**Universidad de Cuenca**
Facultad de Ingeniería
Carrera de Ingeniería en Telecomunicaciones

Tyrone Novillo
tyrone.novillo@ucuenca.edu.ec

Practica 3
PIC18F4550 - Dispositivos de Visualización: Multiplexación de Displays 7 segmentos

# 1. Objetivos

- Visualizar información de carácter numérico en un conjunto de displays de 7 segmentos.
- Utilizar la técnica de multiplexación de displays mediante interrupción del TMR0 para optimizar el hardware.
- Realizar un dispositivo temporizador descendente controlado.

# 2. Marco teórico

## 2.1. Refrescamiento y multiplexación de displays de 7 segmentos

Los displays de 7 segmentos son dispositivos utilizados para mostrar información numérica y algunas letras en aplicaciones electrónicas. Estos displays están compuestos por siete LEDs dispuestos en forma de barra, denominados segmentos, que pueden encenderse de manera individual para formar diferentes dígitos y caracteres .
El refrescamiento de displays de 7 segmentos se refiere al proceso de actualizar la visualización de los segmentos encendidos a una frecuencia suficiente para que el ojo humano perciba una imagen estable. Debido a las limitaciones en la cantidad de pines de entrada/salida disponibles en un microcontrolador, es común que varios displays compartan los mismos pines para los segmentos, encendiendo solo uno a la vez a una velocidad muy alta. Este método asegura que todos los displays parezcan estar encendidos simultáneamente .

## 2.2. Multiplexación de Displays

La multiplexación es una técnica que permite controlar múltiples displays utilizando un conjunto reducido de pines del microcontrolador. En este método, los segmentos de todos los displays están conectados en paralelo, pero cada display tiene su propio pin de control que determina cuándo se enciende . Al activar un display a la vez y cambiar rápidamente entre ellos, es posible dar la impresión de que todos los displays están encendidos al mismo tiempo.
El proceso de multiplexación se realiza en varios pasos:
Activación Secuencial: Se activa un display a la vez mediante transistores o registros de desplazamiento que controlan los pines de activación de cada display. Actualización de Datos: Se envía el dato correspondiente al display activado a los pines comunes de los segmentos. Cambio Rápido: Se cambia al siguiente display y se repite el proceso a una frecuencia suficientemente alta para que el ojo humano no perciba el parpadeo .

## 2.3. Uso del Timer para la Multiplexación

Para implementar la multiplexación, se utiliza un temporizador (timer) del microcontrolador para generar interrupciones periódicas que controlan el ciclo de activación de los displays. En cada interrupción, el microcontrolador actualiza el display activo y avanza al siguiente en la secuencia. Configuraciones típicas

del timer incluyen el uso de un prescaler para ajustar la frecuencia de las interrupciones, asegurando una tasa de refresco adecuada .
En el caso del PIC18F4550, el timer TMR0 se puede configurar para operar en modo de 8 bits con un prescaler de 1:256. Al configurar el temporizador para generar interrupciones cada 1 ms, se puede secuenciar a través de los displays rápidamente. Cada display se enciende durante un breve período antes de pasar al siguiente, lo que se percibe como un encendido continuo de todos los displays debido a la persistencia de la visión .

### 2.4. Ventajas de la Multiplexación

La multiplexación de displays de 7 segmentos ofrece varias ventajas:

- Reducción de Pines: Permite controlar múltiples displays con un número reducido de pines del microcontrolador.
- Eficiencia en el Hardware: Minimiza el hardware necesario para la conexión de múltiples displays.
- Flexibilidad: Facilita el diseño de sistemas con múltiples displays sin requerir un aumento significativo en la complejidad del circuito .

## 3. Lista de materiales y equipos utilizados

- PIC18F4550 encapsulado de 40 pines
- Pulsantes (3)
- Resistencias de 10 k$\omega$ (3)
- Resistencias de 1 k$\omega$ (12)
- Led (1)
- Displays 7 segmentos cátodo común (4)
- Transistores 2N3904 (4)
- Capacitores cerámicos de 22 pF (2)
- Cristal de cuarzo de 12 MHz (1)
- Programador Pickit 3

## 4. Diseño del sistema microcontrolador

A continuación, se explicará el procedimiento para mostrar en varios displays 7 segmentos diferentes valores asignados mediante multiplexación de valores a una tasa de refrescamiento definida. Una vez definido esto, se implementará un contador de 10 minutos descendente en dichos displays 7 segmentos.

