UNIVERSIDAD
Blas Pascal

SECUENCIAS

Asignatura: Microprocesadores

Fecha:

Autores: BAIGORRIA, Constanza

CAMPELLONE, Victoria

DOUGLAS, Octavio

GOMEZ, Martina

SOSA, Matias

Profesor: Ing. Ezequiel Giovanardi

SECUENCIAS

Por:

BAIGORRIA, Constanza

CAMPELLONE, Victoria

DOUGLAS, Octavio

GOMEZ, Martina

SOSA, Matias

ÍNDICE:

1. INTRODUCCIÓN

En el siguiente informe daremos a conocer cómo es que, con el programa MPLAB, logramos crear un código para el encendido y apagado secuencial de 8 leds y luego este mismo es enviado a un pic16f84a que le habilita la placa RASPBERRY a ejecutar dicho código.

Con el lenguaje de programación ASSEMBLER, nosotros somos capaces de programar diferentes cosas. En este caso lo que logramos es crear una secuencia de encendido/apagado de 8 leds.

Para lograr esto es necesario contar con:

- 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 es puede ser programado tanto en ASSEMBLER como en BASIC.
  Una placa RASPBERRY: Serie de ordenadores de placa simple de bajo costo.
- Leds a utilizar.

2. DESARROLLO

Al momento de comenzar a crear el código, debemos tener en cuenta que tenemos que guardarlo con la extensión ". ams" ya que de esta forma será posible ejecutarlo correctamente.

Una vez creado el archivo con esta extensión, es posible comenzar a crear nuestro código en el lenguaje ASSEMBLER.

Para comenzar a crear el código debemos tener en cuenta que es cada cosa que debemos usar. Como esto es una secuencia de encendido/apagado de leds, necesitamos:

- Un controlador de tiempo.
- Un controlador de voltaje.
- Una función que inicie el programa.
- Un encendido y un apagado.
- Una función que permita que dicho proceso se repita indefinidamente.
- Una función que nos permita demorar al encendido/apagado de nuestros leds.

```
list p=16f84
include<P16F84A.inc>
__CONFIG    _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC;
; CONFIGURACION:
;_CP_OFF = Permite que se haga el proceso inverso (desde el pic puedo obtener el codigo)
;_WDT_OFF = Indica si rebaja la pila (si llega al tope de la pila, reinicia el pic)
;_PWRTE_ON = Mide si tiene 5 v el pic (si no lo tiene, no enciende)
;_XT_OSC = Oscilador externo al pic (piedra de cuarzo)

TIEMPO      equ     0x0C; Coloca la variable tiempo en el banco 0c de la memoria
TIEMPO2     equ     0X0D; Coloca la variable tiempo2 en el banco 0d de la memoria
            org     0x00; Define el inicio de la memoria en el 00
            bsf     STATUS,RP0; Pone en 1 el bit 5 (RP0) del registro STATUS (define el output)
            clrf    PORTB; Coloca todos los bits del puerto b en 0
            bcf     STATUS,RP0; Vuelve a 0 el RP0 para trabajar en el banco 1 de la memoria


INICIO
;FUNCION DEL PROGRAMA:  hacer una secuencia de led que encienda del centro hacia afuera

            movlw   B'00011000'; Le coloca los valores binarios al registro w
;(los 1 son led encendidos/los 0 led apagados)
            movwf   PORTB; Prende los led 3/4
            call    RETARDO; Llama a la subrutina RETARDO.
            call    RETARDO

            movlw   B'00100100'
            movwf   PORTB; Prende los led 2/5
            call    RETARDO; Llama a la subrutina RETARDO.
            call    RETARDO



            movlw   B'01000010'
            movwf   PORTB; Prende los led 1/6
            call    RETARDO; Llama a la subrutina RETARDO.
            call    RETARDO

            movlw   B'10000001'
            movwf   PORTB; Prende los led 0/7
            call    RETARDO; Llama a la subrutina RETARDO.
            call    RETARDO

            goto    INICIO; Genera el loop para que se repita indefinidamente el programa.




RETARDO
            movlw   D'255'; Le pone el valor 255 en decimal a la variable TIEMPO
            movwf   TIEMPO; Le resta 1 al valor de Tiempo
DEC         decfsz  TIEMPO; Comienza la subrutina DEC donde pregunta si el valor de tiempo es 0
            goto    DEC1; Genera un loop hasta que el valor de tiempo2 sea 0
            return; Vuelve a donde comienza la rutina
DEC1        movlw   D'255'; Le pone el valor 255 en decimal a la variable TIEMPO2
            movwf   TIEMPO2; Le resta 1 al valor de TIEMPO2
DEC2        decfsz  TIEMPO2; Comienza la subrutina DEC2 donde pregunta si el valor de TIEMPO2 es 0
            goto    DEC2; Genera loop a la instruccion anterior.
            goto    DEC; Vuelve a la subrutina DEC.

END
```

