

Microcontroladores

Semestre: 2022-1

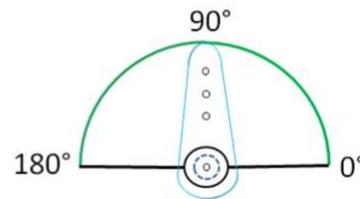
Profesor: Kalun José Lau Gan

Semana 11: Manejo de un hobby servo

1

Preguntas previas:

- Mi servo gira la vuelta completa. ¿Es normal?
 - No, los servos no giran la vuelta completa, tienen un movimiento de solo 180°



- ¿Cuándo es LB3?
 - Semana 12 según sílabo

2

El servo

- Elemento electromecánico realimentado (posee un lazo de control cerrado)
- Empleado comúnmente en hobby para radiocontrol de vehículos terrestres, aéreos y acuáticos, para el control de posición (ángulo)
 - Acelerador, freno, dirección vehicular, alerones (flaps), timón, robótica.
- Son clasificados por: tamaño, torque, velocidad, precisión, tamaño



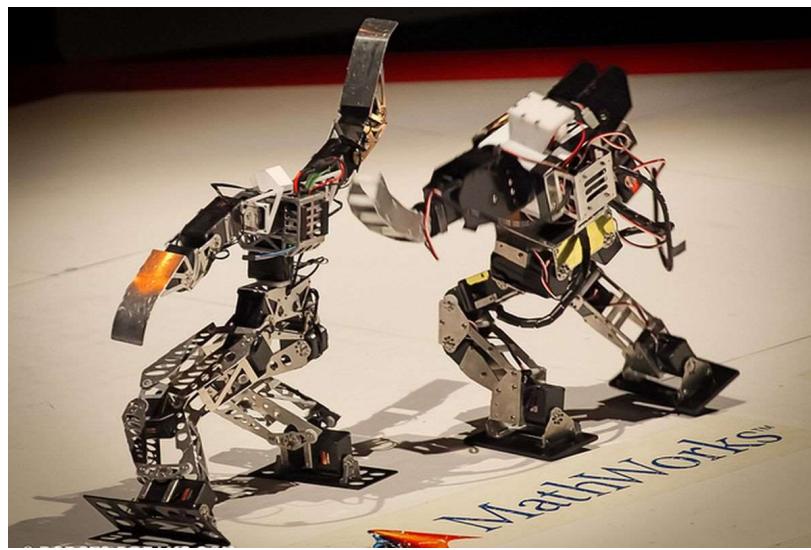
3

Servos en el hobby profesional



4

Servos en el hobby profesional



5



6

El servo

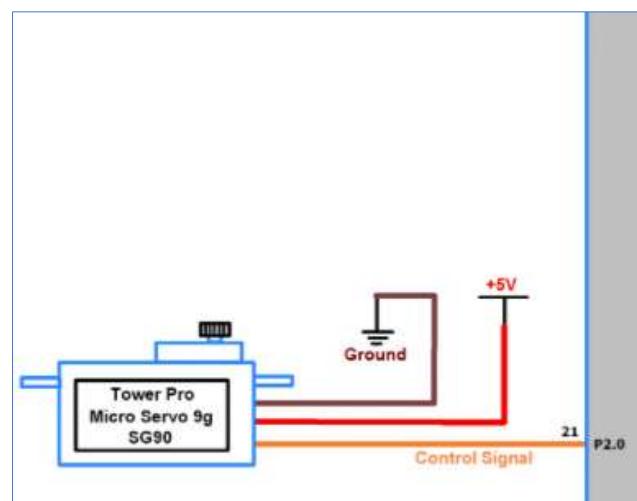
- Internamente posee un motor realimentado con un potenciómetro y mecanismo de reducción.



7

El servo

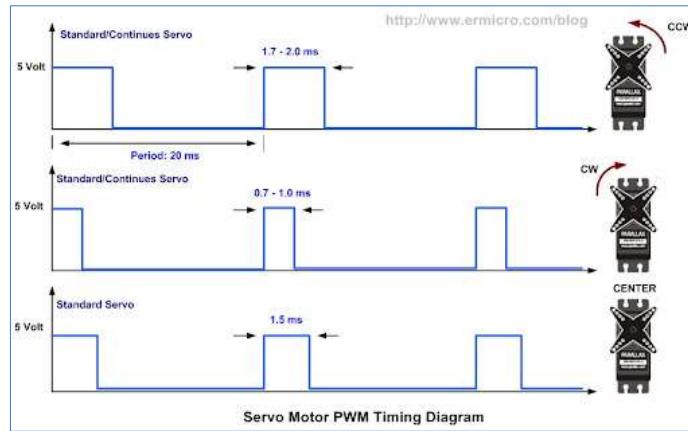
- Conexión con el microcontrolador:



8

El servo

- Modo de funcionamiento:
 - Tren de pulsos de periodo 20ms
 - El ancho del pulso (1.0ms – 2.0ms) positivo determinará la posición del eje del servo
 - Al quitarle el tren de pulsos el servo se inactivará



9

¿Se podrá el CCP modo PWM para generar la señal requerida por el servo?

