NORFOLK STATE UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRONIC ENGINEERING

A PROJECT REPORT ON

IMPLEMENTATION OF INTERRUPT-DRIVEN LED BLINKING USING MICROCONTROLLER

BY

EMMANUEL ABOAH BOATENG

COURSE: MICROCONTROLLERS

NOVEMBER, 2018

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1 Project Overview

This is a project report on the implementation of the blinking of two LEDs with a switch and connected to a PIC24 microcontroller. A scenario for the description of the project was such that, the two LEDs namely; LED1 and LED2 were designed to be initially blinking, and whenever the switch (SW1) is pressed and released, the two LEDS are supposed to alternate between terminating the blinking and resuming the blinking [1].

There are two major approaches to the solution of the problem elaborated above, which are as follows;

1. The use of Polled I/O instructions for switching within a while (1) loop with two states for each press and release action.
2. The use of Periodic Interrupt and change notification interrupt for sampling the switch input (SW1).

The periodic interrupt approach was adopted in this project because it provides a more efficient way for solving the problem by making use of interrupts to reduce the CPU cycles and also conserve power by avoiding the polling of I/Os.

1.2 Objective of this Project

The objective of this project report is to:

- Provide the details of the implementation of an interrupt-driven LED blinking using a microcontroller

1.3 Resources Employed

The resources employed in doing this project include:

- Review of relevant literature
- Use of Visio software for the design of basic diagrams
- Use of PIC24FJ64GB002 Microstick
- Use of MPLAB software for programming the Microstick
- Utilization of the electronics lab for project implementation and testing

1.4 Work Organization

This project is accessible in four chapters. Chapter 1 introduces the entire project: problem definition, objectives of the project, methods used and work organization. Chapter 2 presents the methodology in design of the project. Chapter 3 gives the results and discussions. Chapter 4 ends with conclusions and recommendations

METHODOLOGY

2.1 Design Procedure

The central idea of this project was to implement an efficient LED blinking scenario using change notification and timer interrupts. The change notification interrupt detects whenever the switch is pressed and released, and the Central Processing Unit (CPU) is run whenever there is a change notification from the press and release of the switch button. Timer 2 interrupt was configured to cause the delay. The states behavior for each press and release of the switch are outlined below.

2.1.1 Switch Button Pressed – State 1

Initially, the two LEDs (LED1 and LED2) were programmed to be blinking with serial pulses. Whenever the switch is pressed, a change notification interrupt of higher priority is set and this causes the Interrupt Service Routine (ISR) of the interrupt to be executed. The resulting effect is, whenever the switch is pressed, the blinking behavior of the LEDs are terminated.

2.1.2 Switch Button Released – State 2

Consequently, when the switch is released, ISR is supposed to disable the change notification (CN) interrupt and clear the CN flag. The resulting effect is; the two LEDs return to their initial mode of blinking.

2.2 Procedure for Blinking LEDs

The two LEDs (LED1 and LED2) were connected to the RB1 and RB4 pins of the microcontroller respectively through resistors. The pushbutton switch for triggering the change notification interrupt was also connected to RB13 pin. The initial blinking state of the LEDs were implemented by the use of timer interrupt as outlined below.

2.2.1 Timer 2 Configuration to Produce Interrupt

The *configTimer2()* function was employed to configure the timer 2 for 16-bit mode with a prescale of 64, set the PR2 to 15 ms, and enable the timer 2 interrupt. The timer 2 interrupt generates a square wave on the RB1 and RB4 ports hence the corresponding interrupt service routine blinks (toggles) the two LEDs on each interrupt and clears the T2IF flag before returning. By so doing, the interrupt service routine performs all the functions of blinking the LEDs by producing the square waves and hence the *while (1)* loop has only the *doHeartbeat()* function as seen in Appendix A.

Fig. 2.1 Shows the PIC24 Microstick design for implementation of the project.

