Seminario de Sistemas Embebidos
Práctica 4

Jorge Luis Madrid Gómez

17 de octubre de 2020

UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e Ingenierías

Índice

Objetivo General

El Objetivo de esta práctica es realizar un temporizador que simule la función de un microondas convencional, utilizado multiplexación en el muestreo del tiempo en 4 displays de 7 segmentos por medio de un microcontrolador.

Marco Teórico

Multiplexores

El multiplexor, también conocido de manera acortada como *MUX*, es un circuito lógico combinacional diseñado para conmutar una o varias líneas de entrada en una sola línea de salida común mediante la aplicación de una lógica de control [1]. Si utiliza un multiplexor de 4 canales de entrada. Una de los cuatro canales de entrada será escogido para pasar a la salida y ésto se logra con ayuda de las señales de control o selección.
La cantidad de líneas de control que debe de tener el multiplexor depende del número de canales de entrada, todo mediante la fórmula: $in = 2^n$, donde in es el número de entradas y n el número de líneas de selección.

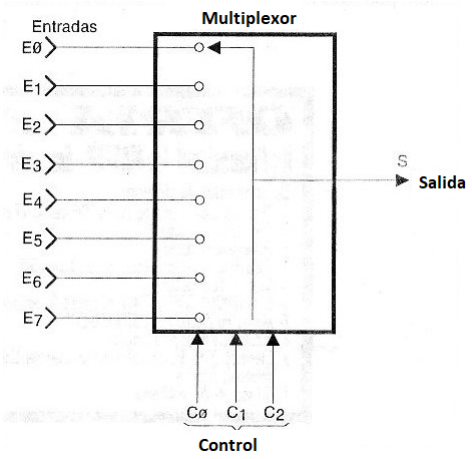

Figura 1: Multiplexor de 8 entradas 1 salida

En la figura 1 se presenta un multiplexor de 8 entradas por lo que aplicando la fórmula $8 = 2^n$, $n = 3$. Se obtienen 3 líneas de control. Cabe aclarar que si se presenta un número de entradas que no es potencia de 2, se escoge

la n que satisfaga $in \leq 2^n$, de preferencia el valor más próximo para no usar líneas de control de más.

Transistor BJT

El transistor de unión bipolar o BJT (*Bipolar Junction Transistor*) es el más común de los transistores, y como los diodos, puede ser de germanio o silicio. En ambos casos el dispositivo tiene 3 patillas y son: el emisor (E), la base (B) y el colector (C).

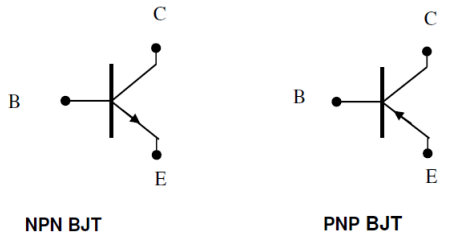

Figura 2: Transistores NPN y PNP

El nombre transistor se obtiene de la combinación de las palabras '*transfer*' y '*resistor*'. El motivo del nombre se debe a que es un resistor que puede amplificar una señal eléctrica mientras es transferida de su terminal de entrada al terminal de salida [2].
Existen dos tipos transistores: el NPN y el PNP, y la dirección del flujo de la corriente en cada caso, lo indica la flecha que se observa en su ilustración correspondiente en la figura 2.

La ganancia Beta

El transistor bipolar es un amplificador de corriente, esto quiere decir que si le introducimos una cantidad de corriente por una de sus patillas (base), el entregará por otra (emisor), una cantidad mayor a esta, en un factor que se llama amplificación. Este factor de amplificación se llama β (beta) o ganancia y es un dato propio de cada transistor. La beta esta definida mediante

$$\beta = \frac{i_C}{i_B}$$

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**.

Para la práctica se requiere desarrollar un circuito que simule la función de temporizador de un microondas. Para ello se deben utilizar 4 displays de 7 segmentos que mostraran los minutos y segundos del tiempo definido. Para asignar el tiempo hay 3 maneras de hacerlo.

- Botón de palomitas. 01:30 (*popcorns*).
- Botón de descongelar. 5:30 (*defrost*).
- Selección del tiempo manual.

Los botones de 'palomitas' y 'descongelar' tienen un tiempo ya definido.
Una vez determinado el tiempo se procede con el inicio del temporizador. Para poder mostrar el tiempo restante en todos los displays es necesario desarrollar un multiplexor con transistores. Para ello, el transistor tiene la aplicación de un *switch*, por lo tanto el circuito quedaría como en la figura 3.