- Valor de PR2 teniendo FOSC=48MHz:

$$\text{PWM Period} = [(PR2 + 1) \cdot 4 \cdot TOSC \cdot (\text{TMR2 Prescale Value})]$$

- ✗ PR2 = 14999 con PSC 1:16
- ✗ PR2 = 29999 con PSC 1:8
- ✗ PR2 = 59999 con PSC 1:4
- ✗ PR2 = 23999 con PSC 1:1

- No se puede generar lo solicitado!

10

¿Se podrá el CCP modo PWM para generar la señal requerida por el servo?

- Valor de PR2 teniendo FOSC=4MHz:

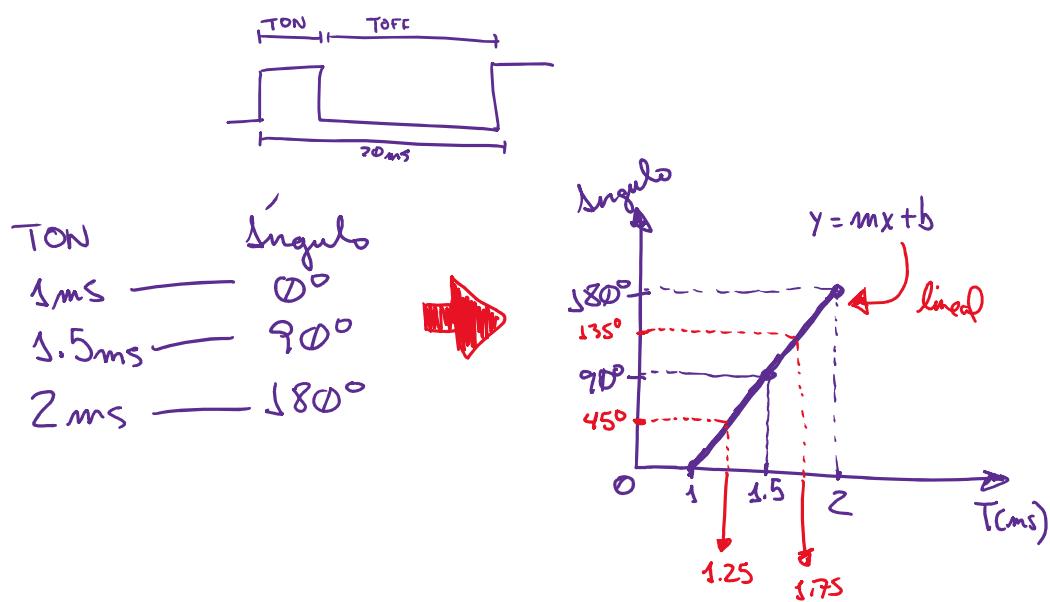
$$\text{PWM Period} = [(PR2 + 1) \cdot 4 \cdot Tosc] \\ (\text{TMR2 Prescale Value})$$

$\times PR2 = 5249$ con PSC 1:16
 $\times PR2 = \cancel{\cancel{}} \quad \text{con PSC 1:4}$
 $\checkmark PR2 = \cancel{\cancel{}} \quad \text{con PSC 1:1}$

- No se puede generar lo solicitado!
- Entonces. ¿Qué puedo usar?

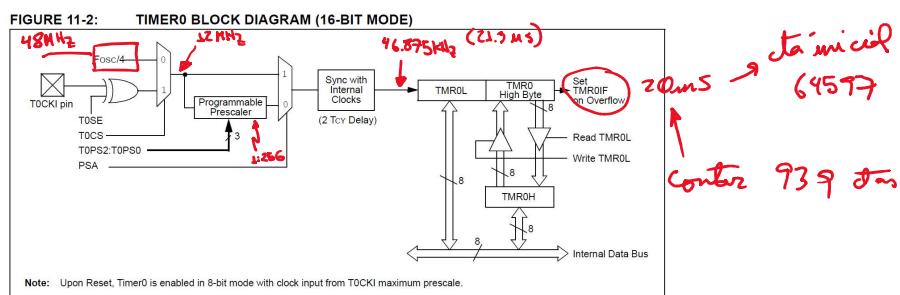
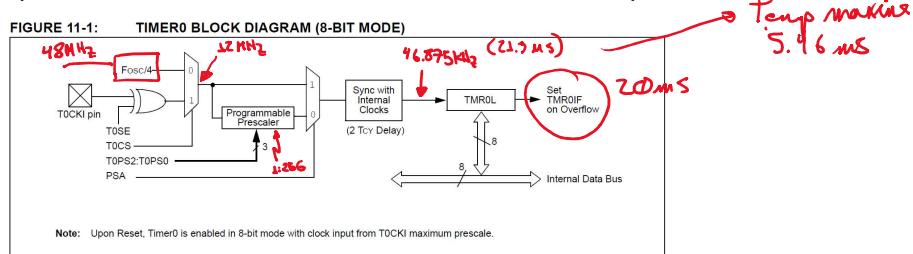
11