### 4.1. Plataforma de Hardware

La Figura 4.1 muestra el diagrama del circuito a desarrollar con todas sus conexiones a los diferentes componentes. Para hacer uso del método de multiplexación para escribir valores diferentes a los displays controlados por el mismo bus de datos, se utilizarán transistores 2N3904 NPN los cuales permiten encender y apagar cada uno de los 4 displays 7 segmentos. Adicionalmente, cada una de las entradas de los displays 7 segmentos están conectadas a los mismos pines de salida del puerto RD1 - RD7. Se conecta además un led piloto que indicará que el programa ha empezado a ejecutarse en el pin RD0.
Además, se han conectado dos botones (entradas) en los puertos RC4 y RC5 debido a que, según el datasheet del PIC, todos los pines del puerto C a excepción de estos dos son configurados como entradas digitales en el modo power-on Reset. Cabe mencionar que para usar RC4 y RC5 como entradas, el módulo USB ha sido descativado así como el on-chip USB transceiver desde los bits de configuración. Estos valores estarán activos en bajo, por lo que se han conectado resistencias en pull-up.
La necesidad de colocar una resistencia en pull-up en el puerto E3, se debe a que este pin es el MCLR (master clear reset), mismo que reinicia el microcontrolador para volver a un estado conocido, generalmente al inicio del programa. Es una forma de reiniciar manualmente el microcontrolador en ocasiones

necesarias. El cristal de cuarzo generalmente de 4 MHz, se conecta para proporcionar la señal de reloj al microprocesador, en los pines 13 y 14 ya que estos son los predeterminados para el oscilador.

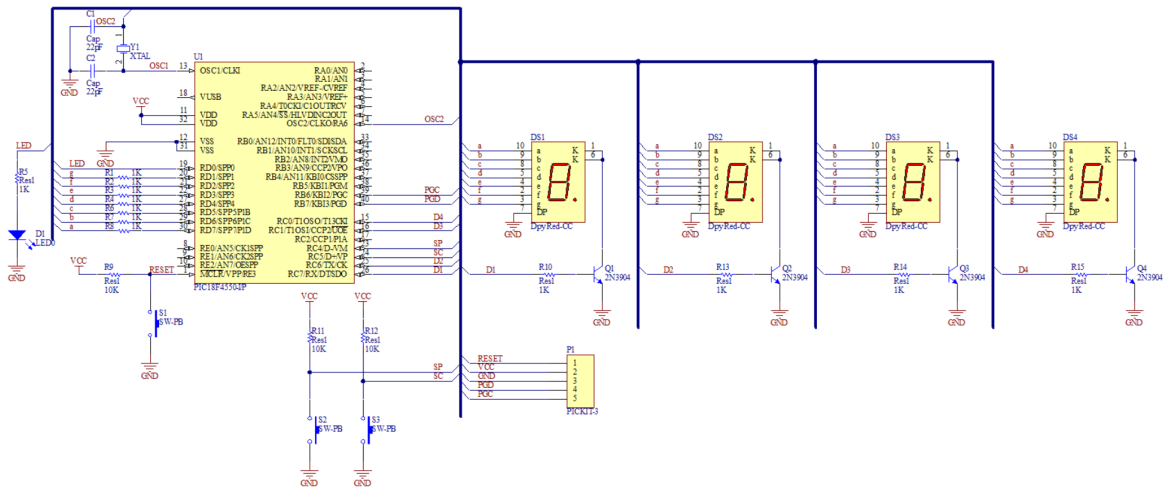


Figura 4.1: Circuito para manejo de Displays de 7 segmentos: PIC18F4550

## 4.2. Funcionamiento del Software

Cada uno de los siguientes programas requiere un led piloto que indica el inicio de los programas, dicho LED está conectado en el puerto RD0, por lo que se definirá el registro LATD, 0 en 1 indicando que es un puerto de salida. Esta etiqueta estará al inicio de cada uno de los programas siguientes.

### 4.2.1. Rutina de refrescamiento y multiplexación de displays

Cada uno de los siguientes programas requiere un led piloto que indica el inicio de los programas, dicho LED está conectado en el puerto RD0, por lo que se definirá el registro LATD, 0 en 1 indicando que es un puerto de salida. Esta etiqueta estará al inicio de cada uno de los programas siguientes.
Para lograr mostrar diferentes valores en cada uno de los displays conectados a los mismos pines, se trabajará con el timer TMR0. Para ello, se deben configurar los siguientes bits para dicho timer:

