UNIVERSIDAD DE SONSONATE

Arquitectura de Computadoras

Guía 2 –Display de 7 Segmentos

Instructor: Ricardo González

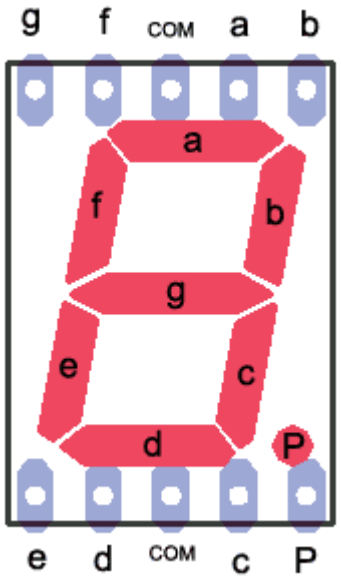
Facultad de Ingeniería y Ciencias Naturales

Contenido

¿Qué es un Display de 7 Segmentos?

El display de 7 segmentos, es un componente que se utiliza para visualizar la representación de números en muchos dispositivos electrónicos. Aunque su uso se está haciendo menos frecuente, en cambio se observan más LCD para la visualización de elementos (Debido a su bajísimo consumo de energía), aunque existen algunos dispositivos que utilizan el display de 7 segmentos para visualizar elementos de manera sencilla.

Este elemento se ensambla de manera que se pueda activar cada segmento (diodo LED) por separado logrando de esta manera combinar los elementos y representar todo los números compuesto entre el 0 y el 9. Un ejemplo de un display de 7 segmentos es el que se muestra a continuación:



El display más común es el display de 7 segmentos de color Rojo, por su facilidad de visualización.

Como podeos observar cada uno de los elementos del display tiene asignado una letra que identifica su posición en el arreglo del display, de esta manera si deseamos visualizar por ejemplo los siguientes números:

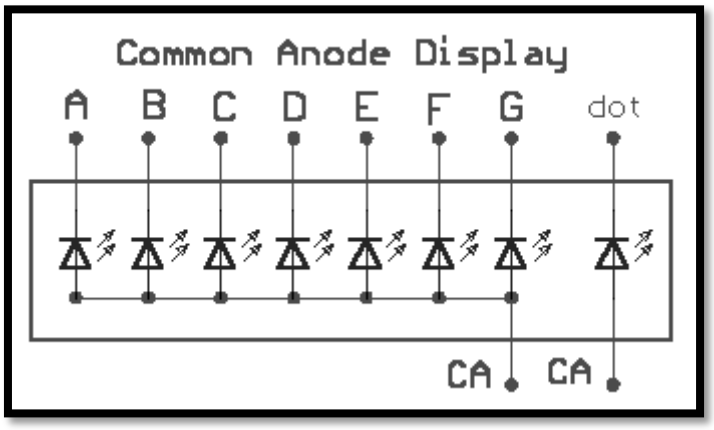
- Si deseamos visualizar el numero ***8*** debemos de activar todos los segmentos.
- Si deseamos visualizar el numero ***0*** debemos de activar los segmentos ***a, b, c, d, e, f.***
- Si deseamos visualizar el numero ***4*** debemos de activar los segmentos ***b, c, f, g.***

Existen 2 tipos de display:

- ***El display de ánodo común.***
- ***El display de cátodo común.***

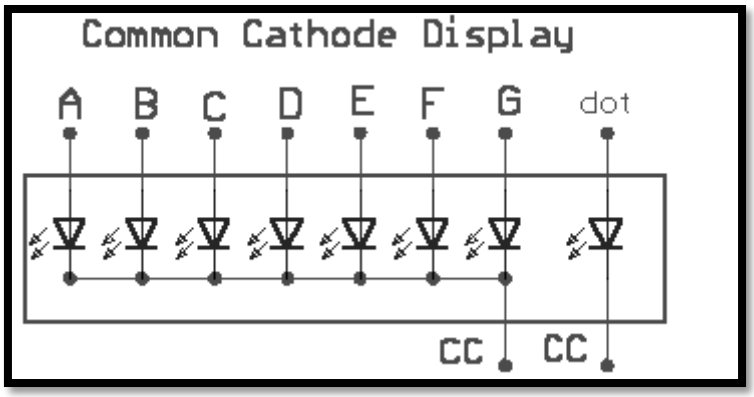
El Display de Ánodo Común.

