



Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Telecomunicaciones

Tyrone Novillo
tyrone.novillo@ucuenca.edu.ec
Javier Sotamba O.
javier.sotamba99@ucuenca.edu.ec

Practica 1 PIC18F4550

Manejo de entradas y salidas digitales

1. OBJETIVOS

- Implementar una plataforma de hardware para la operación de un microcontrolador.
- Reconocer la distribución de pines de un microcontrolador PIC18F4550.
- Manejar operaciones de entrada y salida (E/S) digitales del microcontrolador.
- Comprender la sintaxis del lenguaje ensamblador de Microchip.

2. MARCO TEÓRICO

2.1. PIC 18F4550

El PIC18F4550 es un microcontrolador (μ C) desarrollado por Microchip, este μ C es útil en aplicaciones que requieren alta capacidad de procesamiento de datos, ya que posee una arquitectura de 8 bits, además, una velocidad de reloj de hasta 48 MHz. Dentro de sus características también posee una memoria flash de 32 KB, para almacenar el programa de aplicación y una memoria RAM de 2 KB para almacenar los datos temporales durante la ejecución del programa. Posee la arquitectura Harvard [1].

Su estructura posee 40 pines para conexión, se destinan 5 puertos entre A y D. El puerto A es de 6 bits, los puertos B al D son de 8 bits cada uno, mientras que el puerto E posee únicamente 3 bits. 12 pines pueden ser utilizados como entradas analógicas. En la figura 2.1 se muestra el esquema del PIC18F4550.

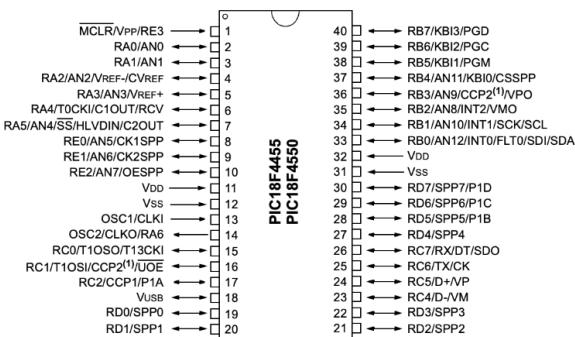


Figura 2.1: Diagrama de pines del PIC18F4550.

2.2. REGISTROS DEL PIC18F4550

Los registros del μ C son ubicaciones de memoria accesibles únicamente a través de programación, que sirven para almacenar y manipular datos mientras se ejecuta el programa [2].

Los registros más comunes del μ C son los registros TRISx, LATx y PORTx. Cada registro es utilizado en función del requerimiento, por lo tanto, el registro TRISx es utilizado para configurar un puerto específico como entrada o salida, mientras que el registro LATx es necesario para escribir un dato en el puerto x y por último, el registro PORTx se utiliza para leer un dato en el puerto x. Estos registros tienen sus propias ubicaciones dentro de la memoria de datos [3].

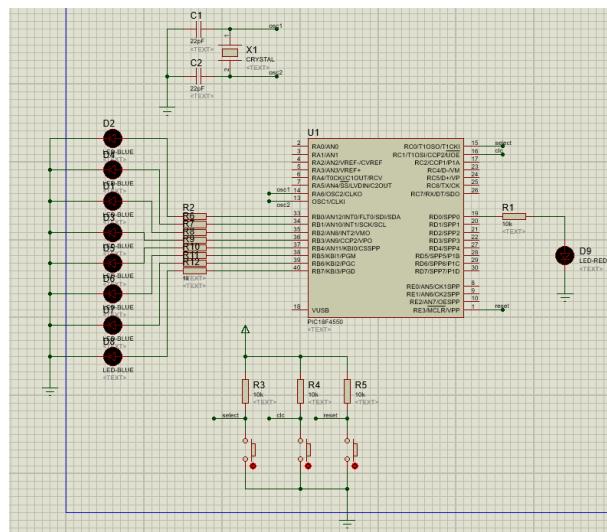
2.3. LISTA DE MATERIALES Y EQUIPOS UTILIZADOS