- Habilitación de interrupciones globales: El bit GIE (Global Interrupt Enable) en el registro INTCON se establece en 1. Esto habilita todas las interrupciones globales del microcontrolador, permitiendo que las interrupciones ocurran y sean atendidas por el sistema.
- Modo de operación de 8 bits: El bit T08BIT en el registro T0CON se configura en 1. Esto selecciona el modo de operación de 8 bits para el Timer0, lo que significa que el temporizador contará hasta 255 antes de desbordarse y reiniciar el conteo.
- Fuente de reloj interna: El bit T0CS en el registro T0CON se establece en 0. Esto selecciona la fuente de reloj interna para el Timer0, que suele ser el oscilador interno del microcontrolador.
- Uso del prescaler: El bit PSA (Prescaler Assignment) en el registro T0CON se configura en 0. Esto indica que el Timer0 utilizará el prescaler, un divisor de frecuencia que reduce la velocidad del reloj de entrada para el temporizador.
- Valor del prescaler:Los bits T0PS en el registro T0CON se configuran en 0b111. Esto selecciona un prescaler de 1:256, lo que significa que el reloj de entrada del Timer0 se dividirá por 256 antes de ser utilizado para el conteo.
- Valor inicial del TMR0: El registro TMR0L se establece en 0xD2. Esto carga el Timer0 con un valor inicial de 210 (en hexadecimal), lo cual se ha calculado para obtener un conteo de 1 milisegundo considerando la frecuencia de reloj y el prescaler configurado.
- Habilitación de la interrupción por desbordamiento del Timer0: El bit TMR0IE en el registro INTCON se configura en 1. Esto habilita la interrupción por desbordamiento del Timer0, permitiendo que se genere una interrupción cuando el Timer0 se desborde (pase de 255 a 0 en modo de 8 bits).
- Inicio del Timer0: El bit TMR0ON en el registro T0CON se establece en 1. Esto inicia el conteo del Timer0, poniendo en marcha el temporizador con la configuración especificada.

En resumen, estos valores configuran el Timer0 del PIC18F4550 en modo de 8 bits con un prescaler de 1:256, utilizando la fuente de reloj interna, y habilitan tanto las interrupciones globales como la interrupción específica del Timer0. El temporizador se inicia con un valor de conteo inicial para generar una interrupción cada 1 milisegundo. Para obtener el valor inicial con el que inicializa el conteo el timer para definir el tiempo de interrupción es el siguiente:

$$t = \text{Cantidad de instrucciones} \cdot \frac{1}{F_{osc}} \cdot 4 \cdot \text{Preescaler}$$

