Seminario de Sistemas Embebidos
Práctica 11

Jorge Luis Madrid Gómez

29 de noviembre de 2020

UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e Ingenierías

Índice

Objetivo General

Marco Teórico

PWM

La modulación por ancho de pulso o PWM (*Pulse Width Modulation* por su siglas en inglés) se usa para controlar el ancho de una señal digital con el propósito de controlar a su vez la potencia que se entrega a ciertos dispositivos. Modificando el ancho del pulso activo (que está en On) se controla la cantidad de corriente que fluye hacia el dispositivo.
Un PWM funciona como un interruptor, que constantemente se activa y desactiva, regulando la cantidad de corriente y por ende de potencia, que se entrega al dispositivo que se desea controlar [1].

Ciclo de trabajo

En un sistema PWM, el dispositivo alimentado recibe corriente por un tiempo y deja de recibirlo por otro, repitiéndose este proceso continuamente. Si se aumenta el tiempo en que el pulso está en nivel alto, se entrega más potencia y si se reduce el tiempo entrega menos potencia. Para describir la cantidad de tiempo en la que se entrega el flanco de subida de la señal de pulso (en nivel alto), se utiliza el concepto de **ciclo de trabajo** (*duty cycle*). En la figura 1 se observa que el ciclo de trabajo se define en porcentaje del periodo de la señal cuando es el valor de flanco de subida [2].

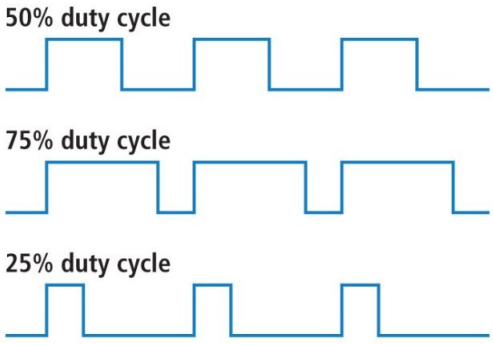

Figura 1: 50 %, 70 % y 25 % de ciclo de trabajo

Frecuencia de PWM

Al ser una señal lógica con flancos de subida y bajada, se tiene que definir la frecuencia del sistema PWM, que generalmente es el doble que la frecuencia máxima con la cual trabaja el dispositivo alimentado. Tanto la frecuencia PWM y el ciclo de trabajo los determinamos nosotros.
Para manejar PWM en el **PIC18F4550** es necesaria la configuración de distintos registros. Se comienza con la determinación del valor para los registros `PR2`, `CCPRxL` y `CCPxCON<5:4>`. Para ello se utilizan las siguientes ecuaciones.

$$\text{PR2} = \frac{\text{Xtal}_f}{4(\text{F}_{PWM})(\text{TMR2 Prescal Value})} - 1 \tag{1}$$

$$\text{CCPRxL} : \text{CCPxCON} < 5 : 4 > = (CT\,\%)\frac{(\text{PR2}+1)4}{100} \tag{2}$$

Donde Xtal_f es la frecuencia del reloj del microcontrolador, F_{PWM} es la frecuencia PWM que determinamos, TMR2 Prescal Value es el preescalador del Timer 2, que generalmente se le asigna el valor de 1 y $CT\,\%$ es el porcentaje del ciclo de trabajo que determinamos. El resultado obtenido en (2) se expresa en un número binario de 10 bits, los 8 bits más significativos se guardan en el registro `CCPRxL` y los últimos 2 bits menos significativos se guardan en los bits 5 y 4 del registro `CCPxCON`.

Comunicación Serial

La comunicación en serie es un método comúnmente utilizado para intercambiar datos entre ordenadores y dispositivos periféricos. La transmisión serie entre el emisor y el receptor está sujeta a protocolos estrictos que proporcionan seguridad y fiabilidad y han llevado a su longevidad. Muchos dispositivos, desde ordenadores personales hasta dispositivos móviles, utilizan la comunicación en serie.
En la transmisión de datos en serie se utilizan pulsos binarios para transmitir los datos. El dígito binario uno está representado por cinco voltios o una lógica ALTA. Por el contrario, el cero binario se denota con una lógica BAJA o cero voltios. Para implementar la comunicación en serie, se requieren un origen y un destino. También se les conoce como emisor y receptor. Se pueden emplear varios tipos de comunicación serie y se designan como Simplex, Half Duplex y Full Duplex (figura 2).

