INFORME DE LABORATORIO 3: INTERRUPCIONES EXTERNAS

Luis Miguel Rincon Pinilla
e-mail: lmrinconp@itc.edu.co
Cristian Camilo Pianda Rodríguez
e-mail: ccpiandar@itc.edu.co

RESUMEN: *En este informe se detalla los procedimientos seguidos y los resultados que se obtuvieron como parte del tercer laboratorio con el entorno de programación MPLAB IDE. En el cual se adquirieron las destrezas necesarias para el uso de interrupciones externas usando el microcontrolador PIC18F4550 por medio de la modificación de sus registros de configuración.*

PALABRAS CLAVE: Análogo, C++, Bandera, Compilador, Interrupción, Microcontrolador, Oscilador, Programación, Registro.

1 INTRODUCCIÓN

El medio de conexión para monitorear o controlar el hardware externo a los microcontroladores son las entradas y salidas analógicas o digitales con las que cuenta. Por lo cual es necesario comprender la manera en que se configuran cada uno de los puertos y registros con los que cuenta el PIC, especialmente los registros de configuración para las interrupciones externas.

Las interrupciones externas permiten atender eventos suspendiendo la ejecución del programa principal logrando dar prioridad a interacciones externas. Por otra parte, la conversión analógica-digital nos permite procesar en el microcontrolador las distintas señales analógicas provenientes del entorno a través de sensores.

2 OBJETIVO

Utilizar las interrupciones externas para responder a eventos procedentes de dispositivos externos y obtener la respuesta deseada en las salidas del microcontrolador.

3 ELEMENTOS UTILIZADOS

Se Utilizaron los siguientes elementos durante el laboratorio:

- Displays de 7 segmentos de cátodo común. (Simulador)
- Fuente de voltaje D.C. (Simulador)
- Integrados 74LS48. (Simulador)
- LEDs. (Simulador)
- Microcontrolador PIC18F4550. (Simulador)
- MPLAB X IDE 5.40.
- Proteus 8.9.
- Resistencias varias. (Simulador)
- Sensor de temperatura LM35. (Simulador)
- Teclado matricial 3x4. (Simulador)

4 DESARROLLO

Para el primer sistema se requería desarrollar un programa en el microcontrolador PIC que emule el funcionamiento de un horno microondas con las siguientes características:

- El tiempo se asignará únicamente en segundos desde un teclado matricial y se dará marcha con la tecla '*'.
- El tiempo se debe visualizar en cuenta regresiva cuando inicie el funcionamiento del horno, por medio de un display 7 segmentos (2 dígitos).
- En cualquier momento del funcionamiento se debe poder pausar la cuenta pulsando la tecla '#'.
- El horno debe mostrar un piloto (led) encendido que indique si está funcionando o está en reposo.
- Si la temperatura sube de 60°C en el horno se debe encender un led de color rojo que alarme al usuario
- Al oprimir un pulsador externo al teclado, el horno debe entrar en modo de mantenimiento para lo cual debe probar los displays con una cuenta de 1 a 9 y los leds indicadores prender y apagar 3 veces. Si se presiona este modo del horno, se debe suspender cualquier acción que esté realizando.

4.1 MONTAJE DEL CIRCUITO

Primeramente, se realizó el montaje del circuito con un microcontrolador PIC18F4550 (*Fig. 1*) con las siguientes características:

- Se conecta la salida del sensor LM35 al pin 2 (**AN0**).
- Se conectó un teclado matricial 3x4 al puerto B (**RB1:RB7**).
- Se conecto un botón junto con una resistencia de Pull-Down al pin 33 (**RB0**).
- Para la conversión BCD a 7 segmentos se usaron dos integrados 74LS48 conectados al puerto D.
- Se conectaron dos displays 7 segmentos uno a cada uno de los 74LS48.
- Se conectaron 2 Leds a los pines 8-9 (**RE0:RE1**).

