



UNIVERSIDAD Blas Pascal

Contador con librería de retardos y display 7 segmentos.

Asignatura: Microprocesadores

Fecha: 18/05/2023

Autor: BAIGORRIA, Constanza
CAMPELLONE, Victoria
GOMEZ, Martina

Profesor: Ing. Ezequiel Giovanardi

RESUMEN

En este informe veremos como es posible realizar un contador ascendente de 0 a 9 utilizando el PIC como salida para la conexión con el display de 7 segmentos de ánodo común, y utilizando la librería de retardos de tiempo.



INDICE:

1. INTRODUCCIÓN	3
2. DESARROLLO.....	3
2.1 MATERIALES	3
2.1.1 CONSTRUCCION	3
3. RESULTADOS.....	4
4. CONCLUSIONES	6

1. INTRODUCCIÓN

En el siguiente informe, daremos a conocer cómo es que, con el programa MPLAB, logramos crear un código para un contador el cual irá de forma ascendente, de 0 a 9, y este será mostrado en un display luego de la implementación del código, el cuál contendrá la librería de retardos de tiempo para el correcto funcionamiento. Se demostrará el proceso utilizando el PIC.

2. DESARROLLO

2.1 MATERIALES

Lo mencionado anteriormente, será posible por el lenguaje ASSEMBLER que otorga MPLAB. Con el uso de este lenguaje, y de los siguientes componentes, podremos realizar esta actividad de la que hablaremos más a detalle.

- MPLAB: Programa que nos permite crear un código en lenguaje Assembler.
- PROTEUS: Programa de simulación para crear la placa con el PIC para corroborar que no se queme con el voltaje seleccionado.
- Un PIC 16f84a: Microcontrolador de 8 bits, 18 pines y un conjunto de instrucciones, de la familia PIC perteneciente a la gama media. Este tipo de PIC puede ser programado tanto en ASSEMBLER como en BASIC.
- Una placa RASPBERRY: Serie de ordenadores de placa simple de bajo costo.
- Display 7 segmentos de ánodo común.

2.2 CONSTRUCCION

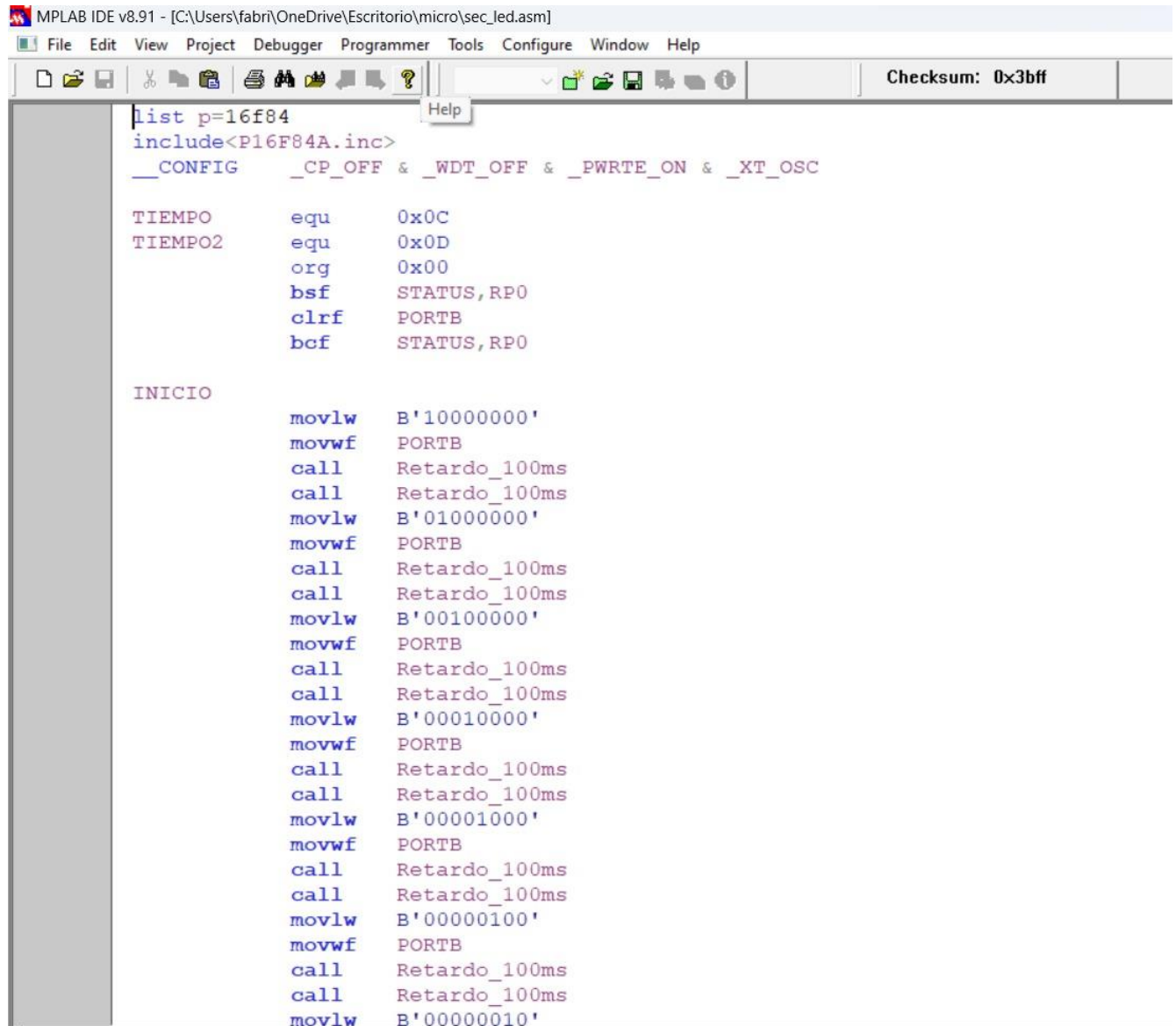
El programa comienza por establecer el valor 10000000 en el puerto B, lo que significa que el bit más significativo del puerto se establece en 1 y los demás en 0. Luego, se llama a la subrutina retardo_100ms, que realiza un retardo de 100 ms.

