Student Name	蔡汉霖
Fuzhou University ID	*832002117*

Student Name	卿祺果
Fuzhou University ID	*832002127*

Report for EE302FZ Lab 6

Introduction

The following Figure 1 depicts a basic configuration of an embedded system based around the PIC16F877A. And Figure 2 below depicts a system overview of the embedded system and a bidirectional connection to a PC.

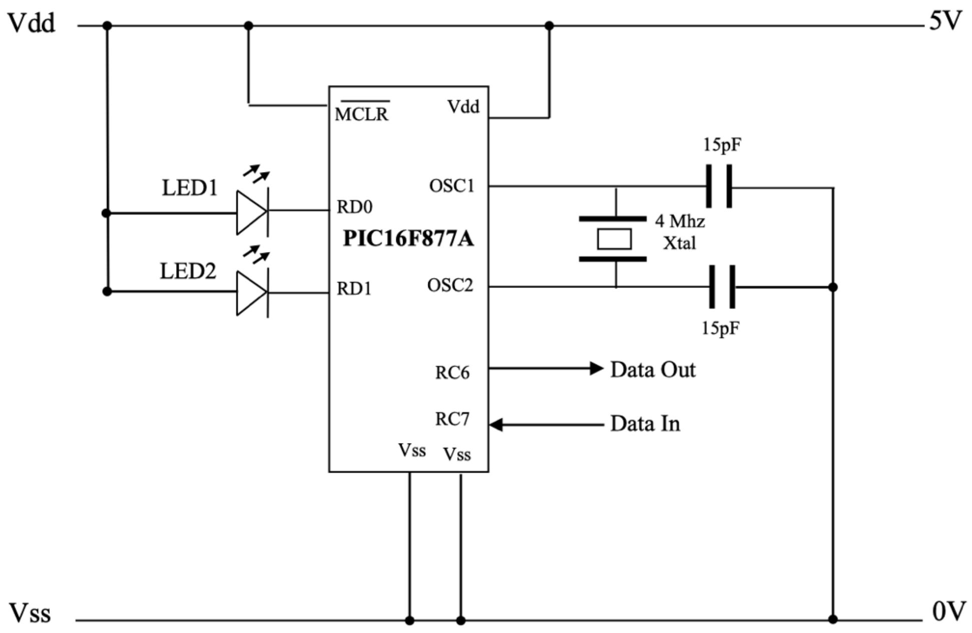


Figure 1 Basic Configuration

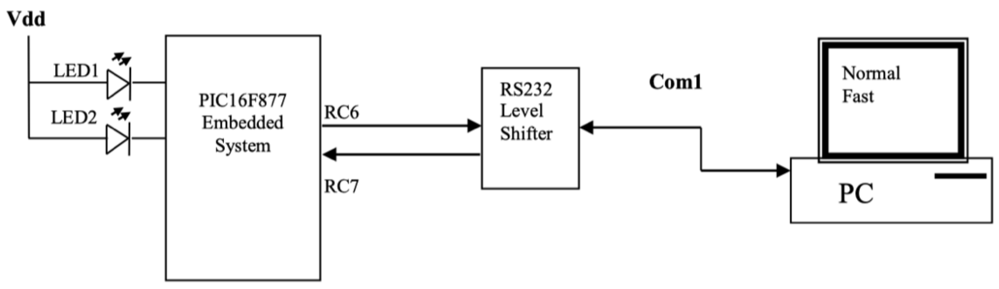


Figure 2 Basic Configuration

Software: MPLAB, WCHSerialPort (in macOS);

Equipment: PIC16F877A, PICkit-3, LED*4, PC and corresponding USB cables.

Part 1

If Fosc = 20Mhz, what is the maximum interrupt period possible with Timer0?

The crystal oscillation frequency in XC8 chip is 4MHZ, and a clock cycle is 4 crystal oscillations. Hence,

$$period = 4 * \frac{1}{F_{osc}} = 4 * \frac{1}{20000000} = 0.2\mu s$$

Part 2

Using the Timer1 interrupt, write a program that flashes LED1 at a rate of ~2hz and LED2 at a rate of ~4hz. The Superloop should contain no code.