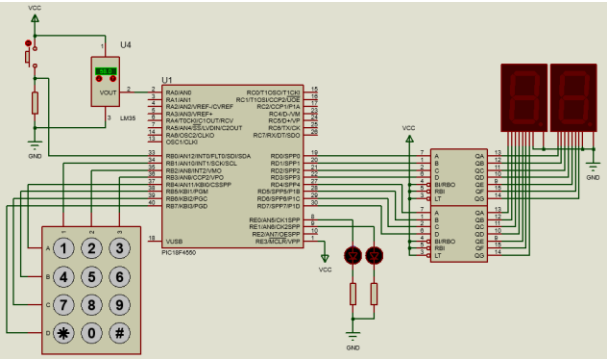

Figura 1. Montaje del Circuito.

4.2 REGISTROS DE CONFIGURACIÓN

4.2.1 CONVERSIÓN ANÁLOGO/DIGITAL

Para habilitar la entrada análoga en AN0 era necesario modificar el registro **ADCON1** que se configuró de la siguiente manera:

- Bits 6 y 7 no es necesaria su modificación pues no están implementados.
- Bit 5 en 0 para usar Vss en vez de RA2 como Vref-.
- Bit 4 en 0 para usar Vdd en vez de RA3 como Vref+.
- Bits 3-0 como 1110 para habilitar AN0 como la única entrada análoga.

Para el registro ADCON0 se usarán las siguientes configuraciones:

- **CHS3:CHS0:** para este grupo de bits usaremos el valor 0 ya que solamente usaremos AN0 como la entrada analógica.
- **GO/DONE:** para este bit usaremos 1 para iniciar la conversión A/D.
- **ADON:** este bit se usará en 1 para habilitar el módulo de conversión A/D.

Para el registro ADCON2 se usarán los siguientes parámetros:

- **ADFM:** para este de bit usaremos el valor 1 para justificar hacia la derecha, usaremos los 10 bits resultantes de la conversión.
- **ADCS:** para este grupo de bits usaremos el valor 1 debido a:

$$T_{OSC} = \frac{1}{8\ MHz} = 125\ ns$$

$$T_{AD} = 0{,}8\ \mu s$$

$$T_{AD} = 0{,}8\ \mu s * \frac{T_{OSC}}{125\ ns} = 6{,}4\ T_{OSC} \approx 8\ T_{OSC}$$

- **ACQT:** este grupo de bits lo configuraremos a 4 debido a:

$$T_{AD} = 8 * 125\ ns = 1\ \mu s$$

$$T_{ACQ} = 2{,}45\ \mu s$$

$$T_{ACQ} = 2{,}45\ \mu s * \frac{T_{AD}}{1\ \mu s} = 2{,}45\ T_{AD} \approx 4\ T_{AD}$$

4.2.2 INTERRUPCIONES

Para habilitar el uso del teclado matricial y del pin RB0 era necesario modificar los registros **INTCON** e **INTCON2** que se configuraron de la siguiente manera:

INTCON

- Bit 7 en 1 para habilitar las interrupciones globales.
- Bit 4 en 1 para habilitar la interrupción INT0 (RB0).
- Bit 3 en 1 para habilitar la interrupción RB.
- Bit 2 en 0 inicia la bandera de INT0 en 0.
- Bit 1 en 0 inicia la bandera de RB en 0.

INTCON2

- Bit 7 en 0 para habilitar las resistencias Pull-Up del puerto.
- Bit 6 en 1 para detectar flancos de subida en INT0.