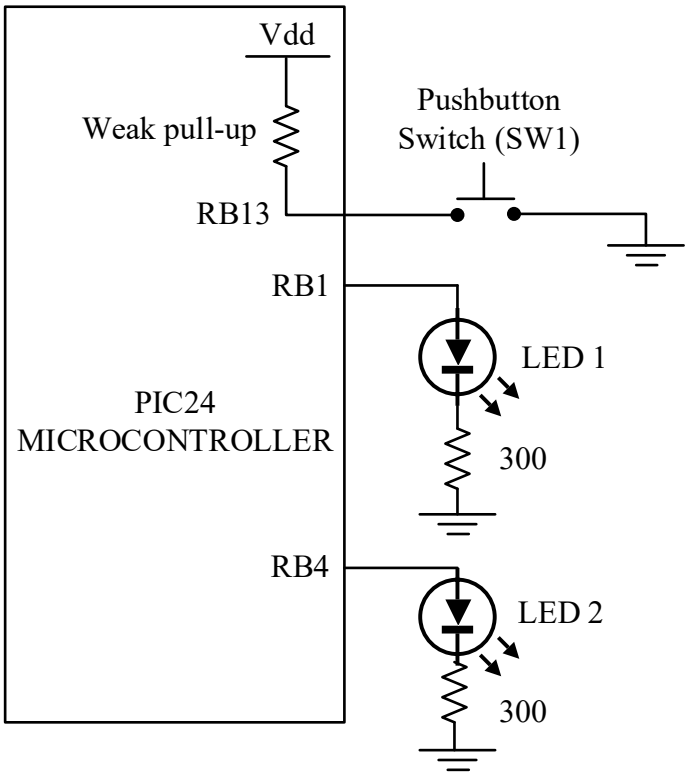

Fig. 2.1 Components Connections to PIC24 Microcontroller

2.2.2 Weak Pull-up and External Resistance Determination

The internal weak pull-up was enabled on pin RB13 which is connected to the pushbutton switch. This eliminated the need for an external pull-up resistor on the input switch (SW1) and as a result the pull-up resistor acts as a high impedance P-transistor [1]. This allows the pushbutton input to be read as either 1 or 0 whenever it is pressed.

The appropriate resistance value was determined in order to get the required current to maximize the illumination of the LEDs. The selected LEDs have a voltage drops of 2 V and it was known that PIC24uC can provide a maximum current of 5 mA for a supply voltage of about 3.3 V [1]. Therefore, by Ohm's law, V = IR, hence computing the resistance value from the equation, R = V/I, provides an output of, R = (3.3 – 2.0)/0.005 = 260 Ohms. This resistance value is the lowest resistance allowed for maximum current. Finally, in order to give some margin of safety, a 300 Ohms resistor was selected.

2.3 Software Implementation of the Project

The software program execution and processes involved in the implementation of the project are summarized in the flow chart of Fig. 2.2. At start, the timer2 interrupt is being executed in order to resume the blinking. Whenever the switch is pressed, because of its higher interrupt priority than the timer interrupt, its ISR is executed to terminate the blinking. Alternately, when the switch is released, the CN flag is cleared and the timer2 interrupt is executed for the LEDs to resume blinking.