We offer two solutions for this part. The complete code of C program using Timer0 is shown in the following Table 1. **And the C program using Timer1 is illustrated in the Table 2.**

Table 1

The C program using Timer0

```
#include <xc.h>
// 'C' source line config statements
// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR
disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit
Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used
for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit
(Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits
(Write protection off; all program memory may be written to by EECON
control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit
(Code protection off)
```

```c
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#ifndef _XTAL_FREQ
// Unless already defined assume 4MHz system frequency
// This definition is required to calibrate the delay functions,
__delay_us() and __delay_ms()
#define _XTAL_FREQ 4000000
#endif


// Definitions____________________________________________________________

#define LED1 RD0       //define RD0 as LED1
#define LED2 RD1       //define RD1 as LED2

// globals _____________________________________________________


// Prototypes_____________________________________________________________

void setup(void);

void loop(void);


void main(void) {

    setup();           // Call initialization function

/* super loop */

    for (;;) {
    }
}


void setup(void) {
    /* setup stuff */

    TRISD = 0x00;          //      Port D bits 0 and 7 as outputs
    PORTD = 0xff;          //      Turn off all LEDs on PORTD
    OPTION_REG = 0x02;       //     T0CS = 0: Timer 0 internal clock
    //   PSA      = 0: Prescaler assined to Timer0
    //   PS2 = 0: Prescaler bits set to 0 1 0 (x8)
    //   PS1 = 1
    //   PS0 = 0

    INTCON = 0xA0;        //       T0IE = 1: enable interrupt on TMR0
overflow
    //   TMR0IF = 0: Timer0 interrupt flag cleared
    //   GIE = 1: Global interrupt enable


    TMR0 = 0;             //       Initialise TMR0 to 0x00
}

void loop(void) {
    LED1 ^= 1;           // Toggle LED1 just to do something
}

/* Interrupt Service Routine */

int cnt = 0;

// 2ms
void __interrupt()       // Interrupt identifier
isr(void)               // Here be interrupt function - the name is
unimportant.
{
    cnt++;

    T0IF = 0;           // Clear interrupt flag, ready for next interrupt

    if (cnt % 125 == 0) {
        LED1 ^= 1;
    }

    if (cnt % 250 == 0) {
        LED2 ^= 1;          // toggle LED2, just to do something
    }


    //TMR0 = 0x08;     // Offset Timer0 to start counting from 0x08 for
    //more accurate timing.

}
```

And the C program using Timer1 is illustrated in the following Table 2.

Table 2

The C program using Timer1

```
#include <xc.h>
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR
disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit
Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used
for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit
(Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits
(Write protection off; all program memory may be written to by EECON
control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit
(Code protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#ifndef _XTAL_FREQ
// Unless already defined assume 4MHz system frequency
// This definition is required to calibrate the delay functions,
__delay_us() and __delay_ms()
#define _XTAL_FREQ 4000000
#endif

#define uint unsigned int
#define uchar unsigned char
#define V0 RD0

#define LED1 RD0       //define RD0 as LED1
#define LED2 RD1       //define RD1 as LED2

void setup() {
    TXSTA = 0x24;
    RCSTA = 0x90;
    SPBRG = 0x19;
    TRISD = 0x00;           //      Port D bits 0 and 7 as outputs
    PORTD = 0xff;

    T1CON = 0x01;
//    0011 0001
//    TMR1CS = 0;
//    T1CKPS0 = 1;
//    T1CKPS1 = 1;

//    TMR1ON = 1;

//    TMR1IF = 0;
    TMR1IE = 1;

    PEIE = 1;
    GIE = 1;

    TMR1H = 0x9E;
    TMR1L = 0x58;
}

void main() {
    setup();

    while (1) {

    }
}

void send(char* str) {
    for (int i = 0; str[i]; i++) {
        if (str[i] < 0 || str[i] >= 256) {
            continue;
        }
        while (!TXIF);
        TXREG = str[i];
    }
    while (!TXIF);
    TXREG = 0x0D;
    __delay_ms(1);
}

int cnt = 0;
```

```c
// 25ms

void __interrupt()
isr(void) {
//    send("xixixi");

    TMR1IF = 0;
    TMR1H = 0x9E;
    TMR1L = 0x58;

    cnt++;

    if (cnt % 10 == 0) {
        LED1 ^= 1;
    }

    if (cnt % 5 == 0) {
        LED2 ^= 1;          // toggle LED2, just to do something
    }
}
```