Relación TON vs ángulo del servo:



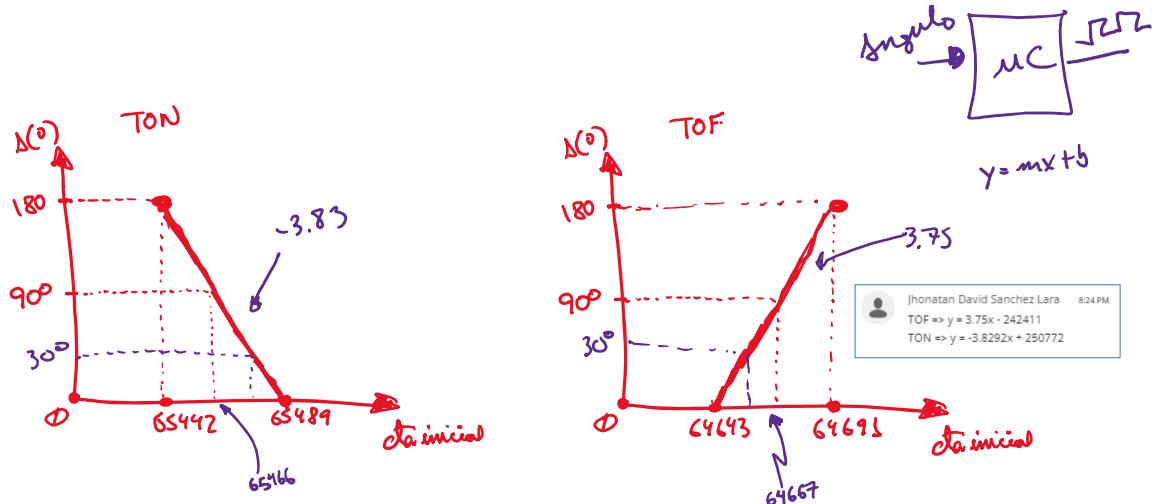
12

¿Puedo utilizar el Timer0 para generar la señal de 20ms hacia el servo (considerando Fosc = 48MHz)?



13

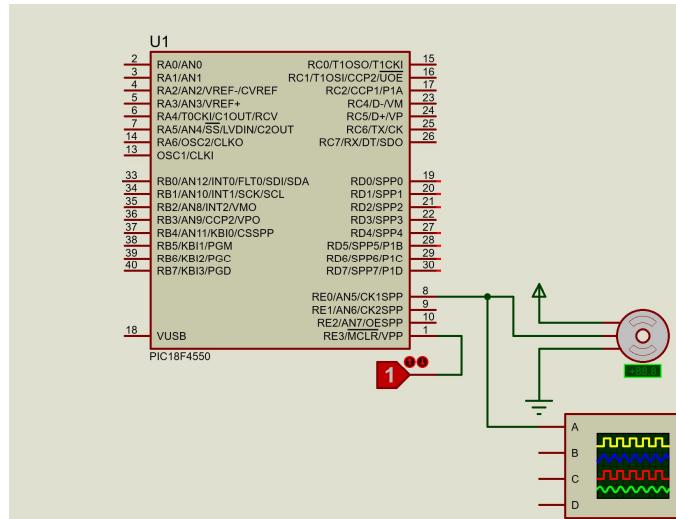
Gráficas de ángulo vs cta inicial de Timer0



14

Circuito de prueba

- En Proteus se usará el Motor – PWM Servo



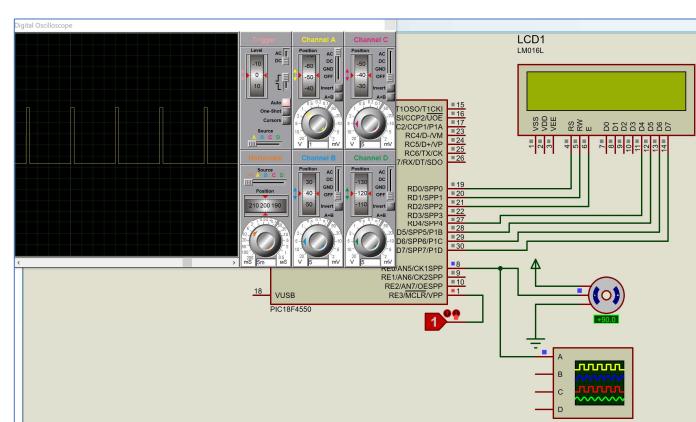
15

Prueba inicial: Servo a posición media (TON=1.5ms)