$$t = (256 - 210) \cdot \frac{1}{48MHz} \cdot 4 \cdot 256 = 1ms$$

Este cálculo se lo hace de igual manera para cada uno de los tiempos de encendido de cada uno de los displays que corresponde a la tasa de refresco de los 4 displays 7 segmentos. La subrutina de interrupción del timer TMR0 lo que hace es asignar un valor a un display apagando el resto de displays. Este proceso se lo hace cada interrupción, es decir, cada 1ms asigna al puerto D un valor que obtiene de un array al primer display encendiéndolo y apagando el resto mediante los valores de RC0, RC1, RC6 y RC7 que cierran el circuito de cada uno de los displays. Para iterar sobre cada display se hace uso de un contador que va de 0 a 3 (cantidad de displays).
Al configurar la tasa de refresco (tiempo de interrupción del timer TMR0) en 50ms, se puede observar que la interrupción enciende el primer display y lo asigna a un valor y luego se apaga para después encender el segundo display y escribir un valor sobre él. Para el tiempo de 25ms, el efecto es el mismo pero más rápido. Para el tiempo de refresco de 10ms, todavía se alcanza a visualizar el mismo efecto pero es mucho más rápido a tal grado de que los displays tienen una atenuación, es decir, no muestran el brillo completo de los displays. para el tiempo de 5ms, se pueden ver los displays encendidos al mismo tiempo pero no tienen toda la opacidad que los displays alcanzan a entregar. Finalmente, para el tiempo de refresco de 2.5ms se muestran al mismo tiempo los displays y con el brillo máximo. Esto se debe a que el ojo humano no alcanza a ver a una tasa de refresco muy pequeña y el efecto visual es que están encedidos los displays al mismo tiempo. Adicionalmente, cada número coincide con la posición de los elementos del array dis el cual contiene los valores que se va asignando al puerto para encender cada uno de los segmentos de los displays. Es decir, para asignar el número cero, se debe encender el segmento a, b, c, d, e, f, por lo que el número binario que se debe escribir en el puerto D será 11111101 siendo el último bit el correspondiente al led piloto que se debe mantener encendido.

```
int dis[14] = {0xFD, 0x61, 0xDB, 0xF3, 0x67, 0xB7, 0x3F, 0xE1, 0xFF, 0xF7, 0x6F, 0xFD, 0
    x1D, 0xEF};
int i=0;
int con = 0;
int sel = 0;
int d1 = 0;
int d2 = 0;
int d3 = 0;
int d4 = 0;

void RD0_Toggle(void){ // Toggel de D0
    LATDbits.LD0 = ~PORTDbits.RD0;
}
void init(void){ //set up inicial
     // perifericos configuracion
    UCONbits.USBEN = 0;
    UCFGbits.UTRDIS = 1;
    //**

    // E/S configuracion
    TRISD = 0x00; // segmentos de displays
    TRISC = 0x30; // activacion de displays y entradas de pulsantes
   //**


    // Condiciones iniciales
    LATD = 0x00; // display y otros perifericos en off inicial
    LATC = 0x00; //
    //****
}
//si s3 entonces esto no tiene que parpadear
//si s2 entonces esto tiene que hacer una transicion
void __interrupt() inter(void){
    if (INTCONbits.TMR0IF){
        // Aqui se prende cada display.
        // Limpia la bandera de interrupci n
```

```
        INTCONbits.TMR0IF = 0;
        TMR0L = 0x2E; //inicializa el conteo
        if (con == 0){
            LATC = 0x80; //Control
            LATD = d1; //Display
            con++;

        }else if(con == 1){
            LATC = 0x40; //Control
            LATD = d2; //Display
            con++;

        }else if (con == 2){
            LATC = 0x02; //Control
            LATD = d3; //Display
            con++;

        }else{
            LATC = 0x01; //Control
            LATD = d4; //Display
            con = 0;
        }

    }

}

// Programa principal

void main() {

    init();

    //Subrutina de inicialiacion
    //char es un tipo de datos de 8 bits
    for (char conteo = 0; conteo < 19 ; conteo++){
        RD0_Toggle();
        __delay_ms(250);
    }


    // Configuraci n del TMR0
    INTCONbits.GIE = 1; // habilta interrupciones globales
    T0CONbits.T08BIT = 1; // Modo de 8 bits
    T0CONbits.T0CS = 0; // Fuente de reloj interna
    T0CONbits.PSA = 0; // Utiliza el prescaler
    T0CONbits.T0PS = 0b111; // Prescaler 1:256

    // Valor inicial del TMR0
    TMR0L = 0x2E; // para conteo de 1 ms

    // Habilita la interrupci n por desbordamiento
    INTCONbits.TMR0IE = 1;

    // Inicia el temporizador


    T0CONbits.TMR0ON = 1; //iniciamos el temporizador


    // main loop, el LED en RD0 permanece encendido
    while(1){

        if(PORTCbits.RC4==0){
            d1 = 0x6F;
            d2 = 0x3B;
            d3 = 0x1D;
            d4 = 0xEF;
            INTCONbits.TMR0IF = 1; //Interrupcion forzada
            // iniciar la interrupcion

        }else if(PORTCbits.RC5==0){
            while(PORTCbits.RC4 == 1){
                d1 = 0x6F;
                d2 = 0x3B;
                d3 = 0x1D;
```

```
                d4 = 0xEF;
                __delay_ms(500);
                d1 = 0x01;
                d2 = 0x01;
                d3 = 0x01;
                d4 = 0x01;
                __delay_ms(500);
            }
        }
    }

    return;
}
```

#### 4.2.2. Temporizador de 10 minutos

Luego de inicializar el temporizador TMR0 y habilitar las interrupciones globales, el temporizador comienza a decrementar cuando se detecta que el botón conectado al pin RC4 se ha presionado. El tiempo que transcurre para cambiar los valores de unidades y decenas se lo controla con software con la función `__delay_ms(1000)`. Una vez se está ejecutando el temporizador, el temporizador de 10 minutos puede ser reseteado a su valor inicial si se detecta una señal en el botón conectado al pin RC5. Esto permite al usuario reiniciar el temporizador sin tener que esperar a que llegue a cero. Una vez que el temporizador llega a 00:00, se muestra la palabra boon indefinidamente. Todo el control del temporizador se ha realizado mediante blucles que escuchan constantemente a los botones de incio y reinicio.

## 5. Pruebas y verificaciones

Se ha verificado en el circuito que los displays muestran diferentes valores en cada display y adiconalmente, que el timer de 10 minutos al terminar de forma descendente muestra la palabra boon en los displays.