- PIC18F4550 encapsulado de 40 pines
- Pulsantes (3)
- Resistencias de 10 k ω (3)
- Resistencias de 1 k ω (9)
- Leds (9)
- Capacitores cerámicos de 22 pF (2)
- Cristal de cuarzo de 12 MHz (1)
- Programador Pickit 3

2.4. DISEÑO DEL SISTEMA MICROCONTROLADOR

2.4.1. PLATAFORMA DE HARDWARE

En la figura 2.2 se muestra el diagrama del circuito desarrollado en Proteus, con el que se pretende desarrollar la funcionalidad en el manejo de perifericos con el μ C.



implementar.

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.

Se declara los puertos RC0 y RC1 como entradas, por lo tanto, en estos se conectan resistencias en pull-up respectivamente.

El puerto RE3 se declara como salida, este es un LED indicador del inicio del programa.

3. 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.

```
LED_inicio:  
    bsf      LATD, 0      ;Apaga el led  
    movlw  d'250'        ;Se asigna al acumulador 250  
    call   Delay_ms     ;Delay generico para 250 ms  
    bcf      LATD, 0      ;enciende el LED en RD0;  
    movlw  d'250'        ;Se asigna al acumulador 250  
    call   Delay_ms     ;Delay generico para 250 ms  
    decfsz veces, 1     ;Se resta uno a la variable veces y se pregunta si es 0  
    goto  LED_inicio    ;Si no es cero, se repite el proceso  
    bsf      LATD, 0      ;Si es cero, se deja encendido el LED
```

3.1. MANEJO DE SALIDAS

Luego de la secuencia definida anteriormente, los LEDs conectados en el puerto B oscilarán siguiendo la siguiente secuencia:

- Durante 100ms, se mostrará el dato hexadecimal 0xAA
- Durante 100ms, se mostrará el dato hexadecimal 0x55

Para ello, simplemente se repetirá la lógica anterior y se irá iterando estos datos moviéndolos desde el acumulador hacia el registro LATB que sirve para escribir sobre el puerto B.

```
main:  
    movlw  0xAA          ;Se mueve el valor hexadecimal AA al acumulador  
    movwf  LATB;         ;Se mueve el contenido del acumulador al puerto B  
    movlw  d'250'        ;Se asigna al acumulador 250  
    call   Delay_ms     ;Delay generico para 250 ms  
    movlw  0x55          ;Se mueve el valor hexadecimal 55 al acumulador  
    movwf  LATB;         ;Se mueve el contenido del acumulador al puerto B  
    movlw  d'250'        ;Se asigna al acumulador 250  
    call   Delay_ms     ;Delay generico para 250 ms  
    goto   main          ;Se repite infinitamente
```