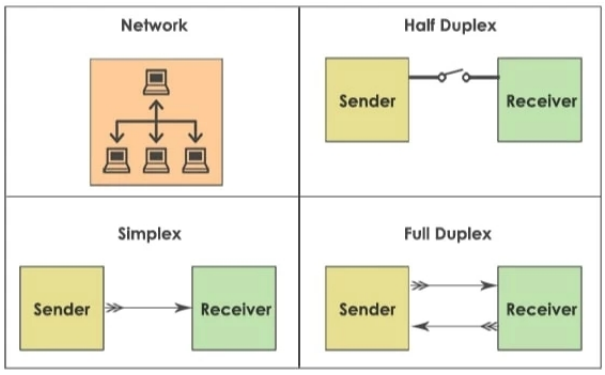

Figura 2: Tipos de comunicación serial

Modulo EUSART

El módulo Transmisor/Receptor Universal Síncrono/Asíncrono mejorado (*Enhanced Universal Synchronous Asynchronous Receiver Transmitter* - EUSART) es un periférico de comunicación serie de entrada/salida. Asimismo es conocido como Interfaz de comunicación serie (*Serial Communications Interface* - SCI). Contiene todos los generadores de señales de reloj, registros de desplazamiento y búfers de datos necesarios para realizar transmisión de datos serie de entrada/salida, independientemente de la ejecución de programa del dispositivo. Como indica su nombre, aparte de utilizar el reloj para la sincronización, este módulo puede establecer la conexión asíncrona, lo que lo hace único para algunas aplicaciones [3]. Por ejemplo, en caso de que sea difícil o imposible proporcionar canales especiales para transmisión y recepción de datos y señales de reloj (por ejemplo, mando a distancia de radio o infrarrojas), el módulo EUSART es definitivamente la mejor opción posible. El EUSART integrado en el **PIC18F4550** posee las siguientes características:

- Transmisión y recepción asíncrona en modo Full-duplex.
- Caracteres de anchura de 8 – 9 bits programables.
- Detección de dirección en modo de 9 bits.
- Detección de errores por saturación del búfer de entrada.
- Comunicación Half Duplex en modo síncrono.

Para habilitar la transmisión de datos por medio del módulo EUSART, es necesario configurarlo para que funcione como un transmisor. En otras palabras, es necesario definir el estado de los siguientes bits:

- `CREN`. Habilita la recepción asíncrona.
- `BRGH`. Determina el nivel de velocidad de baudios, para computadoras es alta (*high*).
- `SPBRG`. Velocidad de baudios, en computadoras es de 9600.
- `SYNC`. El tipo de comunicación (síncrona o asíncrona).
- `SPEN`. Activación del puerto serial.
- `TX9/RX9` La cantidad de bits por paquete de datos (8 o 9 bits).
- `TXEN`. Habilita la transmisión.
- `TXREG`. Registro de transmisión.

Materiales y Métodos

Para el desarrollo de la práctica se necesitan los siguientes recursos.

- MPLAB con compilador XC8
- Proteus versión 8.++

Ya que la práctica solo requiere simulación y programación, no se requieren materiales físicos, por lo que en párrafos posteriores se mostrarán los elementos de la simulación necesarios. El microcontrolador usado en el curso será el **PIC18F4550**.
Se requiere que el usuario determine la frecuencia y ciclo de trabajo de PWM entre 3 opciones dadas.

- Frecuencia = 2KHz y Ciclo de trabajo = 50 %.
- Frecuencia = 10KHz y Ciclo de trabajo = 80 %.
- Frecuencia = 1KHz y Ciclo de trabajo = 30 %.

El microcontrolador otorgará el voltaje promedio de acuerdo a la configuración seleccionada. Esta selección se hará a partir de una comunicación serial entre el microcontrolador y la computadora por el modulo EUSART.

Simulación Proteus

Dentro del entorno de simulación de Proteus se realiza la construcción del circuito para la práctica, donde necesitamos los siguientes elementos electrónicos disponibles en las librerías que nos ofrece.

- El microcontrolador (PIC18F4550)
- Botón (BUTTON)
- Resistencia de 1 kΩ (RES)
- Osciloscopio (OSCILLOSCOPE)
- Terminal VIrtual (VIRTUAL TERMINAL)

La construcción del circuito se muestra en la figura 3.

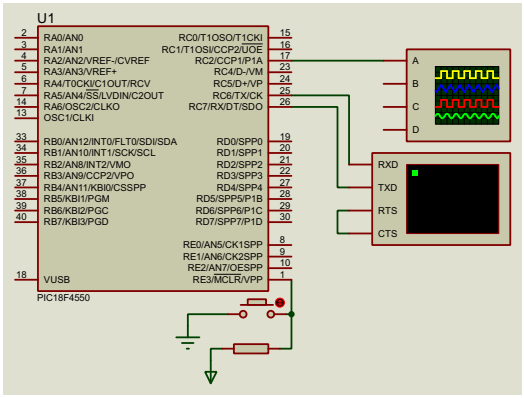

Figura 3: Simulación en Proteus

El registro `CCPxL` para PWM escogido es el `CCP1` por lo tanto, se conecta el osciloscopio al bit 2 del puerto C correspondiente para poder medir tanto el ciclo de trabajo como la frecuencia que el ussuario escoja. En los bits 6 y 7 del puerto C se encuentran los bits de comunicación serial `TX` y `RX` y se conectan a la terminal virtual; el `TX` y `RX` del micro al `RX` y `TX` de la terminal respectivamente. Por último se coloca el botón de `reset` en el microcontrolador.