- Se empleará __delay_us() para la generación de TON y TOFF de la señal que va al servo

```

1 #include "cabecera.h"
2 #include <xc.h>
3 #define _XTAL_FREQ 48000000UL
4
5 void configuro(void) {
6     ADCON1 = 0x0F;           //Todos en digital
7     TRISEbits.RE0 = 0;       //RE0 como salida
8 }
9
10 void main(void) {
11     configuro();
12     while(1) {
13         LATEbits.LE0 = 1;
14         __delay_us(1500);
15         LATEbits.LE0 = 0;
16         __delay_us(18500);
17     }
18 }
```



16

Ejemplo: Movimiento del servo en diferentes posiciones y de manera secuencial automática

- Se está empleando funciones for para preservar el envío constante de la señal PWM hacia el servo

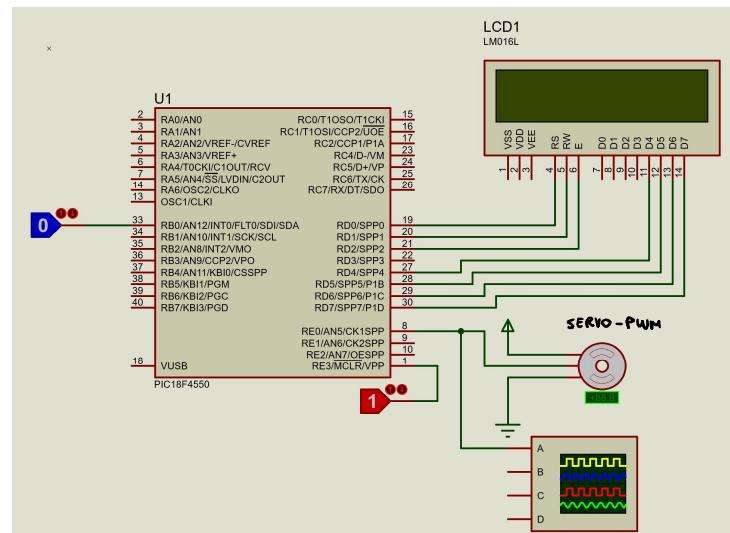
```

9 void main(void) {
10     configuarto();
11     while(1){
12         unsigned int x_var = 0;
13         for(x_var=0;x_var<50;x_var++){
14             LATBbits.LB7 = 1;
15             _delay_us(1000);
16             LATBbits.LB7 = 0;
17             _delay_us(19000);
18         }
19         for(x_var=0;x_var<50;x_var++){
20             LATBbits.LB7 = 1;
21             _delay_us(1500);
22             LATBbits.LB7 = 0;
23             _delay_us(18500);
24         }
25         for(x_var=0;x_var<50;x_var++){
26             LATBbits.LB7 = 1;
27             _delay_us(2000);
28             LATBbits.LB7 = 0;
29             _delay_us(18000);
30         }
31         for(x_var=0;x_var<50;x_var++){
32             LATBbits.LB7 = 1;
33             _delay_us(1500);
34             LATBbits.LB7 = 0;
35             _delay_us(18500);
36         }
37     }
38 }
```

17

Añadiendo el LCD al ejemplo:

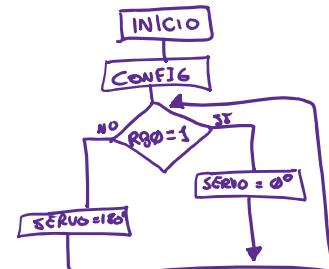
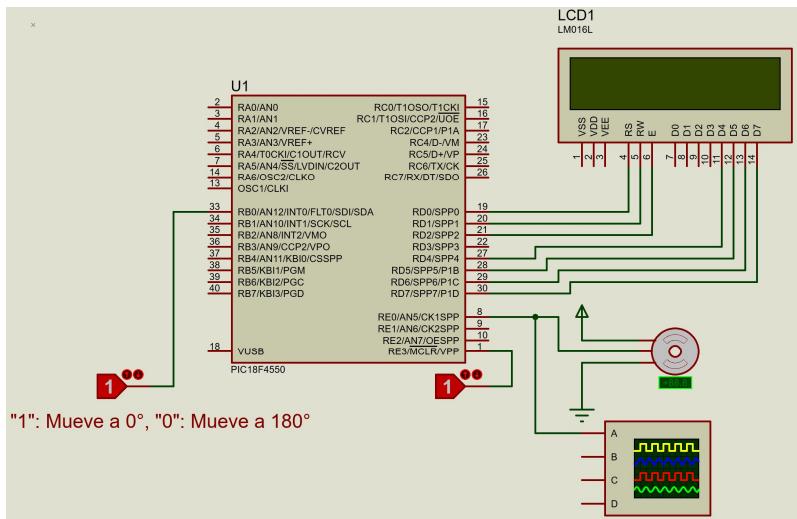
- Se empleará el LCD para visualizar el ángulo actual del servo.