3.2. MANEJO DE ENTRADAS Y SALIDAS

Luego de la secuencia de inicio, se estará escuchando constantemente el pulsante S2, al presionarlo, la secuencia de luces inicia. Luego, al presionar S3 la secuencia de luces definida anteriormente debe parar y los LEDs deberán apagarse. Para eso, se llama constantemente a una etiqueta que pregunta si los pulsantes están en estado bajo. Si es el caso, se llama a una etiqueta específica.

```
consulta_1:                      ;Escucha infinitamente el boton S2  
    movlw  0x00  
    movwf  LATB          ;Apaga todos los LEDs del puerto B  
    btfsc  PORTC,0       ;Si RC0 (S2) esta en bajo se salta la siguiente linea  
    goto   consulta_1    ;Si RC0 (S2) no esta en bajo, se repite la escucha  
  
    ;Se espera un tiempo muy corto para asegurarse que el boton esta pulsado  
    ;para evitar efectos de rebote  
    movlw  d'50'
```

```

call  Delay_ms

btfscc  PORTC, 0
goto  consulta_1

secuencia:           ;Inicializacion de valores de la secuencia
    movlw  d'7'      ;Variable para saber cuantos desplazamientos debe hacer el byte
    movwf  Contador_led ;Contador para saber en que posicion de LED esta la secuencia
    movlw  0x01
    movwf  LATB        ;Se inicializa la secuencia con un bit encendido en el primer
                        ;LED
    goto  mover_izquierda ;Se inicia la secuencia desde derecha a izquierda

mover_izquierda:
    movlw  d'250'
    call  Delay_ms ;delay generico de 250ms

    btfss  PORTC,1      ;Si RC1 (S1) esta en alto (no pulsado) se salta la siguiente linea
    goto  consulta_1 ;Si RC1 (S1) no esta en alto, se vuelve al inicio del programa

            ;Se espera un tiempo muy corto para asegurarse que el boton esta pulsado
            ;para evitar efectos de rebote
    movlw  d'50'
    call  Delay_ms

    btfss  PORTC,1
    goto  consulta_1

    rlcfc  LATB, 1       ;Si no esta pulsado, se desplaza un bit hacia la izquierda el
                        ;byte
    decfsz Contador_led, 1 ;Se decrementa el contador que cuenta el LED actual de la
                            ;secuencia, si llega a 0, se salta la siguiente linea
    goto  mover_izquierda ;Se repite el movimiento hacia la izquierda

    movlw  d'7'      ;Si todos los 8 LEDs se encendieron, se empieza el desplazamiento
                        ;hacia la derecha
    movwf  Contador_led
    goto  mover_derecha

mover_derecha:
    movlw  d'250'
    call  Delay_ms ;delay generico para 250 ms

    btfss  PORTC,1
    goto  consulta_1

    movlw  d'50'
    call  Delay_ms

    btfss  PORTC,1
    goto  consulta_1

    rrcfc  LATB, 1
    decfsz Contador_led, 1
    goto  mover_derecha

    movlw  d'7'
    movwf  Contador_led
    goto  mover_izquierda

```

3.3. MANEJO AVANZADO DE ENTRADAS Y SALIDAS

Luego de la secuencia de inicio, al presionar S2, la secuencia de la parte 1 deberá iniciar, luego al aplastar S2 nuevamente la secuencia de la parte 2 debe iniciar. El accionar del pulsante S2 debe iterar entre secuencias. Al presionar S3 en cualquier instante, todas las luces de las secuencias deberán apagarse.

```

;-----
;Consulta de boton de inicio

consulta_1:           ;Escucha infinitamente el boton S2
    movlw  0x00
    movwf  LATB        ;Apaga todos los LEDs del puerto B
    btfscc  PORTC,0    ;Si RCO (S2) esta en bajo se salta la siguiente linea

```

```

goto consulta_1 ;Si RCO (S2) no esta en bajo, se repite la escucha

;Se espera un tiempo muy corto para asegurarse que el boton esta pulsado
;para evitar efectos de rebote

    movlw d'50'
    call Delay_ms

    btfsc PORTC, 0
    goto consulta_1

secuencia_1:
    movlw 0xAA
    movwf LATB; enciende el LED en RB0\
    movlw d'250' ;X=50
    call Delay_ms ;delay generico para X ms

    btfss PORTC,1
    goto consulta_1

    btfss PORTC,0
    goto secuencia_2

    movlw 0x55
    movwf LATB ;enciende el LED en RB0;
    movlw d'250' ;X=50
    call Delay_ms ;delay generico para X ms

    btfss PORTC,1
    goto consulta_1

    btfss PORTC,0
    goto secuencia_2

    goto secuencia_1

secuencia_2:           ;Inicializacion de valores de la secuencia
    movlw d'7'          ;Variable para saber cuantos desplazamientos debe hacer el byte
    movwf Contador_led ;Contador para saber en que posicion de LED esta la secuencia
    movlw 0x01
    movwf LATB          ;Se inicializa la secuencia con un bit encendido en el primer
                         LED
    goto mover_izquierda ;Se inicia la secuencia desde derecha a izquierda

mover_izquierda:
    movlw d'250'
    call Delay_ms ;delay generico de 250ms

    btfss PORTC,1      ;Si RC1 (S1) esta en alto (no pulsado) se salta la siguiente
linea
    goto consulta_1    ;Si RC1 (S1) no esta en alto, se vuelve al inicio del programa

    btfss PORTC,0
    goto secuencia_1

    rlcfs LATB, 1       ;Si no esta pulsado, se desplaza un bit hacia la izquierda el
                         byte
    decfsz Contador_led, 1 ;Se decrementa el contador que cuenta el LED actual de la
                           secuencia, si llega a 0, se salta la siguiente linea
    goto mover_izquierda ;Se repite el movimiento hacia la izquierda

    movlw d'7'          ;Si todos los 8 LEDs se encendieron, se empieza el
desplazamiento hacia la derecha
    movwf Contador_led
    goto mover_derecha

mover_derecha:
    movlw d'250'
    call Delay_ms

    btfss PORTC,1      ;Si RC1 (S1) esta en alto (no pulsado) se salta la siguiente
linea
    goto consulta_1    ;Si RC1 (S1) no esta en alto, se vuelve al inicio del programa

    btfss PORTC,0