Como consecuente, el valor en el puerto se cambia a 01000000, esto nos dice que el segundo bit del puerto se establece en 1 y los demás en 0. Este proceso se repite para cada bit del puerto, de modo que cada bit se establece en 1 secuencialmente, con un retraso de 100 ms entre cada cambio. Se restablece el valor del puerto a 00000000 y se repite el ciclo desde el principio.

Después de algunos ciclos de cambio de bits del puerto, se establece el valor 11111111 en el puerto, lo que significa que todos los bits del puerto se establecen en 1. Luego, se restablece el valor del puerto a 00000000 y se establece el valor 10101010 en el puerto, dando a entender que los bits pares se establecen en 1 y los impares en 0. Después de un retardo, se establece el valor 01010101 en el puerto, por lo cual los bits impares se establecen en 1 y los pares en 0.

Este ciclo se repite indefinidamente.

3. RESULTADOS



MPLAB IDE v8.91 - [C:\Users\fabri\OneDrive\Escritorio\micro\sec_led.asm]

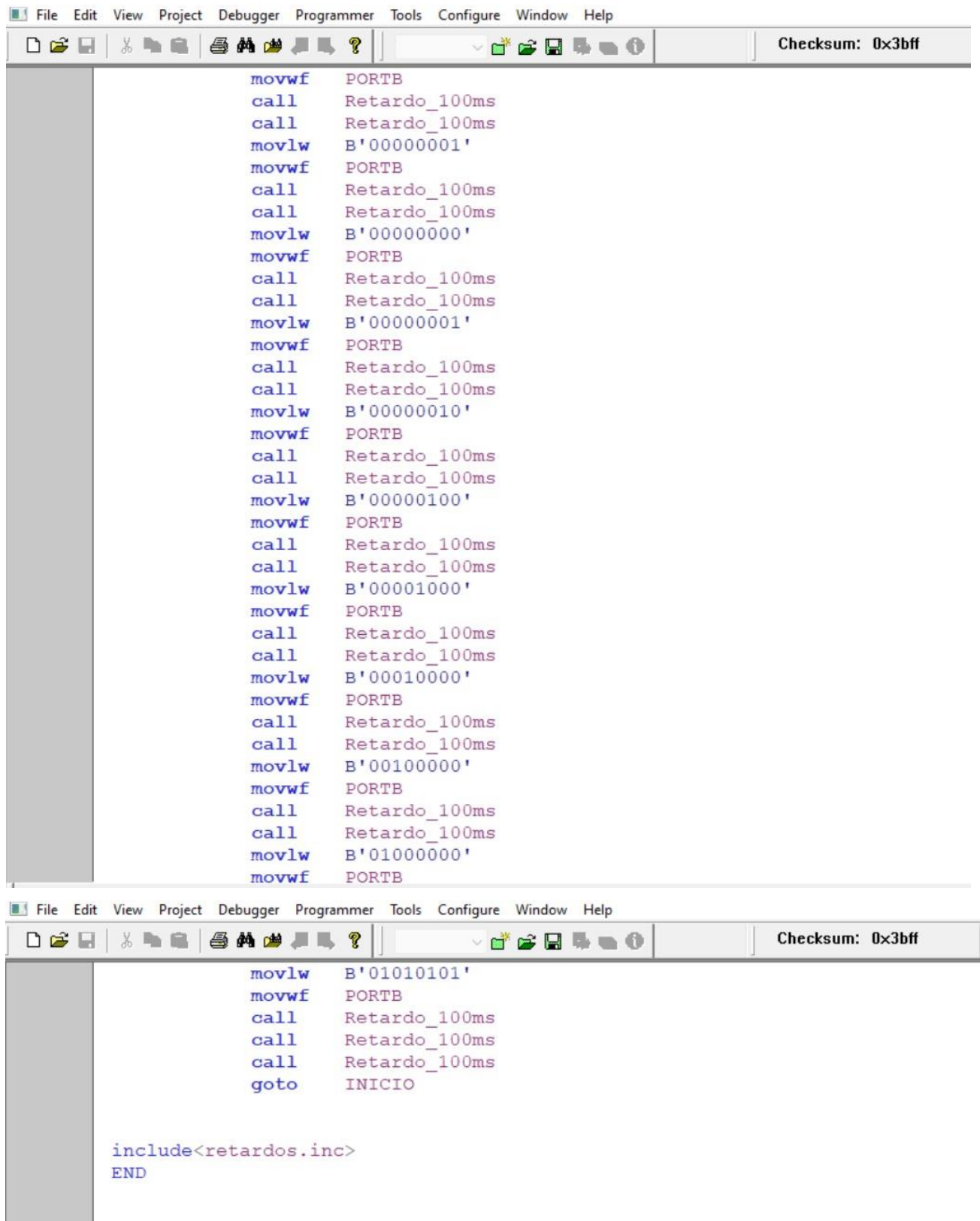
File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x3bff

```
list p=16f84
include<P16F84A.inc>
__CONFIG    _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

TIEMPO      equ    0x0C
TIEMPO2     equ    0x0D
org         0x00
bsf         STATUS,RP0
clrf        PORTB
bcf         STATUS,RP0

INICIO
    movlw   B'10000000'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'01000000'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'00100000'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'00010000'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'00001000'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'00000100'
    movwf   PORTB
    call    Retardo_100ms
    call    Retardo_100ms
    movlw   B'00000010'
```



```
File Edit View Project Debugger Programmer Tools Configure Window Help
Checksum: 0x3bff

movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00000001'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00000000'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00000001'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00000010'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00000100'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00001000'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00010000'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'00100000'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
movlw B'01000000'
movwf PORTB

movlw B'01010101'
movwf PORTB
call Retardo_100ms
call Retardo_100ms
call Retardo_100ms
goto INICIO

include<retardos.inc>
END
```

Lo que obtenemos con nuestro código es un contador ascendente de 0 a 9, mostrado por un display de 7 segmentos.

4. CONCLUSIONES

Este código es un programa para controlar los puertos del microcontrolador PIC16F84A, en particular el puerto B. El código utiliza una serie de instrucciones de movimientos y llamadas a la rutina de retardo para lograr una secuencia de encendido y apagado de los pines de puerto B, en un patrón determinado. Luego de la secuencia, el programa vuelve a la etiqueta INICIO para comenzar de nuevo la secuencia. El código también incluye una directiva de configuración que establece los valores de los bits de configuración del microcontrolador.

En resumen, este código es un ejemplo de cómo utilizar el microcontrolador PIC16F84A para controlar los puertos de entrada/salida. Este código es un programa para controlar los puertos del microcontrolador PIC16F84A, en particular el puerto B. El código utiliza una serie de instrucciones de movimientos y llamadas a la rutina de retardo para lograr una secuencia de encendido y apagado de los pines de puerto B, en un patrón determinado. Luego de la secuencia, el programa vuelve a la etiqueta INICIO para comenzar de nuevo la secuencia.

El código también incluye una directiva de configuración que establece los valores de los bits de configuración del microcontrolador. En resumen, este código es un ejemplo de cómo utilizar el microcontrolador PIC16F84A para controlar los puertos de entrada/salida.

Con esto aclarado, obtenemos un buen funcionamiento de todo el sistema, y logramos la finalidad del programa.