MPLAB

Dentro del entorno de programación de MPLAB se crea un proyecto configurado para nuestro microcontrolador y agregamos un `main.c` para programar el funcionamiento de la práctica.
En el `main.c` se agregan los fusibles habituales, el reloj interno se ajusta a una frecuencia de 1MHz y se agregan la librería `<stdio.h>` para el manejo de cadenas.
Se declaran las variables `p` para guardar el valor determinado por la ecuación (1) y `c`, `c5` y `c4` para guardar de forma binaria el resultado de la ecuación (2); los 8 bits más significativos en `c` y los últimos 2 en `c5` y `c4` respectivamente.
Se crean las siguientes funciones para la comunicación serial.

```
//funciones
void conf_serial(void);
void inicio(void);
void Write(unsigned char data);
void Write_Text(unsigned char *text);
unsigned char Read();
```

En la función `conf_serial` se configuran los bits descritos en la página 6 de este documento y de acuerdo al manual de usuario del **PIC18F4550**.

```
void conf_serial(void){
    CREN=1; //Comunicacion asincrona
    BRGH=1; //Alta velocidad (1)
    BRG16=1;    //Velocidad 16 bits
    SPBRGH=0;
    SPBRG=25;    //9600 baudios
    SYNC=0; //Comunicacion asincrona
    SPEN=1; //Puerto serial habilitado (RX TX)
    TX9=0;  //Paquete de 8 bits (0)
    RX9=0;  //Paquete de 8 bits (0)
    TXEN=1; //Habilita transmisison
    TXREG=1; //Registro de transmision
    TRISC6=1;    //TRISCbits.TRISC6=1;
    TRISC7=1;    //TRISCbits.TRISC7=1;
}
```

La funciones `Write` y `Read` se utilizan para mandar el mensaje a la terminal y recibirlo respectivamente, mediante los registros `TRMT` y `RCIF`.

```
void Write(unsigned char data){
    while(!TRMT);    //ejecuta hasta terminar de escribir
    TXREG=data; //ingresa el caracter
}
unsigned char Read(){
    while(!RCIF); //ejecuta la funcion hasta terminar de leer
    return RCREG;
}
```

La función `Write_Text` unicamente concatena en una cadena caracteres con la función `Write`.
En la función `main` se inicia llamando la función `inicio` que a su vez llama a la función `conf_serial`, ya que es recomendable que se ejecute como una subrutina de una función y no directamente en el programa.
Se declaran las cadenas para los distintos mensajes en la terminal y la variable `dato1` que guardará el caracter que escriba el usuario en la terminal.

```
unsigned char *M1[]={"Seleccion de PWM\r\n"};
unsigned char *M2[]={"1. Freq = 2KHz, CT = 50%\r\n"};
unsigned char *M3[]={"2. Freq = 10KHz, CT = 80%\r\n"};
unsigned char *M4[]={"3. Freq = 1KHz, CT = 30%\r\n"};
unsigned char *M5[]={"Frequency = 2KHz\n CT = 50%\r\n"};
unsigned char *M6[]={"Frequency = 10KHz\n CT = 80%\r\n"};
unsigned char *M7[]={"Frequency = 1KHz\n CT = 30%\r\n"};
unsigned char *ME[]={"ERROR\r\n"};
unsigned char dato1=0;
```

Se configura la frecuencia del reloj interno a 1MHz con el registro `OSCCON` y se utiliza la función `Write_Text` para mostrar los primeros 4 mensajes.

```
//Configuracion de la frecuencia del reloj (1MHz)
OSCCON=0b01000010;

Write_Text(*M1);
Write_Text(*M2);
Write_Text(*M3);
Write_Text(*M4);
```

Después se utiliza un bucle `while` para repetir el programa principal de forma indefinida. Dentro de este bucle se comienza por vaciar la variable `dato` y después asignarle el valor del caracter que seleccione el usuario con la función

`Read`. Hecho esto, se utiliza una estructura `switch` para determinar la opción correspondiente a la selección del usuario.

```
switch(dato1){
     case '1'://Freq=2k CT=50%
          Write_Text(*M5);
          p=124;
          c=0b00111110;c5=1;c4=0;
          break;
     case '2'://Freq=10K CT=80%
          Write_Text(*M6);
          p=24;
          c=0b00010100;c5=0;c4=0;
          break;
     case '3'://Freq=1K CT=30%
          Write_Text(*M7);
          p=249;
          c=0b01001011;c5=0;c4=0;
          break;
     default:
          Write_Text(*ME);
          break;
}
```