18

Código ejemplo con LCD y entrada RBO

- RB0 = '1' : 0°, '0' : 180°



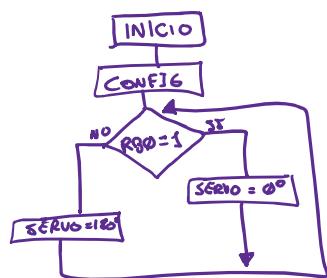
Servo 180°:
TON = 2ms

Servo 0°:
TON = 1ms

19

Código ejemplo con LCD y entrada RBO

- RB0 = '1' : 0°, '0' : 180°



Servo 180°:
TON = 2ms

Servo 0°:
TON = 1ms

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #define _XTAL_FREQ 48000000UL
4
5 void configuro(void){
6     TRISBbits.RB7 = 0;           //salida al servo
7 }
8
9 void main(void) {
10    configuro();
11    while(1){
12        if(PORTBbits.RB0 == 1){
13            LATBbits.LB7 = 1;
14            __delay_us(2000);
15            LATBbits.LB7 = 0;
16            __delay_us(18000);
17        }
18        else{
19            LATBbits.LB7 = 1;
20            __delay_us(1000);
21            LATBbits.LB7 = 0;
22            __delay_us(19000);
23        }
24    }
25 }
```

20

Código ejemplo con LCD y entrada RBO

- La entrada RBO servirá para seleccionar entre el ángulo 45° y 135° del eje del servo.

```

1 #include "cabecera.h"
2 #include "LCD.h"
3 #include <xc.h>
4 #define _XTAL_FREQ 48000000UL
5
6 void configuracion(void){
7     ADCON1 = 0x0F; //Puertos
8     TRISEbits.RB0 = 0; //RB0 com
9 }
10
11 void lcd_init(void){
12     TRISD = 0x00;
13     __delay_ms(15);
14     LCD_CONFIG();
15     __delay_ms(15);
16     BORRAR_LCD();
17     CURSOR_HOME();
18     CURSOR_ONOFF(OFF);
19 }
```

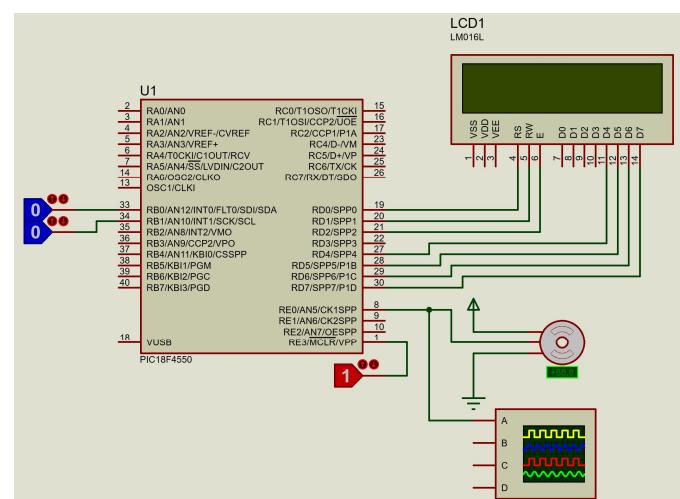
```

21 void main(void) {
22     configuracion();
23     lcd_init();
24     POS_CURSOR(1,0);
25     ESCRIBE_MENSAJE("Servomecanismo",14);
26
27     while(1) {
28         if(PORTBbits.RB0 == 1) {
29             LATEbits.LE0 = 1;
30             __delay_us(1750);
31             LATEbits.LE0 = 0;
32             __delay_us(18250);
33             POS_CURSOR(2,0);
34             ESCRIBE_MENSAJE("Angulo:135",10);
35             ENVIA_CHAR(0xDF);
36         }
37         else{
38             LATEbits.LE0 = 1;
39             __delay_us(1250);
40             LATEbits.LE0 = 0;
41             __delay_us(18750);
42             POS_CURSOR(2,0);
43             ESCRIBE_MENSAJE("Angulo: 45",10);
44             ENVIA_CHAR(0xDF);
45         }
46     }
47 }
```

21

Ejemplo: Dos entradas para seleccionar 4 ángulos en el servo

- Se empleará la sentencia switch-case en el programa en XC8 para la selección del ángulo que va a tener el eje del servo.



22

Ejemplo: Dos entradas para seleccionar 4 ángulos en el servo