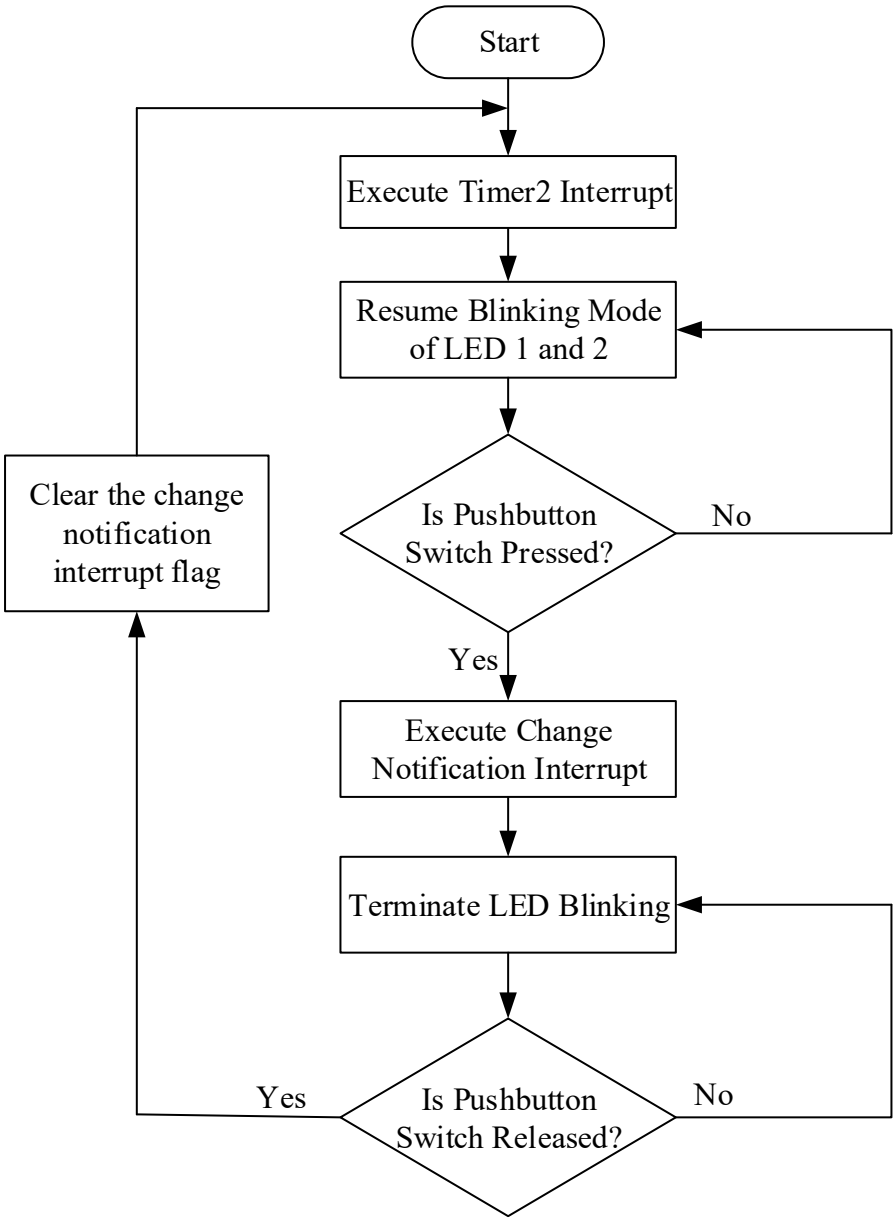

Fig 2.2 Flowchart for the Implementation of the Project

2.4 Hardware Implementation

After successfully building and running the code, the project was implemented in a hardware format at the Electronics Laboratory for testing and demonstration. The basic electronic components used for the implementation of the project are:

1. Breadboard and connecting wires
2. Two LEDs and three 300 Ohms resistors.
3. Pushbutton switch
4. Microstick

The MPLAB codes were built and uploaded onto the microcontroller for execution. Fig 2.3 depicts a snapshot of a working demonstration of the project. The states at which a press and release of the pushbutton switch caused an alternated termination and resumption of the blinking of the LEDs are shown in Fig. 2.3 and 2.4.