## 6. Conclusiones y Recomendaciones

El proyecto logró con éxito la visualización de información numérica en un conjunto de displays de 7 segmentos mediante la técnica de multiplexación. Este método permitió mostrar diferentes valores en cada uno de los displays, a pesar de estar conectados al mismo bus de datos. La implementación de la multiplexación a través de interrupciones del temporizador TMR0 resultó en una solución eficiente en términos de hardware, optimizando la cantidad de puertos del microcontrolador PIC18F4550. La frecuencia de refresco de los displays se configuró adecuadamente para evitar parpadeos visibles y mantener la claridad de los números mostrados. El uso de transistores NPN 2N3904 permitió controlar individualmente cada display, permitiendo encender y apagar cada uno de manera controlada y sincronizada. Se recuerda que el timer TMR0 ocurre de manera independiente y paralela a la función main, esto de cierta forma permite ejecutar varias acciones a la vez en el microprocesador.

## Referencias


[1] "Display de 7 segmentos," https://www.electro-tech-online.com/articles/seven-segment-display-basics.665/, accessed: 2024-06-04.

[2] "Multiplexación de displays," https://www.microcontrollertutorials.com/multiplexing-seven-segment-displays/, accessed: 2024-06-04.

[3] "Timer interrupts in pic microcontrollers," https://embedded-lab.com/blog/multiplexing-seven-segment-displays-using-timer-interrupts/, accessed: 2024-06-04.

[4] "Principles of led multiplexing," https://www.eetimes.com/led-multiplexing-techniques-and-principles/, accessed: 2024-06-04.

[5] "Prescalers and their uses," https://www.allaboutcircuits.com/technical-articles/understanding-prescalers-and-their-uses/, accessed: 2024-06-04.

[6] M. Technology, *PIC18F4550 Datasheet*, https://www.microchip.com/wwwproducts/en/PIC18F4550, 2007, accessed: 2024-06-04.

[7] "Advantages of display multiplexing," https://circuitdigest.com/tutorial/multiplexing-seven-segment-displays, accessed: 2024-06-04.

# 7. Preguntas

- ¿Por qué es necesario obtener los códigos de encendido para cada número correspondientes al display de 7 segmentos?

  Porque cada número se representa mediante una combinación única de segmentos iluminados. Conocer estos códigos, el microcontrolador puede activar las líneas correspondientes para mostrar el número deseado en el display. Esto permite una correcta visualización de los dígitos.

- ¿Cuál es la diferencia en los aspectos de programación si se hubiera utilizado un display de ánodo común con respecto a uno de cátodo común? En un display de cátodo común, los segmentos se encienden aplicando un nivel alto (1 lógico) a los pines correspondientes. En un display de ánodo común, los segmentos se encienden aplicando un nivel bajo (0 lógico) a los pines correspondientes. Esto implica que la lógica de encendido y apagado de los segmentos debe invertirse.

- En el circuito de la figura 4.1: ¿cuál es la función de los 4 transistores ubicados en los terminales comunes de los displays?

  Los transistores ubicados en los terminales comunes de los displays controlan el encendido y apagado de cada display de manera individual. Solo un display se activa a la vez, y los transistores permiten seleccionar cuál de los displays debe estar activo en un momento determinado. Al activar un transistor específico, se cierra el circuito para el display correspondiente, permitiendo que los datos en el bus común se muestren en ese display mientras los otros permanecen apagados.

- Presente una propuesta de circuito que permita encender un display de las mismas características del usado en esta práctica, con la diferencia de que se utilizará en un estadio, es decir con lámparas de mayor potencia: considere el caso de lámparas LED de 24V y 1A de consumo individual por segmento en cada display.

  Para encender un display en un estadio utilizando lámparas LED de 24V y 1A por segmento, se puede utilizar un driver de LED de alta potencia, como el ULN2803, que soporta voltajes y corrientes mayores, junto con MOSFETs de canal N para controlar cada segmento del display. Estos transistores aseguran que los transistores puedan manejar la corriente alta sin dañarse.

- Explique sus observaciones con respecto al cambio de lenguaje de programación desde ensamblador hacia C/C++.

  Programar en C/C++ generalmente resulta en un código más entendible debido a su sintaxis de alto nivel y estructuras de control avanzadas. Además, C/C++ permite el uso de bibliotecas estándar y funciones reutilizables, lo que facilita la programación. Sin embargo, el ensamblador ofrece un control más fino sobre el hardware y puede ser más eficiente en términos de rendimiento y uso de memoria. La transición a C/C++ puede implicar una menor eficiencia en ciertos casos, pero la mejora en la productividad y la capacidad de depuración suele compensar esta desventaja.