```

```

    goto  secuencia_1

    rrcf      LATB , 1
    decfsz   Contador_led , 1
    goto    mover_derecha

    movlw    d'7'
    movwf    Contador_led
    goto    mover_izquierda

```

3.4. APORTE INDIVIDUAL

Luego de la secuencia de inicio, el programa deberá funcionar como antes. Sin embargo se adicionará una tercera secuencia que se activará al presionar S2 por una tercera ocasión.

```

;-----  

;  

;Consulta de boton de inicio  

;  

consulta_1:           ;Escucha infinitamente el boton S2
    movlw  0x00
    movwf  LATB          ;Apaga todos los LEDs del puerto B
    btfsc  PORTC,0        ;Si RCO (S2) esta en bajo se salta la siguiente linea
    goto   consulta_1     ;Si RCO (S2) no esta en bajo, se repite la escucha

    ;Se espera un tiempo muy corto para asegurarse que el boton esta pulsado
    ;para evitar efectos de rebote

    movlw d'50'
    call   Delay_ms

    btfsc  PORTC, 0
    goto   consulta_1

secuencia_1:
    movlw  0xAA
    movwf  LATB;enciende el LED en RB0\
    movlw d'250'          ;X=50
    call   Delay_ms      ;delay generico para X ms

    btfss  PORTC,1
    goto   consulta_1

    btfss  PORTC,0
    goto   secuencia_2

    movlw  0x55
    movwf  LATB          ;enciende el LED en RB0;
    movlw d'250'          ;X=50
    call   Delay_ms      ;delay generico para X ms

    btfss  PORTC,1
    goto   consulta_1

    btfss  PORTC,0
    goto   secuencia_2

    goto   secuencia_1

secuencia_2:           ;Inicializacion de valores de la secuencia
    movlw  d'7'            ;Variable para saber cuantos desplazamientos debe hacer el byte
    movwf  Contador_led    ;Contador para saber en que posicion de LED esta la secuencia
    movlw  0x01
    movwf  LATB            ;Se inicializa la secuencia con un bit encendido en el primer
                           ;LED
    goto   mover_izquierda ;Se inicia la secuencia desde derecha a izquierda

mover_izquierda:
    movlw d'250'
    call   Delay_ms      ;delay generico de 250ms

    btfss  PORTC,1        ;Si RC1 (S1) esta en alto (no pulsado) se salta la siguiente
                           ;linea
    goto   consulta_1     ;Si RC1 (S1) no esta en alto, se vuelve al inicio del programa

