|  |  |
| --- | --- |
|  | **TECNOLÓGICO NACIONAL DE MÉXICO**  **TECNOLÓGICO DE ESTUDIOS SUPERIORES DE JILOTEPEC** |

**DIVISIÓN DE INGENIERÍA EN SISTEMA COMPUTACIONALES**

**ARQUITECTURA DE COMPUTADORAS**

**MANUAL SEMAFORO PIC 16F84A**

**JUAN FIGUEROA TREJO**

**GRUPO 3011**

**LIC. JUAN ALBERTO MARTINEZ ZAMORA**

**JILOTEPEC, MÉXICO**

1. **DESCRIPCION DEL PROBLEMA**

Los semáforos constan de 4 semáforos para peatones y 4 para los vehículos, los semáforos 1 y 3 se comportan de la misma manera, lo mismo pasa con los semáforos 2 y 3.

Así que :

* Cuando el semáforo 1 y 3 permitan el paso de vehículos el 2 y 4 deben negarlo
* Cuando la luz verde o ámbar este encendida, la roja de los peatones debe estar encendida,
* Cuando la luz roja del semáforo este encendida, la verde de los peatones debe estar encendida, excepto en los últimos 10 seg. Cuando esta debe parpadear

1. **ASIGNACION DE PINES**

En total se tendrán 10 señales :

Semáforo 1: Verde (V1), Ámbar(A1), Rojo(R1) para los vehículos, para los peatones: Verde (VP1), Rojo (RP1).

Semáforo 2: Verde (V2), Ámbar(A2), Rojo(R2) para los vehículos, para los peatones: Verde (VP2), Rojo (RP2).

Semáforo 3: Verde (V3), Ámbar(A3), Rojo(R3) para los vehículos, para los peatones: Verde (VP3), Rojo (RP3).

Semáforo 4: Verde (V4), Ámbar(A4), Rojo(R4) para los vehículos, para los peatones: Verde (VP4), Rojo (RP4).

|  |  |  |
| --- | --- | --- |
| Pin de pic 16F84A | Señal | Señal |
| 6 | V1 | V3 |
| 7 | A1 | A3 |
| 8 | R1 | R3 |
| 9 | V2 | V4 |
| 10 | A2 | A4 |
| 11 | R2 | R4E |
| 12 | RP1 | RP3 |
| 13 | RP2 | RP4 |
| 17 | VP1 | VP3 |
| 18 | VP2 | VP4 |

1. CODIFICACION DEL PROGRAMA

El siguiente código se escribió en MPlab para el pic 16F84A y posteriormente se compilo para generar un archivo .HEX el cual nos sirvió para hacer la simulación en Proteus y programar el pic.

\_\_CONFIG \_CP\_OFF & \_PWRTE\_OFF & \_WDT\_ON & \_XT\_OSC

list p=p16f84a, r=dec

#include <p16f84a.inc>

org 0

goto principal ;salto al prrograma principal al resetear

org 4

goto irq ;salto a la rutina de interrupcion

;definicion de constantes

cuenta equ 16 ;valor de cuenta del TMR0 (256-240)

RP2 equ 7 ;bits de los semaforos de peatones

RP1 equ 6

VP2 equ 1

VP1 equ 0

;declaracion de variables (posiciones de memoria que usan)

contador equ 0x10

retardo equ 0x11

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Programa principal

;

; Inicializacion del micro: registros de control, puertos e interrupciones.

; Programa y lanza el temporizador para que se active la interrupcion, y luego

; se queda en espera en un bucle infinito

;

principal:

;inicializacion del micro: programacion del registro option

;bit7. RPBU=1 resistencias de pull-up internas del puerto B deshabilitadas

;(no es necesario, ya que se deshabilitan al usarlo como salida)

;bit6. INTEDG=0. Da igual, ya que no usamos INT

;bit5. TOCS=0 Temporizador con reloj interno

;bit4. TOSE=0 Da igual

;bit3. PSA=0 Prescaler para el TMR0

;bits2-0 = 111 Ajuste del prescaler del TMR0 a 256

movlw b'10000111' ;palabra para OPTION hay que modificar bits 2-0

bsf STATUS, RP0 ;banco 1

movwf OPTION\_REG ;escribe la palabra en el registro OPTION

bcf STATUS, RP0 ;banco 0

;programacion de los puertos

;PuertoB=salidas => TRISB= todo a cero

;Estructura de PB: 7 6 5 4 3 2 1 0

; RP2 RP1 R2 A2 V2 R1 A1 V1

; --^-- --^--

; S2 S1

bsf STATUS, RP0 ;banco 1

clrf TRISB ;borra todos los bits, programando PB como salida

clrf TRISA ;PA tammbién salida

bcf STATUS, RP0 ;banco 0

;inicializacion de contador y puerto

clrf contador ;contador=0

movlw b'10011110' ;S2 rojo, S1 verde. Peatones, Rojo para S1

movwf PORTB ;lo saca por el puerto

bcf PORTA, VP2 ;peatones, Verde para S2

bsf PORTA, VP1

;programacion del temporizador

movlw cuenta

movwf TMR0 ;cuenta inicial del temporizador, lanza la cuenta.