Al final se asignan los valores de `p`, `c`, `c5` y `c4` a sus respectivos registros para configurar el PWM.

```
PR2=p;
//CCPRxL (8 bits)
     CCPR1L=c;
//CCPxCON (bits 5 y 4)
CCP1CONbits.DC1B1=c5;
CCP1CONbits.DC1B0=c4;

TRISCbits.RC2=0; //Bit para el CCP1 de salida (PWM)
T2CON=0b00000100; //Timer 2 (TMR2) solamente el bit 2 se activa
//Configuracon de CCP1 para PWM
CCP1CONbits.CCP1M3=1;
CCP1CONbits.CCP1M2=1;
```

Resultados

A continuación se muestran los resultados obtenidos.

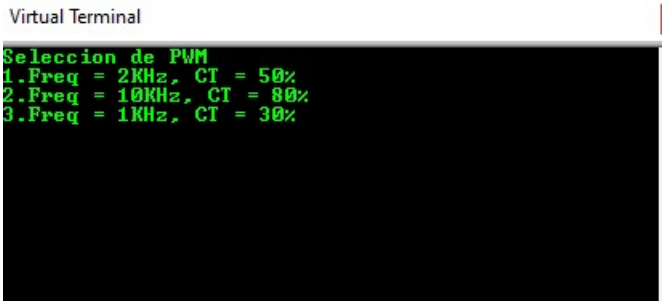

Figura 4: Virtual Terminal

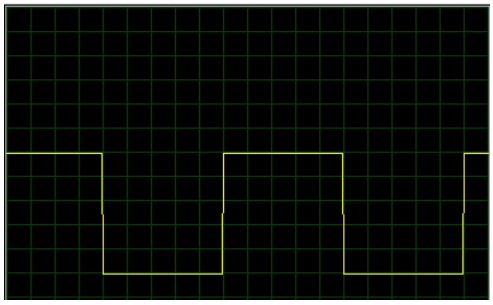

Figura 5: Frecuencia PWM = 2KHz, Ciclo = 50 %

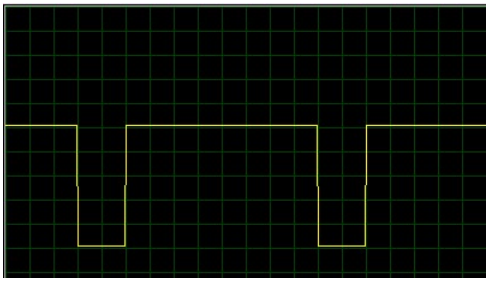

Figura 6: Frecuencia PWM = 10KHz, Ciclo = 80 %

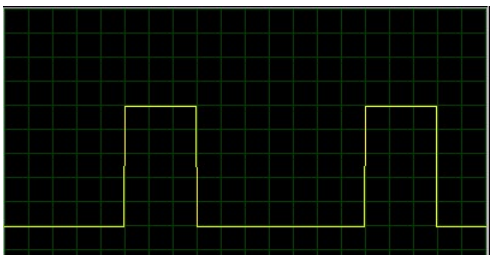

Figura 7: Frecuencia PWM = 1KHz, Ciclo = 30 %

Conclusión

Como se observa en la figura 4 la ejecución de la comunicación serial con el modulo EUSART entre la computadora y el microcontrolador funcionan de manera efectiva, mostrando los mensajes correspondientes en cada selección que realiza el usuario.
Ahora, respecto a la configuraciones de PWM, en las figuras 5, 6 y 7 se visualiza de forma práctica que los ciclos de trabajo corresponden efectivamente a como se configuraron en la selección del usuario, en cambio las frecuencias no son muy bien apreciables, para ello se explican en el vídeo del enlace adjunto.
La práctica se realizó de forma optima y los resultados son satisfactorios dentro de los parámetros que se requirieron y los requisitos de especificados.

Link del vídeo: https://cutt.ly/XhjTLd9

Referencias

[1] J. Posada Contreras, "Modulación por ancho de pulso (pwm) y modulación vectorial (svm). una introducción a las técnicas de modulación," 2005.

[2] J. C. H. Lozada and P. P. Robles, Juan Carlos González y Romero, "Modulación por ancho de pulsos en lógica programable," *Polibits*, no. 33, pp. 25–29, 2006.

[3] J. C. H. Lozada, I. T. Flores, and M. M. Castillo, "Unidad básica de comunicación serial en un microcontrolador," *Polibits*, no. 33, pp. 3–7, 2006.