```

1  #include "cabecera.h"
2  #include <xc.h>
3  #define _XTAL_FREQ 48000000UL
4
5  void configuro(void){
6      ADCON1 = 0x0F; //Todos en digital
7      TRISEbits.RE0 = 0; //RE0 como salida
8 }

```

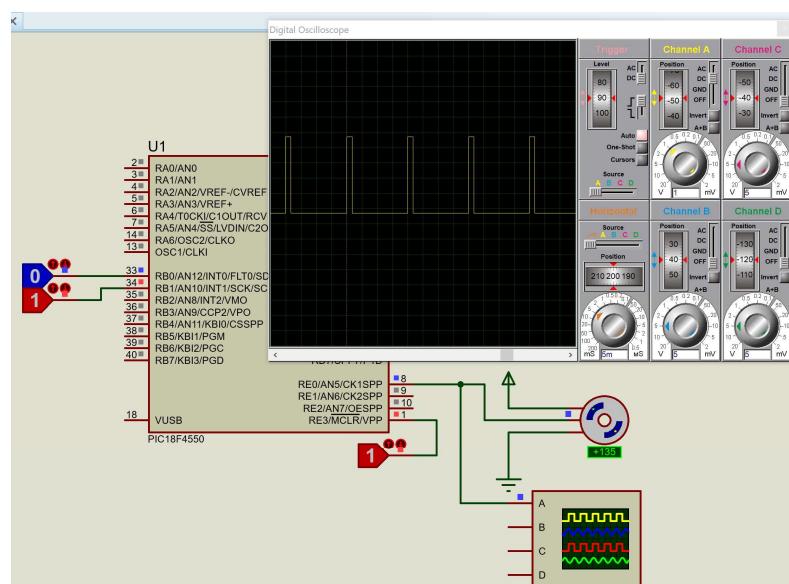
```