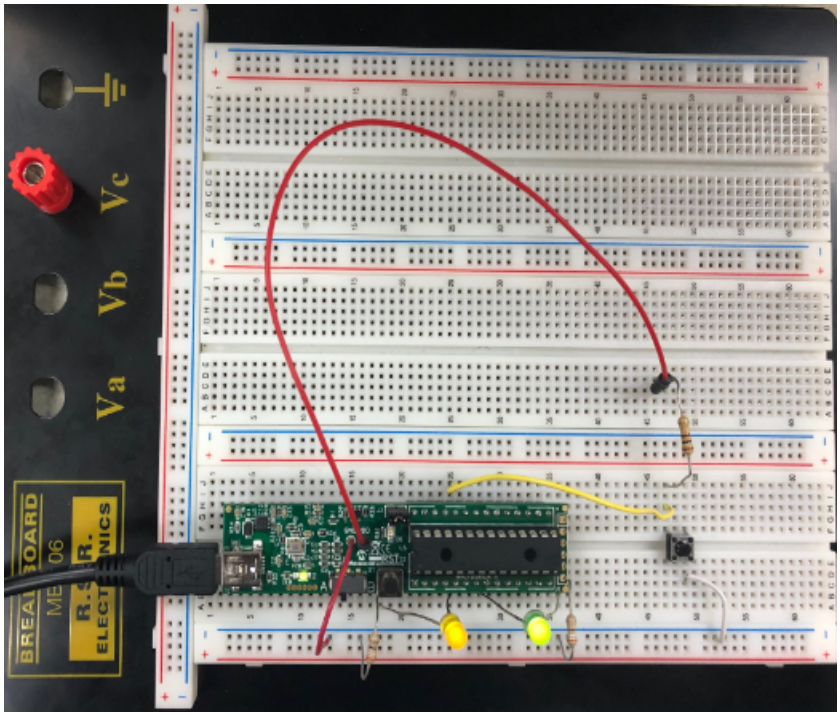

Fig. 2.3 Blinking Modes of the LEDs

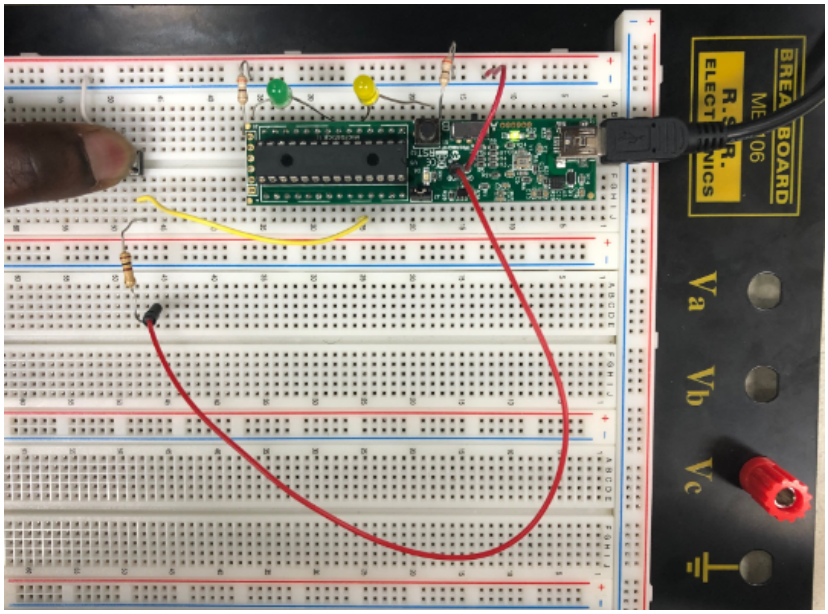

Fig. 2.4 LEDs Blinking Terminated

RESULTS AND DISCUSSIONS

3.1 Results and Discussions

The codes were successfully run in MPLAB without any errors and consequently downloaded on to the microcontroller for testing. Initially, on each timer interrupt, the output digital pins RB1 and RB4 are being toggled to generate a square wave and afterwards the interrupt flag is cleared in the ISR. This repeated process causes the LEDs to continuously blink.

The pin RB13 which was enabled for change notification was connected to the switch. The interrupt priority of the change notification was set to be higher than the timer interrupt (*_CNIP = 2*). As a result, whenever the pushbutton is pressed, the CN interrupt is executed and the current states of the LEDs terminated. On each release of the pushbutton, the change notification flag is cleared and hence it returns from the interrupt instructions. The LEDs then resumed the blinking.