4.3 PROGRAMACIÓN

El código de programación en C que creamos para el microcontrolador es el siguiente:

```
#include <xc.h>
#include <pic18f4550.h>
#define _XTAL_FREQ 8000000
#pragma config FOSC = INTOSCIO_EC
int dec=0;
int uni=0;
int cont=0;
int flag=0;
int time=0;
int conv=0;
int c=0;
int num=0;

void main(void) {
   OSCCONbits.IRCF=0b111;
   ADCON1=0b00001110;
   ADCON0bits.ADON=1;
   ADCON2bits.ACQT=4;
   ADCON2bits.ADCS=1;
   ADCON2bits.ADFM=1;
   ADCON0bits.CHS=0x00;
   INTCON2bits.NOT_RBPU=0;
   TRISAbits.RA0=1;
   TRISB=0b11110001;
   TRISD=0b00000000;
   TRISEbits.RE0=0;
   TRISEbits.RE1=0;
   PORTAbits.AN0=0;
   PORTB=0b00001110;
   PORTD=0b00000000;
   PORTEbits.RE0=0;
   PORTEbits.RE1=0;
   INTCONbits.GIE=1;
   INTCONbits.INT0IE=1;
   INTCON2bits.INTEDG0=1;
   INTCONbits.INT0IF=0;
   INTCONbits.RBIE=1;
   INTCONbits.RBIF=0;

   while(1){
```

```
        ADCON0bits.GO_DONE=1;
        PORTB=0b11111100;
        PORTB=0b11111010;
        PORTB=0b11110110;
        conv = (ADRESH*256)+ADRESL;
        c = (((conv*100)+0.0)/1023)*5;
        if(c>=60){
            PORTEbits.RE1=1;
        }
        else{
            PORTEbits.RE1=0;
        }
        if(flag==1){
            PORTD=((time/10)*16)+(time%10);
            PORTEbits.RE0=1;
            __delay_ms(1000);
            time--;
            if(time==0){
                flag=0;
            }
        }
        else if(flag==2){
            PORTD=((time/10)*16)+(time%10);
            PORTEbits.RE0=0;
        }
        else if(flag==3){
            time=0;
            PORTE=0b00000000;
            for(int i=0;i<=9;i++){
                PORTD=(i*16)+(i);
                __delay_ms(1000);
            }
            for(int i=0;i<3;i++){
                PORTE=0b00000011;
                __delay_ms(500);
                PORTE=0b00000000;
                __delay_ms(500);
            }
            flag=0;
        }
        else{
            PORTD=((time/10)*16)+(time%10);
            PORTEbits.RE0=0;
        }
    }
    return;
}

void __interrupt() teclado(void){
    if(INTCONbits.RBIF==1){
        if(PORTBbits.RB1==0){
            if(PORTBbits.RB4==0){num=1;cont++;}
            if(PORTBbits.RB5==0){num=4;cont++;}
            if(PORTBbits.RB6==0){num=7;cont++;}
            if(PORTBbits.RB7==0){cont=0;flag=1;}//*
        }
        if(PORTBbits.RB2==0){
            if(PORTBbits.RB4==0){num=2;cont++;}
            if(PORTBbits.RB5==0){num=5;cont++;}
            if(PORTBbits.RB6==0){num=8;cont++;}
            if(PORTBbits.RB7==0){num=0;cont++;}
        }
        if(PORTBbits.RB3==0){
            if(PORTBbits.RB4==0){num=3;cont++;}
            if(PORTBbits.RB5==0){num=6;cont++;}
            if(PORTBbits.RB6==0){num=9;cont++;}
            if(PORTBbits.RB7==0){cont=0;flag=2;time++;}//#
        }

while(PORTBbits.RB4==0||PORTBbits.RB5==0||PORTB
bits.RB6==0||PORTBbits.RB7==0);
        INTCONbits.RBIF=0;
    }
    if(cont==1){
        uni=num;
        time=(dec*10)+(uni);
    }
    if(cont==2){
        dec=uni;
        uni=num;
        time=(dec*10)+(uni);
        cont=0;
        dec=0;
        uni=0;
    }
    if(INTCONbits.INT0IF==1){
        flag=3;
        INTCONbits.INT0IF=0;
    }
}
```

5 Enlace del Video

El video de la simulación se encuentra en: https://youtu.be/Y1dkly54-VE

6 CONCLUSIONES

Lo expuesto en este informe de laboratorio permite arribar a la siguiente conclusión:

- La configuración de los registros asignados a las interrupciones externas (**INTCON:INTCON3**) permitió dar prioridad a la interacción del Microcontrolador con la interfaz humano máquina, suspendiendo temporalmente la ejecución del programa principal.
- Junto con los registros de interrupción externa se pueden utilizar otros registros como los relacionados al conversor análogo-digital (**ADCON0:ADCON2**), permitiendo adquirir y analizar información, obtenida del ambiente a través de un dispositivo (sensor).