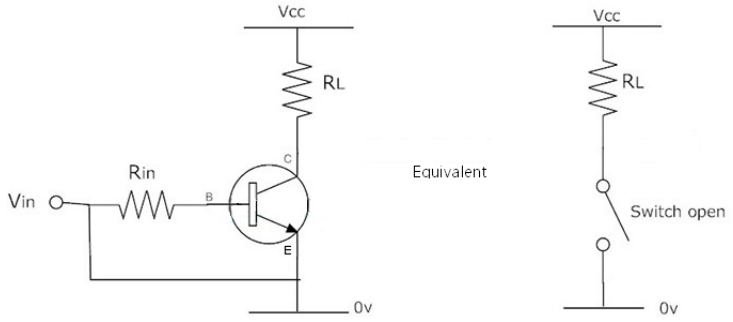

Figura 3: Circuito del transistor

El valor de la resistencia R_in es indispensable para el buen funcionamiento del *switch*, ya que la corriente i_B determina el estado del *switch*. Esta se obtiene mediante

$$R_{in} = \frac{V_{in} - 0.7\text{V}}{i_B}$$
$$i_B = 10\frac{i_{Cmax}}{\beta}$$

Para simulaciones tomamos $\beta = 100$, por lo tanto

$$i_B = \frac{i_{Cmax}}{10} = \frac{0.087\text{Amp}}{10} = 0.0087\text{Amp}$$
$$R_{in} = \frac{5\text{V} - 0.7\text{V}}{0.0087\text{Amp}} = 484\Omega$$

Se reitera que este valor es únicamente servible para simulaciones.
Un circuito de estos se conecta a la terminal `GND` de cada uno de los displays, así al mandar la señal de control se cierra el *switch* y se muestra el valor correspondiente en el display. Más tarde en la programación se determina la velocidad de multiplexeo.

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)
- 7 botones (BUTTON)
- 14 resistencias de distintos valores (RES)
- 4 displays de 7 segmentos (7SEG-COM-CAT-GRN)
- LED azul (LED-BLUE)
- LED verde (LED-GREEN)
- Reloj Externo de Cuarzo a 8MHz (CRYSTAL)

- 2 capacitores de 22pF (CAP)

La construcción del circuito se muestra en la figura 4.

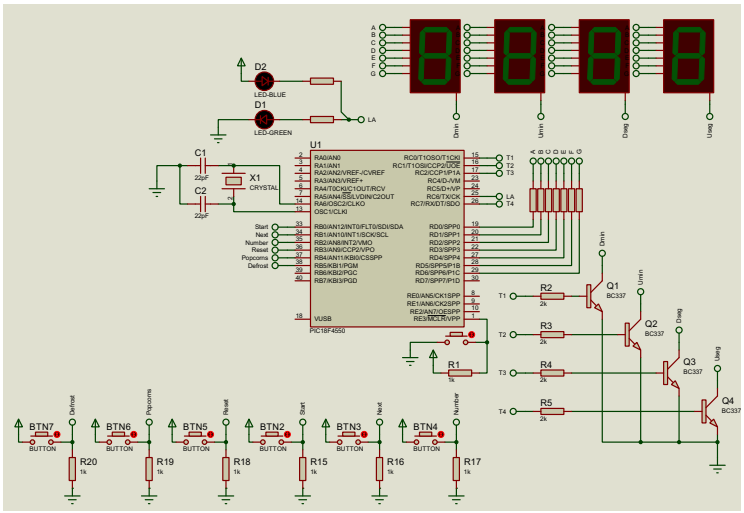

Figura 4: Simulación en Proteus

Los 4 displays mostraran el tiempo; de derecha a izquierda unidad de segundo, decena de segundo, unidad de minuto y decena de minuto.
El LED azul encenderá cuando no esté funcionando el temporizador, y cuando éste se active brillará el LED verde.
La función de cada uno de los botones la describe la terminal conectada a cada uno de ellos. De derecha a izquierda: *Number, Next, Start/Pause, Reset, Popcorns y Defrost.*
Los puertos D y C se configuran de salida mientras que el puerto B será de entrada para los botones.
Por último tenemos los transistores conectados a la terminal de salida (`GND`) de cada uno de los displays correspondientes y un reloj externo a 8MHz.

MPLAB

Ahora se pasa al entorno de programación de MPLAB, donde se desarrolla el programa para la selección de tiempo y el control del temporizador.
Se agregan los fusibles habituales y las librerías necesarias para el código. En esta práctica solo se necesitan las librerías `<xc.h>` y `<math.h>` para la operaciones matemáticas necesarias.

Variables y Funciones

Se utilizan las siguientes variables.

```
//variables
unsigned int TABLA[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
//variables para los segundos y minutos
unsigned int Useg=0,Dseg=0,Umin=0,Dmin=0;
//variables para el funcionamiento
unsigned int cont=0,pos=0,reset=0,paro=0;
```

Como se observa, el vector `TABLA` construido en la práctica pasada nos ayuda a mostrar los números deseados en los displays. las variables `Useg`, `Dseg`, `Umin` y `Dmin` son para guardar los valores numéricos de la unidad de segundo, decena de segundo, unidad de minuto y decena de minuto respectivamente. Por último las variables `cont` y `pos` nos ayudan a acumular el valor numérico de las variables de tiempo y obtener la posición para determinar en que variable de tiempo guardar el dicho valor.
`reset` y `paro` son banderas para los botones de *Reset* y *Start*.
Se utilizan las siguientes funciones.

```
//funciones
void programacion();
void temporizador();
int resta(int n);
void multiplexar(unsigned int veces);
```

La función `programación` es para poder modificar el valor del tiempo manualmente. La función `temporizador` es la encargada de realizar el conteo hacia abajo del tiempo y determinar las condiciones para pasar de minutos a segundos. La función `resta` es para realizar la resta de los valores del tiempo. La función `multiplexar` se utilizar para hacer el encendido y apagado de los displays a cierta frecuencia.