;programacion de interrupciones:

;Bit7. GIE=1 para habilitar las interrupciones

;bit5. TOIE=1 para habilitar la interrupcion del temporizador

;resto a 0 para deshabilitar las demas interrupciones y borrar las banderas

movlw b'10100000'

movwf INTCON ;programa las interrupciones

espera: ;bucle infinito que borra continuamente el perro guardian

clrwdt ;el trabajo lo hace la irq. Solo si hay algun problema, se

goto espera ;reiniciara el micro

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Rutina de atencion a la interrupcion del temporizador

;

; La interrupcion se producira cada 0'05s.

; Contaremos 5 llamadas

; Se incrementara un contador de segundos para controlar la temporizacion del semaforo, y

; se modifica el estado de los semaforos, escribiendo en el puerto segun el tiempo pasado

;

; No se comprueba cual es la fuente de interrupcion, porque solo esta habilitada

; la interrupcion del temporizador.

irq:

;empezamos reprogramando el temporizador, para que vaya contando ya, y no introducir

;un retardo en la temporizacion debido al proceso de varias linea de codigo

movlw cuenta

movwf TMR0 ;reprogramacion del timer

incf retardo

movlw 5

subwf retardo, w ;compara retardo con 5

btfss STATUS, Z ;si no es 5

goto finswitch ;salta al final de la rutina de interrupción

clrf retardo

;este código se ejecutará cada 0'5s. El contador cuenta medios segundos

incf contador, f ;contador=contador+1

;implementacion de la sentencia switch-case

caso20: ;contador<20?

movlw 40

subwf contador, w ;compara contador con 40

btfsc STATUS, C ;¿acarreo?

goto fincaso20 ;no hay acarreo => contador>=40

;contador<40 (no hay acarreo)

movlw b'10011110' ;Coches: S2 rojo, S1 verde, peatones RP2 OFF, RP1 ON

movwf PORTB ;lo escribe en el puerto

bcf PORTA, VP2 ;peatones, Verde para S2

bsf PORTA, VP1

goto finswitch

fincaso20:

caso25: ;contador<25?

movlw 50

subwf contador, w ;compara si el contador es 50

btfsc STATUS, C ;oye hay ¿acarreo?

goto fincaso25 ;no hay acarreo => contador>=50

;contador<25 (no hay acarreo)

movlw b'10011110' ;Coches: RS2 , VS1 . Peatones: RP2 OFF, RP1 ON

movwf PORTB ;lo escribe en el puerto

btfsc contador, 0 ;si contador par (1 segundo)

goto finswitch

movlw b'10' ;mascara XOR para conmutar VP2 sin modificar el resto

xorwf PORTA, f ;conmuta VP2 (cada 1s)

goto finswitch

fincaso25:

caso30: ;25<=contador<30?

movlw 60

subwf contador, w ;compara contador con 60

btfsc STATUS, C ;¿acarreo?

goto fincaso30 ;no hay acarreo => contador>=60

;contador<30 (no hay acarreo)

movlw b'00000010' ;mascara XOR para conmutar S1.Ambar sin modificar el resto

xorwf PORTB, f ;conmuta S1.Ambar (cada 0'5s durante el tiempo entre 25 y 30)

bsf PORTB, 0 ;apaga S1.Verde

btfsc contador, 0 ;si contador par (cada segundo)

goto finswitch

movlw b'10' ;mascara para VP2

xorwf PORTA, f ;conmuta VP2

goto finswitch ;

fincaso30:

caso50: ;30<=contador<50?

movlw 100

subwf contador, w ;compara contador con 100

btfsc STATUS, C ;¿acarreo?

goto fincaso50 ;no hay acarreo => contador>=100

;contador<50 (no hay acarreo)

movlw b'01110011' ;Coches: S2 verde, S1 rojo. Peatones:RP2 ON, RP1 OFF

movwf PORTB ;lo escribe en el puerto

bsf PORTA, VP2 ;peatones, Verde para S1

bcf PORTA, VP1

goto finswitch ;

fincaso50:

caso55: ;30<=contador<55?