Todos los ánodos de los LEDS o segmentos están unidos internamente a una patilla común que debe ser conectada a potencial positivo (nivel "1"). El encendido de cada segmento individual se realiza aplicando potencial negativo (nivel "0") por la patilla correspondiente a través de una resistencia que límite el paso de la corriente. La representación gráfica seria la siguiente:



El Display de Cátodo Común.

Todos los cátodos de los LEDS o segmentos están unidos internamente a una patilla común que debe ser conectada a potencial negativo (nivel "0"). El encendido de cada segmento individual se realiza aplicando potencial positivo (nivel "1") por la patilla correspondiente a través de una resistencia que límite el paso de la corriente. La representación gráfica sería la siguiente:

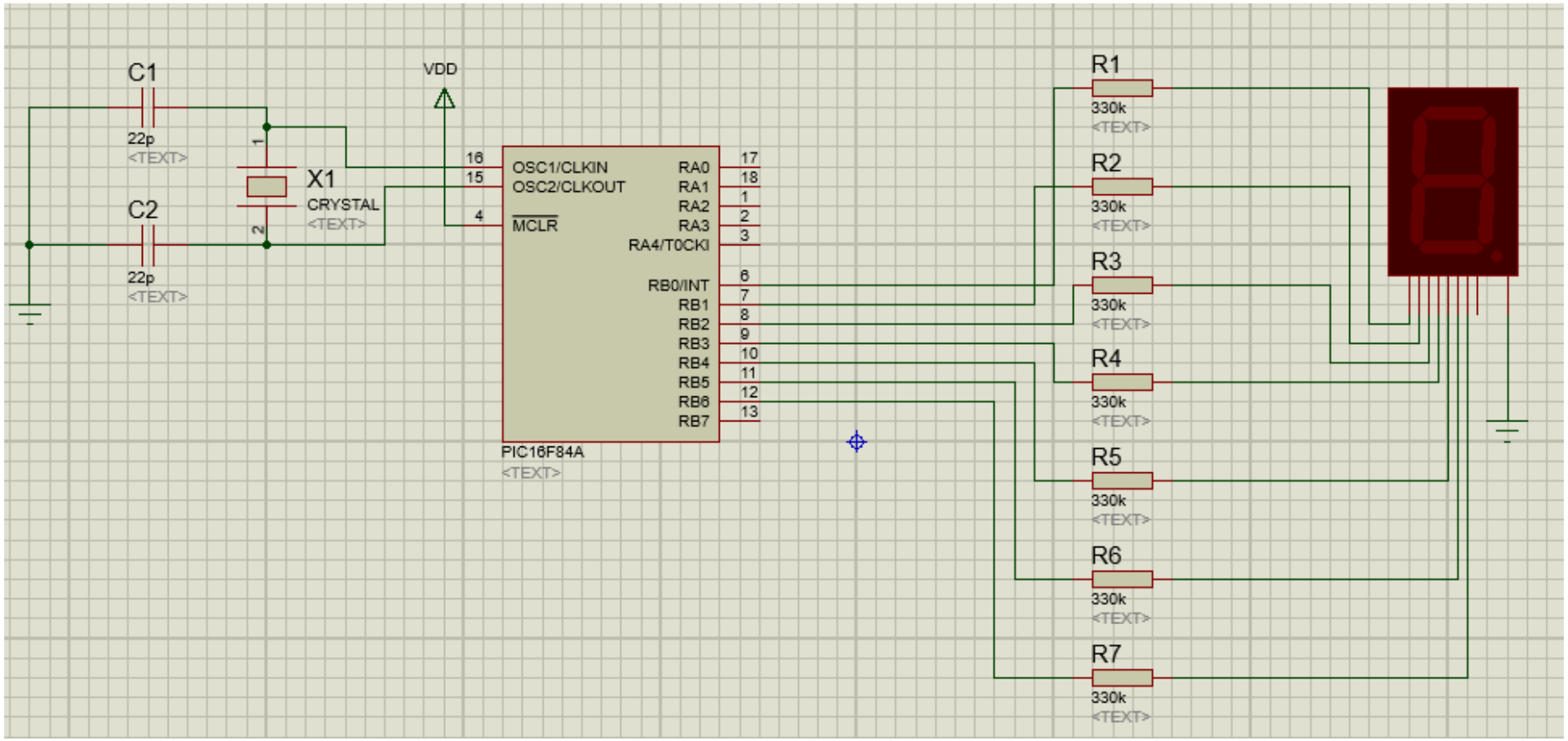


Diseño Básico implementando un display de 7 segmentos (Cátodo Común)

Para crear nuestro circuito base, con una configuración de display de 7 segmentos de Cátodo Común, utilizaremos los siguientes elementos:

- Un display de 7 Segmentos (Cátodo Común): ***7SEG-MPX1-CC.***
- Resistencias para cada uno de los pines de entrada del display: ***RESISTOR***
- Cristal de Cuarzo configurado a 4Mhz: ***CRYSTAL.***
- Condensadores: ***AVX0402NP022P.***

Circuito Básico para la utilización del display e implementando un PIC16F84A:



Para encender el display son necesarios 5v, dicho valor es suministrado por los pines del PIC16F84A, con la configuración del circuito anterior observamos que el puerto que debemos colocar como salida es el ***PUERTOB.***

Ejemplo: Utilizando la configuración básica del circuito del display de 7 segmentos (Cátodo Común), encenderemos todos los segmentos del display, de la siguiente manera:

```
/* Main.c file generated by New Project wizard
 *
 * Created:   jue ago 28 2014
 * Processor: PIC16F84A
 * Compiler:  MPLAB XC8
 */

#include <xc.h>

void main(void)
 {
   //ESTABLECEMOS LA CONFIGURACION
   //================================
   TRISB = 0;                          //Configuramos el puerto como salida
   //================================
   while (1){
      PORTB =0b11111111;