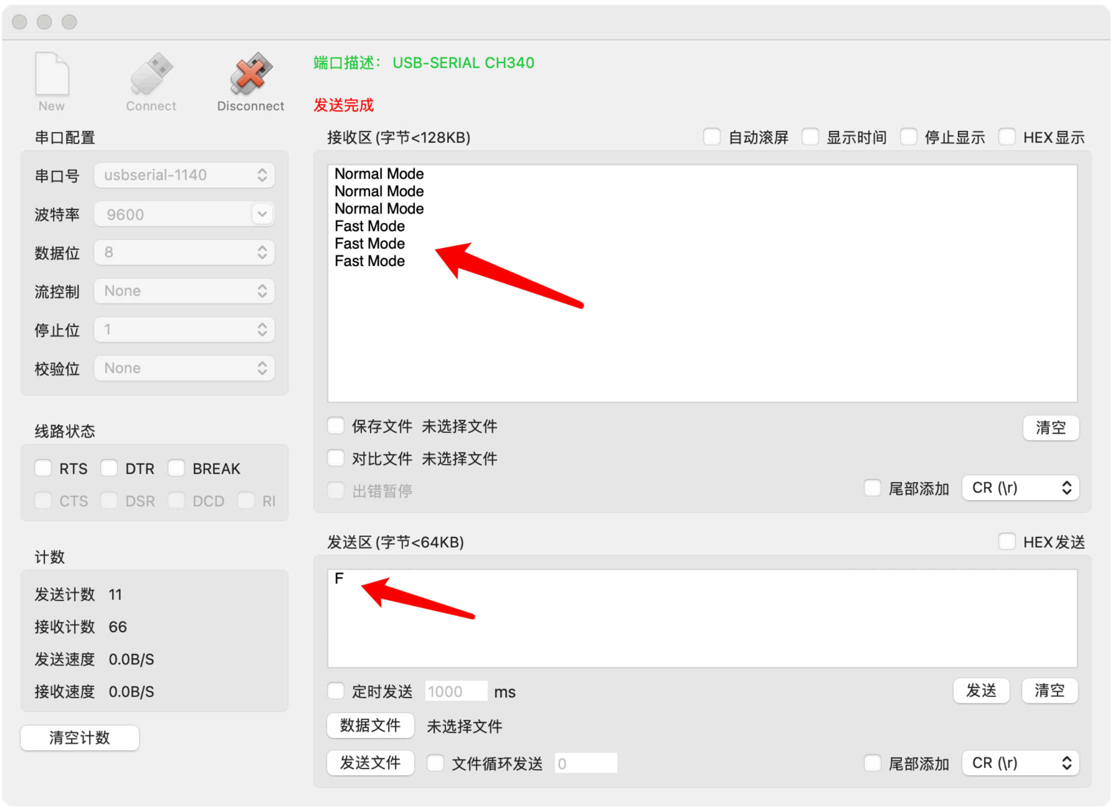
Part 3

Continuing from part 2 above use the USART asynchronous receive interrupt to generate an interrupt when a character is received from Putty.

If the character ‘N’ is received:
LED1 flash rate is ~2hz
LED2 flash rate is ~1hz
Send “Normal Mode” once to Putty

If the character ‘F’ is received:
LED1 flash rate is ~4hz
LED2 flash rate is ~2hz
Send “Fast Mode” once to Putty

NOTE: Sending “Normal Mode” and “Fast Mode” text to Putty should be done in the Superloop and not in the Interrupt Service Routine.