movlw 110

subwf contador, w ;compara contador con 110

btfsc STATUS, C ;¿acarreo?

goto fincaso55 ;no hay acarreo => contador>=110

;contador<55 (no hay acarreo)

movlw b'01110011' ;S2 verde, S1 rojo. Peatones:RP2 ON, RP1 OFF

movwf PORTB ;lo escribe en el puerto

btfss contador, 0 ;contador ¿par? => 1 segundo

goto finswitch

movlw b'01' ;mascara XOR para conmutar VP1 sin modificar el resto

xorwf PORTA, f ;conmuta VP1 (cada 1s)

goto finswitch ;

fincaso55:

caso60: ;55<=contador<60?

movlw 120

subwf contador, w ;compara contador con 120

btfsc STATUS, C ;¿acarreo?

goto fincaso60 ;no hay acarreo => contador>=120

;contador<60 (no hay acarreo)

movlw b'00010000' ;mascara XOR para conmutar S2.Ambar sin modificar el resto

btfss contador, 0 ;si contador par (cada segundo)

bsf W, VP1 ;activa la mascara también para VP1

xorwf PORTB, f ;conmuta AS2 (cada 0'5s durante el tiempo entre 55 y 60)

bsf PORTB, 3 ;apaga S2.Verde

btfsc contador, 0 ;si contador par (cada segundo)

goto finswitch

movlw b'01' ;mascara para VP1

xorwf PORTA, f ;conmuta VP1

goto finswitch ;

fincaso60:

otherwise: ;llego a 60

movlw b'10011110' ;Coches: RS2 , VS1. Peatones: RP2 OFF, RP1 ON

movwf PORTB ;lo escribe en el puerto

bcf PORTA, VP2 ;peatones, VS2

bsf PORTA, VP1

clrf contador ;empieza el ciclo de nuevo

finswitch: ;fin de la sentencia switch-case

movlw b'10100000'

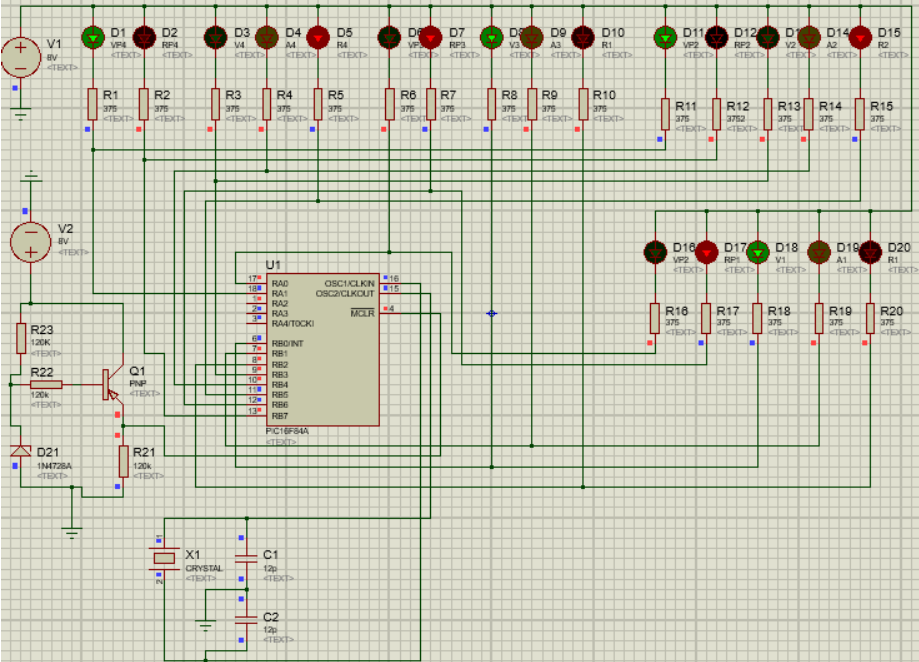
movwf INTCON

retfie ;retorno de interrupcion

end

1. SIMULACION EN PROTEUS

El Archivo .HEX generado lo tenemos que poner en nuestra simulación en proteus.