Nuestro código presenta todos los puntos anteriormente mencionados. Además, obtenemos diferentes variables que son propias de nuestro lenguaje usado tales como:

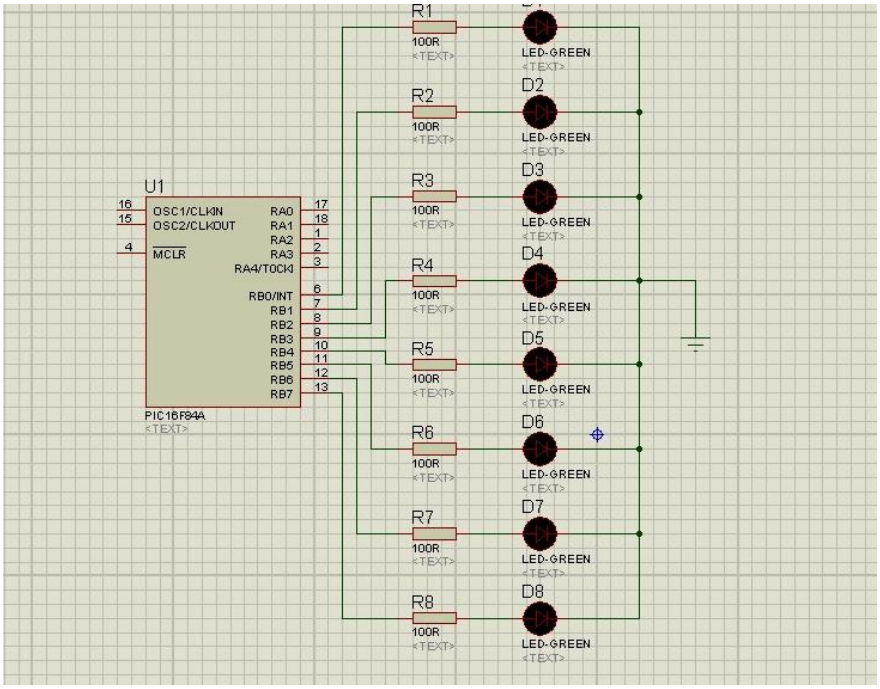
- EQU: Asigna un nombre simbólico al valor de una expresión. El compilador del código, cuando encuentre dicho nombre simbólico, lo sustituirá por el valor de dicha expresión.
- ORG: Define memoria a partir de la cual el programa se guarda.
- BSF: Pone en 1 al bit B del archivo F.

- CLRF: El contenido del registro F se pone en 0: 0x00.
- BCF: Pone en 0 al bit B del archivo F.
- MOVLW: Carga un numero en el acumulador W. El número que se va a cargar en al acumulador está representado por k, este número puede escribirse en decimal, hexadecimal o binario
- MOVWF: Mueve una copia del acumulador W al registro f.
- CALL: nos permite llamar a un procedimiento para que este se ejecute indeterminadas veces. También salta a una subrutina.
- GOTO: Me permite transferir el control a un punto determinado del código donde debe continuar la ejecución.
- DECFSZ: Salta la siguiente instrucción si el resultado es 0.

Estas son variables y funciones que necesitamos para realizar nuestro código secuencial de encendido/apagado de leds. Cabe aclarar que todo esto este hecho en nuestro programa MPLAB que nos permite compilar y verificar si funciona nuestro código.

Luego a este mismo los trasladamos a PROTEUS2 en donde verificamos el voltaje que necesitamos para que nuestros leds prendan y no se quemen en el intento. Al simular esto logramos ver que debemos cambiar, agregar o eliminar.

3. RESULTADOS



Obteniendo una simulación así, lo que obtenemos con nuestro código es una secuencia de encendido y apagado de las luces led que aparecen aquí.

Con el código se realiza una secuencia desde fuera hacia dentro.

4. CONCLUSIÓN

Logramos ver cómo es posible realizar una secuencia de encendido/apagado de 8 leds conectados a una placa RASPBERRY con un pic16f84a.

Pudimos observar cómo es el código necesario para realizarlo, cabe aclarar que en este caso nuestra secuencia va desde afuera hacia dentro, pero es posible hacerlo en el orden que nosotros queramos, pero el procedimiento será exactamente igual.

También tenemos que tener en cuenta que, a nuestro PIC, debemos conectarlo correctamente a la placa, ya que no solo el código no funcionara, sino que además podríamos dañar los componentes.

No podemos olvidarnos de controlar el voltaje de nuestra placa para que no se quemen los leds y tampoco el PIC.

Para obtener un buen funcionamiento de todo este sistema, tenemos que segur el orden, programar el código, el mismo pasa el PIC y luego a la placa.

5. REFERENCIAS

https://www.estudioelectronica.com/wpcontent/uploads/2018/10/instrucciones_1.pdf
https://es.wikipedia.org/wiki/Raspberry_Pi
https://es.wikipedia.org/wiki/PIC16F84