10 void main(void) {
11     unsigned char entrada = 0;
12     configuro();
13     while(1){
14         entrada = PORTB & 0x03;
15         switch(entrada){
16             case 0:
17                 LATEbits.LE0 = 1;
18                 __delay_us(1000);
19                 LATEbits.LE0 = 0;
20                 __delay_us(19000);
21                 break;
22             case 1:
23                 LATEbits.LE0 = 1;
24                 __delay_us(1250);
25                 LATEbits.LE0 = 0;
26                 __delay_us(18750);
27                 break;
28             case 2:
29                 LATEbits.LE0 = 1;
30                 __delay_us(1750);
31                 LATEbits.LE0 = 0;
32                 __delay_us(18250);
33                 break;
34             case 3:
35                 LATEbits.LE0 = 1;
36                 __delay_us(2000);
37                 LATEbits.LE0 = 0;
38                 __delay_us(18000);
39                 break;
40             default:
41                 break;
42         }
43     }
44 }

```

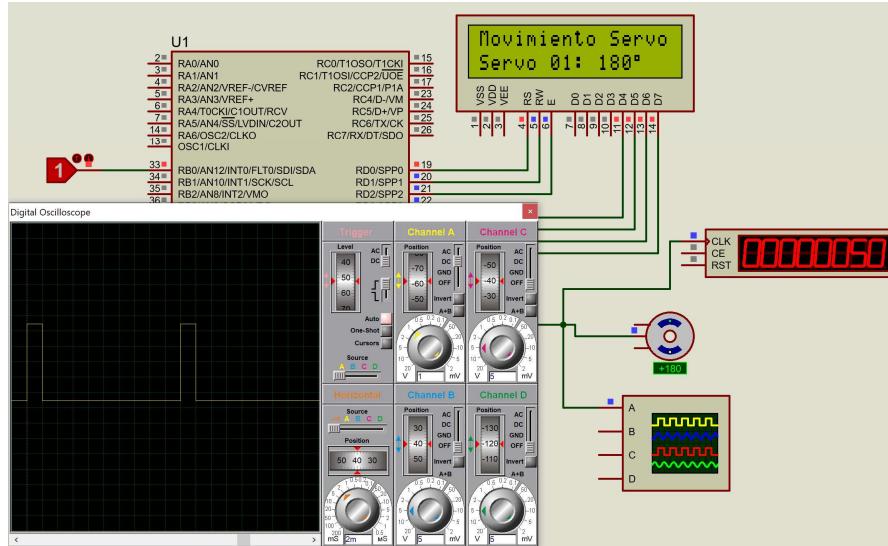
23

Simulación



24

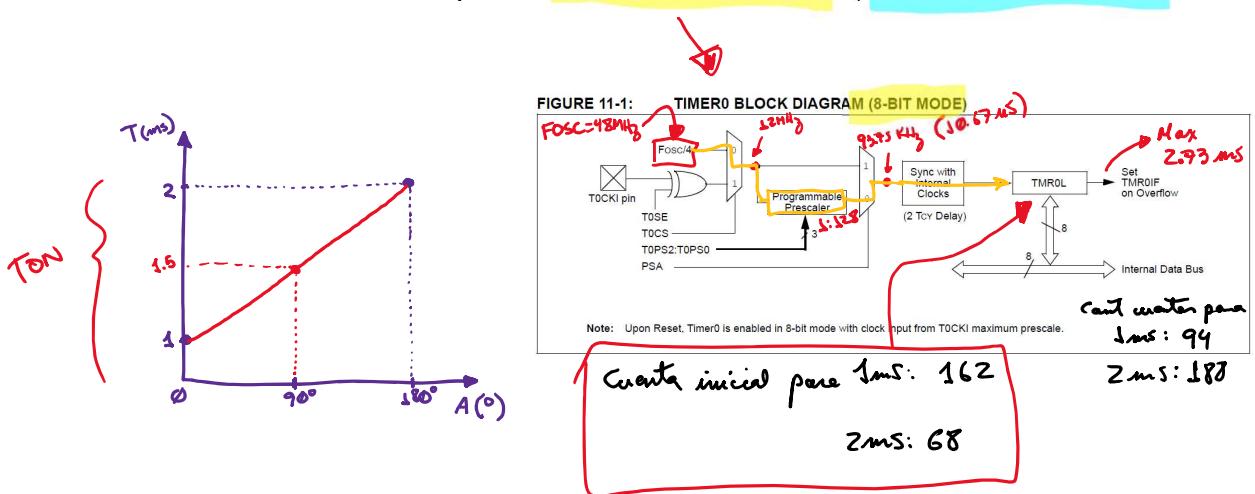
Simulación del movimiento secuencial y visualización de Mensajes en el LCD



25

Trabajando con el Timer0 para manipular el servo

- El Timer0 debe de temporizar 1 á 2ms en el TON y 18 á 19ms en TOFF



26

Configuración de Timer0 (T0CON) - TON

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER							
	R/W-1 TMR0ON	R/W-1 TOPS1T	R/W-1 TOCS	R/W-1 TSE	R/W-1 PSA	R/W-1 TOPS2	R/W-1 TOPS1
bit 7							
Legend:							
R = Readable bit -n = Value at POR	W = Writable bit '1' = Bit is set	U = Unimplemented bit, read as '0' '0' = Bit is cleared	X = Bit is unknown				
bit 7	TMR0ON: Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0	T08BIT: Timer0 8-Bit/16-Bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter	TOCS: Timer0 Clock Source Select bit 1 = Transition on TOCKI pin 0 = Internal instruction cycle clock (CLKO)	TSE: Timer0 Source Edge Select bit 1 = Increment on high-to-low transition on TOCKI pin 0 = Increment low-to-high transition on TOCKI pin	PSA: Timer0 Prescaler Assignment bit 1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.	TOPS2:TOPS0: Timer0 Prescaler Select bits 111 = 1:266 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value	TOPS1: Timer0 Prescaler Select bits 111 = 1:266 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value
bit 6							
bit 5							
bit 4							
bit 3							
bit 2-0							

TOCON: 0XC6

27

Código en XC8

- Empleando Timer0 para temporizar TON
- TOFF se mantiene con `__delay_us()`, pendiente de cambio para ser usado con Timer0 también.

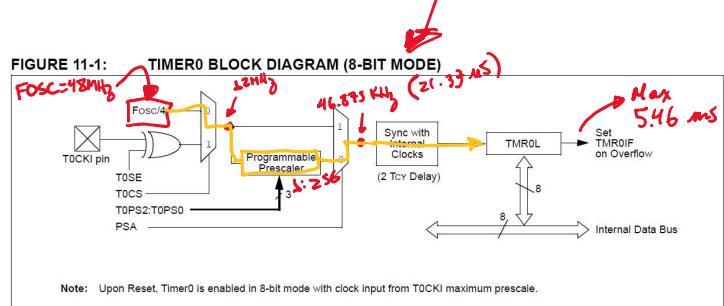
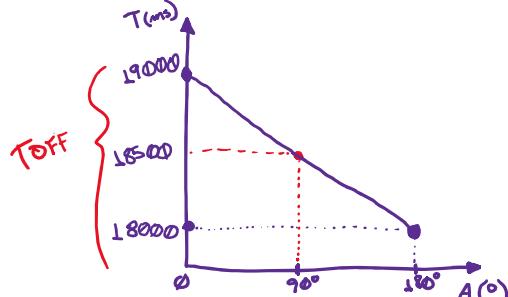
```