    };
 }
```

Programación en C.

Para trabajar con el PIC16F84A en muchas ocasiones será necesario crear una pausa en el momento del procesamiento, esto nos permitirá realizar una espera para que se realice determinado proceso antes de que se dé un nuevo ciclo de procesamiento. Definiremos un macro para establecer dicha espera, y utilizaremos la palabra reservada ***#define.***

#define

#define crea una macro, que es la asociación de un identificador o identificador parametrizado con una cadena de token. Una vez definida la macro, el compilador puede sustituir la cadena de token para cada aparición del identificador del archivo de código fuente.

Ejemplo:

#define *identifier token-string* $_{opc}$
#define *identifier* **(** *identifier* $_{opc}$ **,** ... **,** *identifier* $_{opc}$ **)** *token-string* $_{opc}$

Además de utilizar macros en nuestro código de nuestro PIC16F84A, nos apoyaremos en muchas ocasiones en las funciones, que estas ya se encuentran definidas a nivel de código y no son incrustadas cuando se compilan. Cuando trabajemos con métodos o funciones podemos realizarlo de dos maneras:

- Declarando el prototipo de la función/método y luego su definición.
- Realizando la definición de la función/método sin prototipos.

Definición de Prototipos.

Un prototipo es una declaración de una función. Consiste en una presentación de la función, exactamente con la misma estructura que la definición, pero sin cuerpo y terminada con un ";". La estructura de un prototipo es:

```
[extern|static] <tipo_valor_retorno> [<modificadores>]<identificador>(<lista_parámetros>);
```

Definición de funciones.

Las funciones deben declararse, para lo que usaremos los prototipos, pero también deben definirse, una definición contiene además las instrucciones con las que la función realizará su trabajo, es decir, su código. La sintaxis de una definición de función es:

```
[extern|static] <tipo_valor_retorno> [modificadores] <identificador>(<lista_parámetros>)
{
  [sentencias]
}
```

La utilización de funciones y macros nos permitirá mantener un orden dentro de nuestro código, además nos permitirá estructurarlo de mejor manera.