```

```

        btfss    PORTC,0
        goto    secuencia_3

        rlcf    LATB, 1           ;Si no esta pulsado, se desplaza un bit hacia la izquierda el
                                ;byte
        decfsz  Contador_led, 1 ;Se decrementa el contador que cuenta el LED actual de la
                                ;secuencia, si llega a 0, se salta la siguiente linea
        goto    mover_izquierda ;Se repite el movimiento hacia la izquierda

        movlw    d'7'             ;Si todos los 8 LEDs se encendieron, se empieza el
                                ;desplazamiento hacia la derecha
        movwf    Contador_led
        goto    mover_derecha

mover_derecha:
        movlw  d'250'
        call   Delay_ms

        btfss    PORTC,1           ;Si RC1 (S1) esta en alto (no pulsado) se salta la siguiente
                                ;linea
        goto    consulta_1         ;Si RC1 (S1) no esta en alto, se vuelve al inicio del programa

        btfss    PORTC,0
        goto    secuencia_3

        rrcf    LATB, 1
        decfsz  Contador_led, 1
        goto    mover_derecha

        movlw    d'7'
        movwf    Contador_led
        goto    mover_izquierda

secuencia_3:
        movlw    b'11000000'
        movwf    LATB
        movlw    d'250'
        call   Delay_ms

        btfss    PORTC,1
        goto    consulta_1

        btfss    PORTC,0
        goto    secuencia_1

        movlw    b'00110000'
        movwf    LATB
        movlw    d'250'
        call   Delay_ms

        btfss    PORTC,1
        goto    consulta_1

        btfss    PORTC,0
        goto    secuencia_1

        movlw    b'000001100'
        movwf    LATB
        movlw    d'250'
        call   Delay_ms

        btfss    PORTC,1
        goto    consulta_1

        btfss    PORTC,0
        goto    secuencia_1

        movlw    b'000000011'
        movwf    LATB
        movlw    d'250'
        call   Delay_ms

        btfss    PORTC,1
        goto    consulta_1

```

```
btfss    PORTC,0  
goto    secuencia_1  
  
goto    secuencia_3
```

4. PRUEBAS Y VERIFICACIONES

Las Figuras 4.1, 4.2, 4.3 muestran las secuencias del programa de la parte 4.

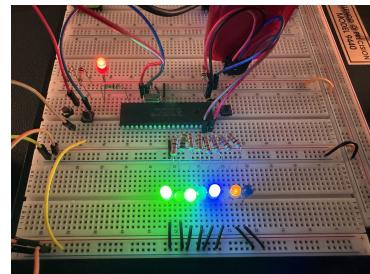


Figura 4.1: Secuencia 1 implementada

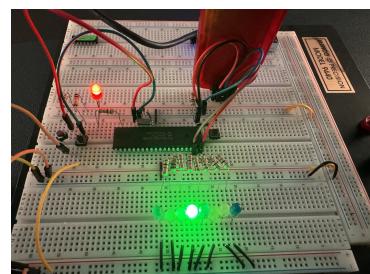


Figura 4.2: Secuencia 2 implementada

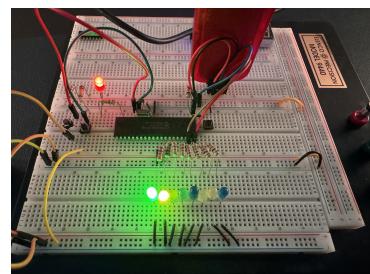


Figura 4.3: Secuencia 3 implementada

Adicionalmente, la Figura 4.4 indica que el programa con más procesos consume poca memoria.

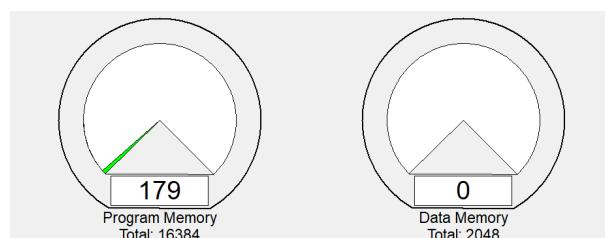


Figura 4.4: Uso de Memoria del programa de la parte 4

5. CONCLUSIONES Y RECOMENDACIONES

Se ha comprobado el uso correcto de los registros necesarios, en este caso, TRISx, LATx y PORTx, y que cada uno de ellos está definido para tareas específicas, como declarar un puerto como entrada o salida, escribir un dato y leer un dato respectivamente.