3.2 Challenges

The major challenge faced in implementing this project was that, initially, when the hardware was assembled and the switch was pressed, it kept bouncing. It was therefore not able to hold the terminated blinking mode of the LEDs but rather the LEDs would blink irregularly. But the internal weak pull-up had been enabled. In order to curb this bouncing effect of the switch, an external weak pull-up of high resistance value (470 Ohm) was connected across the switch and the microcontroller was reset. The circuit worked perfectly on each press and release of the pushbutton switch as expected.

CONCLUSION

In conclusion, the primary objective of this project was successfully achieved. Timer2 interrupt was used to generate square waves to the digital output pins of the microcontroller in which the LEDs are connected to cause the LEDs to be blinking. The instructions were implemented in MPLAB Software. A change notification interrupt was enabled on a digital input pin connected to a switch with both internal and external weak pull-ups enabled. The hardware was assembled using components at the electronics laboratory of Norfolk State University, according to logic and instructions used in writing the program. The corresponding program was written in MPLAB IDE, simulated and downloaded onto the microcontroller. The resulting project ran successfully according to the design instructions. Finally, each press and release of the pushbutton switch caused a change between terminating the blinking and resuming the blinking.

REERENCES

[1] B. A. Jones, R. Reese, and J. W. Bruce, *Microcontrollers: From Assembly Language to C Using the PIC24 Family*, Cengage Learning Inc., USA, pp. 317 – 342.

[2] M. Deo, "*Interrupts and Timers*", Unpublished Lecture Material, Norfolk State University, 33pp.

APPENDIX A

MPLAB CODES FOR PROJECT EXECUTION

```
/*
 * Author: Emmanuel
 *
 * Created on November 15, 2018, 9:59 PM
 */
#include<p24FJ64GB002.h>

// Configuration of Output for LED pins
#define LED1 (_LATB1)
#define LED2 (_LATB4)
void config_LEDsoutput(){
    CONFIG_RB1_AS_DIG_OUTPUT();
    CONFIG_RB4_AS_DIG_OUTPUT();
}
// Configuration of the Pushbutton Switch
void config_SW1() {
    CONFIG_RB13_AS_DIG_INPUT();
  ENABLE_RB13_PULLUP();
  DELAY_US(1); // Give the pullup some time to take effect.

  //Change Notification (CN) Interrupt Configuration
  void config_cn(void) {
  ENABLE_RB13_CN_INTERRUPT();
  // Clear the interrupt flag.
  _CNIF = 0;
  // Choose a priority.
  _CNIP = 2;
  // Enable the Change Notification general interrupt.
  _CNIE = 1;

// Define the time, in ms, between Timer2 interrupts.
#define ISR_PERIOD  (15)



void  configTimer2(void) {
```

```
  T2CON = T2_OFF | T2_IDLE_CON | T2_GATE_OFF
        | T2_32BIT_MODE_OFF
        | T2_SOURCE_INT
        | T2_PS_1_64;
  PR2 = msToU16Ticks(ISR_PERIOD, getTimerPrescale(T2CONbits)) - 1;
  TMR2  = 0;
  // Enable Timer2 interrupts.
  _T2IF = 0;
  _T2IP = 1;
  _T2IE = 1;
  // Start the timer only after all timer-related configuration is complete.
  T2CONbits.TON = 1;
}
// Interrupt Service Routine (ISR) for Timer2.
void _ISR _T2Interrupt(void) {
  // Toggle the output pin to generate a square wave.
  LED1 = !LED1;
  LED2 =! LED2;
  _T2IF = 0;
}
// Change notification ISR Configuration
void _ISR _CNInterrupt(void) {
  T2CONbits.TON=!T2CONbits.TON;
 _CNIF = 0;

}
int main (void) {
    configTimer2();
    config_LEDsoutput();
    config_SW1();
    Config_cn();

 // The interrupt does work of generating the square wave
  while (1) {
    doHeartbeat();                  // Blink the heartbeat LED to show that the PIC is running.
  }
}
```