Función Main

Se configuran los puertos.

```
ADCON1 = 0x0F;   //Desabilita todas las entradas analogicas
TRISD=0;         //Puerto D de salida
TRISC=0;         //Puerto C de salida
TRISB=1;         //Puerto B de entrada
```

Dentro de un `while` infinito se repite el programa. Para determinar que tiempo se tiene que agregar al temporizador se usa la siguiente línea de código.

```
while(1){
            //Si se presiona el boton de palomitas
            if(PORTBbits.RB4==1){
                //tiempo de 1:30
                Useg=0;Dseg=3;Umin=1;Dmin=0;
                break;
            }
            //Si se presiona el boton de descongelar
            if(PORTBbits.RB5==1){
                //tiempo de 5:30
                Useg=0;Dseg=3;Umin=5;Dmin=0;
                break;
            }
            //Si se presiona el boton de inicio
            if(PORTBbits.RB0==1){
                //Se programa manualmente el tiempo
                programacion();
                break;
            }
        }
```

Ahora mientras no se presione el botón de *Reset* (`reset=false`) se ejecuta la función `multiplexar` y mientras no se presione el botón de *Start/Pause* (`paro=false`) se ejecuta la función `temporizador`. En código sería lo siguiente.

```
while(reset==0){
            //si se presiona el boton de reset
            if(PORTBbits.RB3==1){
                reset=1;      //reset=true
                LATCbits.LATC6=0;    //se enciende el LED azul
                __delay_ms(200);
            }
            //si se presiona el boton de inicio/pausa
            if(PORTBbits.RB0==1){
                paro++;       //paro se hace par o impar
                              //dependiendo del estado anterior
                              //del temporizador
            }
            //Si paro es par
            //Continua el temporizador
            if(paro%2==0){
```

```
                LATCbits.LATC6=1;
                temporizador();
            }
            //si es impar para el temporizador
            else   LATCbits.LATC6=0;    //se enciende el LED azul
            //mientras reset=false
             multiplexar(10);
        }
    }
```

Función multiplexar

Para esta función se manda al display el valor de tiempo correspondiente, se agrega un *delay* de 15ms y se apaga. Esto se hace consecutivamente empezando con la unidad de segundo y terminando con la decena de minuto. A continuación se muestra la línea de código para el primer diplay, el cuál es similar para los otros 3.

```
//Unidad de segundo
LATD=TABLA[Useg];
LATCbits.LATC7=1;
__delay_ms(15);
LATCbits.LATC7=0;
```

Todo esto se repite dentro de un `while` la veces que indica el parámetro `veces`.

Resultados

A continuación se muestran los resultados obtenidos.

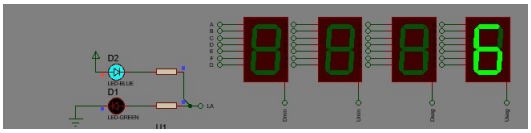

Figura 5: Entrada a la unidad de segundo

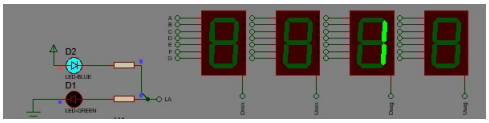

Figura 6: Entrada a la decena de segundo

Como se puede observar en la figura 5, se ingresa 6 en el espacio de unidad de segundo, y el LED azul indica que aún no inicia la cuenta regresiva del temporizador. En la figura 6 Se introduce un 1 en la decena de segundo. Los resultados del temporizador se muestran en el vídeo adjunto, ya que por el multiplexeo no se aprecia de manera adecuada en figuras.

Link de vídeo: https://cutt.ly/Rgh0iC8

Conclusión

Los multiplexores tienes distintas funciones lógicas en distintas áreas de la electrónica, estos permiten la categorización de entradas por medio de señales de control. Uno de estos usos es la de mostrar un valor de multiples dígitos en múltiples displays conectados a un solo puerto, esto para optimizar la cantidad de datos utilizados. En la práctica esto se observa que funciona de manera adecuada, ya que permite una optimización en el uso de los puertos. Pero, en simulación, esto trae desventajas, ya que la función correcta de la multiplexación esta ligada al procesador de la computadora donde ocurre la simulación, aunque en prácticas físicas, si se implementa de manera correcta, es imperceptible los cambios en los displays.

Referencias

[1] A. Papoulis, *Sistemas y circuitos: digitales y analógicos*. Marcombo, 1989.

[2] G. V. Madrid and M. A. Z. Izquierdo, "Transistores de union bipolar (bjt),"