Se ha utilizado correctamente las instrucciones orientadas a bit y las instrucciones orientadas a byte, como medio para mejorar la eficiencia en la declaración de registros.

Se ha optado por usar sentencias `goto` en vez de `call` debido a que estas últimas consumen memoria de pila. Aún así, bajo recomendación del docente, cuando se usa la sentencia `call` para hacer branching, no es óptimo salir de la nueva secuencia con un `goto` a menos que dicha línea esté dentro del mismo proceso llamado por el `call`. Además, siempre una etiqueta llamada por un `call` debe contener un return debido a que generan apuntadores en la memoria de stack.

REFERENCIAS

- [1] “ microchip pic18f4550 — microcontrolador pic.” [Online]. Available: <https://microcontroladoress.com/pic/pic18f4550/>
- [2] “Indice.” [Online]. Available: www.microchip.com
- [3] “Io puertos digitales tris, port, lat con pic18f4550 - makerelectronico.” [Online]. Available: <https://www.makerelectronico.com/io-puertos-digitales-tris-port-lat-con-pic18f4550/>

6. PREGUNTAS

- ¿Cuál es la función del cristal de cuarzo (XTAL) conectado al microcontrolador?

El cristal proporciona la señal externa de relój al microcontrolador.

- Explique cuál es y cómo se obtiene la velocidad del oscilador del CPU

En este caso se usa un oscilador externo, este es un cristal de cuarzo de 12 MHz, se activa el PLL mediante la configuración de bits para obtener un prescaler de 1:3, y por lo tanto, la velocidad del oscilador del CPU es de 48 MHz.

- ¿Por qué es necesario configurar los registros TRIS para manipular los puertos de entrada y salida digitales del microcontrolador?

Si no se configuran los puertos como salidas no se puede escribir datos sobre ellos, al igual que si no se declara los puertos como entradas, entonces no se puede leer datos de estos. Además, no sería posible la administración de periféricos, indicadores o por otra parte, la obtención de información.

- Explique cómo se calcula el tiempo de las demoras por software que implementó. Se ha configurado el PIC para que trabaje a una frecuencia de 48MHz. Por lo que el tiempo de oscilación por instrucción considerando que cada instrucción consume 4 ciclos de máquina es:

$$T = \frac{1}{F_{osc}} \cdot 4 = \frac{1}{48MHz} \cdot 4 = 83,33ns$$

Los contadores lo que hacen es generar bucles que se asemejan a los for para generar instrucciones que suman tiempos en múltiplos de 83.33ns.

- Explique cómo consiguió que un mismo micropulsante (entrada digital) pueda utilizarse para diferentes funciones de su programa (en este caso, las secuencias de salida).

Se utilizan btfsc para testear si el pulsante recibe una señal de bajo, en un primer evento si esto sucede entonces, se genera la primera secuencia, dentro de esta secuencia está incluida la intrucción btfsc para testear si existe un nuevo evento para el mismo pulsante, si esto sucede nuevamente, entonces se hace que el programa se mueva a una próxima etiqueta.

- Explique las diferencias que encontró en el lenguaje ensamblador, sobre los otros lenguajes de programación convencionales que conoce; concretamente, sobre la forma de implementar estructuras de control: if, then, while, etc.

La diferencia entre el lenguaje ensamblador y los demás lenguajes de programación, es que no se puede implementar un estamento if, for, while, case, etc. De echo el lenguaje ensamblador se desarrolla comparando valores, moviendo valores de registros, decrementando los valores, saltándose las instrucciones, una manera de obtener un if en ensamblador es utilizando las instrucciones de decrementar, por ejemplo si utilizamos la instrucción DECFSZ, esta instrucción decremente el contenido del registro que se le indique y si llega a cero entonces se salta una línea de código y continúa.

En el desarrollo de este programa, una funcionalidad similar al ciclo while se desarrolló mediante el uso de etiquetas, a las que se las podía referenciar mediante la instrucción goto. Y si la instrucción escrita una línea antes del goto se ejecutaba y como resultado se salta la línea del goto, entonces, se puede dar por terminado el ciclo.