Ejemplo: Utilizando el circuito base mostrado anteriormente, para el manejo del display de 7 segmentos, crearemos un circuito el cual nos permita realizar un conteo del 0 al 9, utilizando los macros crearemos la función delay, dicha función nos permitirá realizar una pausa acorde a la velocidad del reloj que tenemos configurado en nuestro cristal de cuarzo. Utilizando las funciones tanto con prototipos y sin prototipos, crearemos una función en la cual estableceremos los valores lógicos (***0 y 1***), en los pines del PUERTOB para encender los segmentos respectivos para mostrar dichos números (0 al 9).

El código del firmware se muestra en la siguiente página:

```
9
10  //DEFINIMOS LA FUNCION DELAY
11  //************************************
12  //Definimos la frecuencia del crystal de quarzo
13  #define FRECUENCIA 4000000L
14  //Definimos la funcion que se utilizara para crear el delay
15  #define delay_us(x){  unsigned int us; \
16                        us = (x) / (12000000 / FRECUENCIA ) | 1; \
17                        while(--us != 0) continue; }
18  //************************************
19  void contador(unsigned int valor);
20
21  /**
22  * Metodo para realizar una pausa definiendo los milisegundos
23  */
24  void delay_ms(unsigned int ms){
25     unsigned int i;
26     do{
27        i=4;
28        do{
29         delay_us(164) ;
30       }while(--i);
31       }while(--ms);
32  }
33
34  void main(void)
35   {
36     //ESTABLECEMOS LA CONFIGURACION
37     //==================================
38     TRISB = 0;               //Configuramos el puerto como salida
39     //==================================
40     while (1){
41        for(int i = 0; i <= 10; i++){
42          contador(i);
43       };
44      };
45  }
```

```
50  void contador(unsigned int valor){
51     delay_ms(100);
52     switch(valor){
53        case 1:
54       PORTB = 0b00000110;
55       break;
56        case 2:
57       PORTB = 0b01011011;
58       break;
59        case 3:
60       PORTB = 0b01001111;
61       break;
62        case 4:
63       PORTB = 0b01100110;
64       break;
65        case 5:
66       PORTB = 0b01101101;
67       break;
68        case 6:
69       PORTB = 0b01111101;
70       break;
71        case 7:
72       PORTB = 0b00000111;
73       break;
74        case 8:
75       PORTB = 0b01111111;
76       break;
77        case 9:
78       PORTB = 0b01100111;
79       break;
80        case 0:
81       PORTB = 0b00111111;
82       break;
83        default:
84       PORTB = 0b00000000;
85       break;
86     };
87  }
```

EJERCICIOS.

Para cada uno de los ítems se tiene que realizar en distintos proyectos utilizando el PIC16F84A.

1. Realizar el último ejemplo pero utilizando un ***button,*** esto quiere decir que el contador se realizara de forma manual, cada vez que se presione el ***button***, se debe de incrementar el valor mostrado en el display.

   Para la configuración del circuito, el puerto B será configurado como salida, mientras que el puerto A como entrada. Al momento de utilizar el puerto A como entrada y utilizando un botón la resistencia que se usa ya no será una resistencia ***PULLDOWN***, si no que será una resistencia configurada a 10k (***RESISTOR***).

2. Ya que sabemos cómo utilizar un ***button***, para manejar el contador de los números, crear una calculadora que realice las operaciones matemáticas de:
   - **Suma**: *Configurar un PIN del puerto A para determinar que es una suma.*
   - **Resta:** *Configurar un PIN del puerto A para determinar que es una suma.*
   - **Multi:** *Configurar un PIN del puerto A para determinar que es una multiplicación.*

   Ya que contamos con 5 pines hasta el momento se han utilizado 4, 1 que es el contador que mostrara los números, y los otros 3 que determinan las operación a realizar, el ultimo pin lo utilizaremos como botón ***igual.*** Esto quiere decir que los pasos a seguir de la calculadora serían los siguientes:

   1) Establecemos el valor numérico con el botón de contador. Eje: 4 (En el display).
   2) Presionamos uno de los botones de la operación. Eje: Suma.
   3) Establecemos el valor numérico con el botón de contador. Eje: 5 (En el display).
   4) Presionamos el botón igual para observar el resultado. Eje: 9 (En el display).