if(PORTGbits.RG6==1)
   {       appData.flag=2;    }
   if(PORTGbits.RG7==1)
   {
      appData.flag=1;
   }
   // appData.state=CN_INTERRUPT;
   PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_CHANGE_NOTICE);
}
void __ISR(_TIMER_3_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance0(void)
{    if(appData.flag==2)
      appData.state = TIMER_INTERRUPT;
   if(appData.flag==1)
      appData.state = CN_INTERRUPT;
   PLIB_INT_SourceFlagClear(INT_ID_0,INT_SOURCE_TIMER_3);
}
```

## Discussion

We used interrupts and timers to handle the changes that happens to the buttons instead of writing blocking code to check the buttons.

## Conclusion

This lab taught us about timers, how to set them up, and how to do simple math operations to determine the desired duration. Additionally, we studied interrupts and their purposes. We utilized Change Notice for two buttons and combined all of them into a basic program.