1. CONSTRUCCION DE SEMAFORO

Materiales:

* 8 leds rojos
* 8 leds verdes
* 4 leds ámbar
* Pic 16F84A programado
* 20 resistencias de 320 Ohms
* 1 protoboard
* 5m de cable rojo
* 5m de cable verde
* 3m de cable amarillo
* 6 popotes flexibles
* 1 caja de zapatos
* 1 cartulina negra
* 1 pliego de papel lustre negro
* Gises
* Silicon
* 1 cable usb
* 1 Cinta de aislar
* Papel aluminio
* 6 cajas pequeñas de chicles

Procedimiento:

1. Vamos a unir los leds con el cable y las resistencias, las resistencias deben ir en el pin mas largo del led, ese es el positivo



1. Para evitar que los cables se toquen los forramos con cinta de aislar
2. Repetir los pasos 1 y 2 con todos los leds
3. Pintaremos las cajas de chicles de color negro



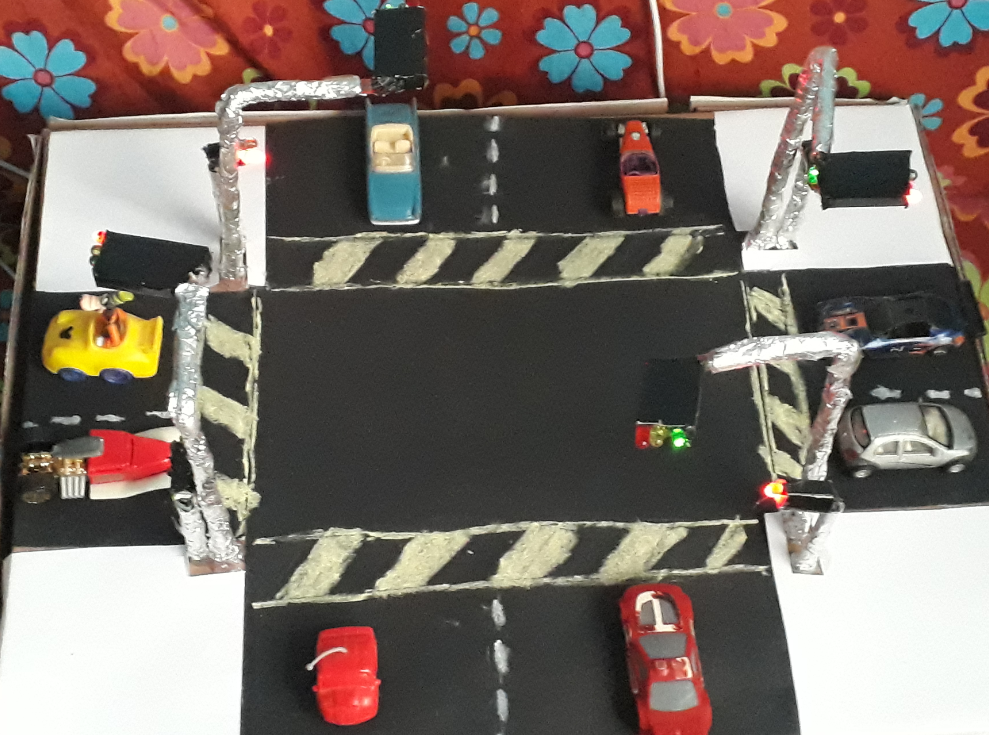
1. En los popotes meteremos los 3 leds de los semáforos de vehículos
2. Cubriremos los leds y parte del cable con las cajas de chicles
3. Un popote lo cortamos a la mitad y nos saldrán 2 mitades estos serán para los semáforos de los peatones
4. Hacemos lo mismo que hicimos para el semáforo de los vehículos
5. Una vez que tengamos todos los semáforos forraremos todos los popotes con papel aluminio



1. Ahora en la caja de zapatos marcaremos una carretera simulando una calle, la carretera la haremos con la cartulina negra, y dibujaremos los pasos peatonales con los gises.



1. Ahora haremos orificios en la caja para que pasen los cables de los semáforos, pegaremos los semáforos con silicón para que no se muevan
2. Colocamos el pic programado en la protoboard y esta la pegaremos adentro de la caja
3. Pelamos el cable USB y lo conectamos a la protoboard esto nos servirá como fuente de alimentación.
4. Una vez echo todo lo demás, solo queda conectar los cables al pic (los cables positivos), los negativos irán a tierra en la protoboard.
5. Para conectar los cables correctamente debes tenerlos bien identificados (puedes guiarte con la tabla de asignación de pines)
6. El resultado Final Seria este:



1. CONCLUCIÓN

Este proyecto fue de gran utilidad para poder comprender como es que funciona el Hardware, como a partir de materiales sencillos como un controlador (pic) y si es que si sabemos programarlo bien podemos hacer que el hardware haga lo que nosotros queramos.