端口描述： USB-SERIAL CH340

发送完成

串口号 usbserial-1140
波特率 9600
数据位 8
流控制 None
停止位 1
校验位 None

```
Normal Mode
Normal Mode
Normal Mode
Fast Mode
Fast Mode
Fast Mode
```

尾部添加 CR (\r)

```
F
```

定时发送 1000 ms

文件循环发送 0

尾部添加 CR (\r)

发送计数 11
接收计数 66
发送速度 0.0B/S
接收速度 0.0B/S

Figure 3 Testing result of the following program

As shown in the above Figure 3, the testing result have been given. And the complete code of C program is shown in the following Table 3.

Table 3

The C program based on MPLAB

```
#include <xc.h>

#pragma config FOSC = XT       // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF      // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF     // Power-up Timer Enable bit (PWRT
disabled)
#pragma config BOREN = OFF     // Brown-out Reset Enable bit (BOR
disabled)
#pragma config LVP = OFF       // Low-Voltage (Single-Supply) In-Circuit
Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used
for programming)
#pragma config CPD = OFF       // Data EEPROM Memory Code Protection bit
(Data EEPROM code protection off)
#pragma config WRT = OFF       // Flash Program Memory Write Enable bits
(Write protection off; all program memory may be written to by EECON
control)
#pragma config CP = OFF        // Flash Program Memory Code Protection bit
(Code protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#ifndef _XTAL_FREQ
// Unless already defined assume 4MHz system frequency
// This definition is required to calibrate the delay functions,
__delay_us() and __delay_ms()
#define _XTAL_FREQ 4000000
#endif

#define uint unsigned int
#define uchar unsigned char
#define V0 RD0

#define LED1 RD0       //define RD0 as LED1
#define LED2 RD1       //define RD1 as LED2

void setup() {
    TXSTA = 0x24;
    RCSTA = 0x90;
    SPBRG = 0x19;

    TRISC = 0xC0;          //      Port D bits 0 and 7 as outputs
```

```c
    PORTC = 0x00;

    TRISD = 0x00;
    PORTD = 0xFF;

    T1CON = 0x01;
//    0011 0001
//    TMR1CS = 0;
//    T1CKPS0 = 1;
//    T1CKPS1 = 1;

//    TMR1ON = 1;

//    TMR1IF = 0;
    TMR1IE = 1;

    PEIE = 1;
    GIE = 1;

    TMR1H = 0x9E;
    TMR1L = 0x58;
}

char recv() {
    while (!RCIF);
    char ret = RCREG;
    RCREG = 0;
    return ret;
}

void send(char* str) {
    for (int i = 0; str[i]; i++) {
        if (str[i] < 0 || str[i] >= 256) {
            continue;
        }
        while (!TXIF);
        TXREG = str[i];
    }
    while (!TXIF);
    TXREG = 0x0D;
    __delay_ms(1);
}

int status = 0;
```

```c
// 1 -> N
// 2 -> F

void main() {
    setup();

    for(;;) {
        char x = recv();
        if (x == 'N') {
            status = 1;
            send("Normal Mode");
        } else if (x == 'F') {
            status = 2;
            send("Fast Mode");
        }
        __delay_ms(100);
    }
}

int cnt = 0;

// 25ms

void __interrupt()
isr(void) {
//    send("xixixi");

    TMR1IF = 0;
    TMR1H = 0x9E;
    TMR1L = 0x58;

    cnt++;
    if (status == 1) {
        if (cnt % 10 == 0) {
            LED1 ^= 1; // 2hz
        }
        if (cnt % 20 == 0) {
            LED2 ^= 1; // 1hz
        }
    } else if (status == 2) {
        if (cnt % 5 == 0) {
            LED1 ^= 1; // 4hz
        }
        if (cnt % 10 == 0) {
            LED2 ^= 1; // 2hz
        }
    }



}
```

Summary for Lab 6

In Lab 6, we have learned the basic knowledge of **Timer** and we explored how to utilize PIC DIP-40 to verify the function of our C programs. And at the end of this semester, we would like to express our sincere thanks to Dr. Wu for her generous guidance. Also, we hope to thank to our tutor for their helpful guidance.

Hanlin CAI 832002117

Qiguo QING 832002127

In 2022/12/08