1  #include "cabecera.h"
2  #include <xc.h>
3  #define _XTAL_FREQ 48000000UL
4
5  void configuracion(void){
6      ADCON1 = 0x0F;           //Todos los I/O como digitales
7      TRISEbits.RE0 = 0;       //RE0 como salida
8      TOCON = 0xC6;           //TMR0 ON, PSC 1:128, FOSC/4
9  }
10
11 void main(void) {
12     configuracion();
13     while(1){
14         if(PORTBbits.RB0 == 1){
15             LATEbits.LE0 = 1;
16             TMROL = 162;
17             while(INTCONbits.TMR0IF == 0);
18             LATEbits.LE0 = 0;
19             __delay_ms(20);
20             INTCONbits.TMR0IF = 0;
21         }
22         else{
23             LATEbits.LE0 = 1;
24             TMROL = 68;
25             while(INTCONbits.TMR0IF == 0);
26             LATEbits.LE0 = 0;
27             __delay_ms(20);
28         }
29     }
30 }
31 }
```

28

Trabajando con el Timer0 para manipular el servo

- El Timer0 debe de temporizar 1 á 2ms en el TON y 18 á 19ms en TOFF

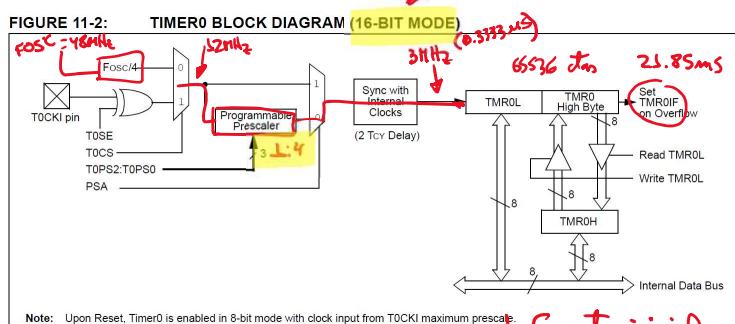
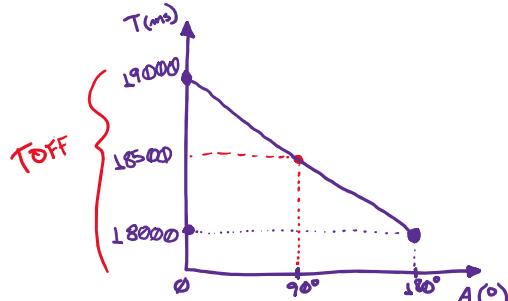


NOTA: en modo 8 bit y FOSC 48MHz obtenemos la temporización máxima de 5.46ms y no alcanza los 20ms requeridos para hacer el TOFF

29

Trabajando con el Timer0 para manipular el servo

- El Timer0 debe de temporizar 1 á 2ms en el TON y 18 á 19ms en TOFF



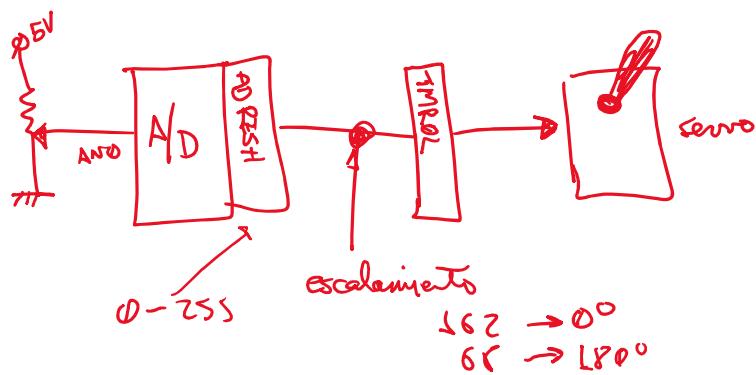
Para 18ms : 54005 dts
Para 19ms : 57000 dts

Cuenta inicial
11531
8526

30

Propuesta: Enlace A/D con Timer0 para controlar la posición del servo a través de un potenciómetro en AN-0

- Tener en cuenta la resolución tanto del A/D como del Timer0, asimismo el proceso de escalamiento entre el valor obtenido de AN.0 a ser trasladado a la cuenta inicial de Timer0

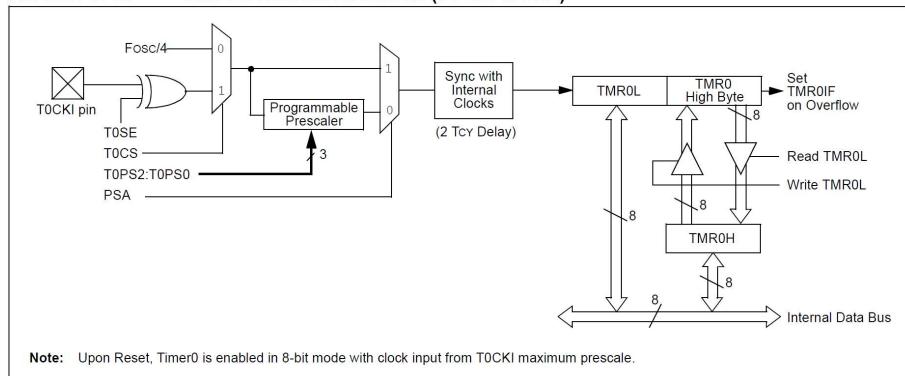


31

Usando el Timer0 en modo 16 bits para la temporización:

- El timer0 debe de generar tanto TON como TOF de la señal que va hacia el servo

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



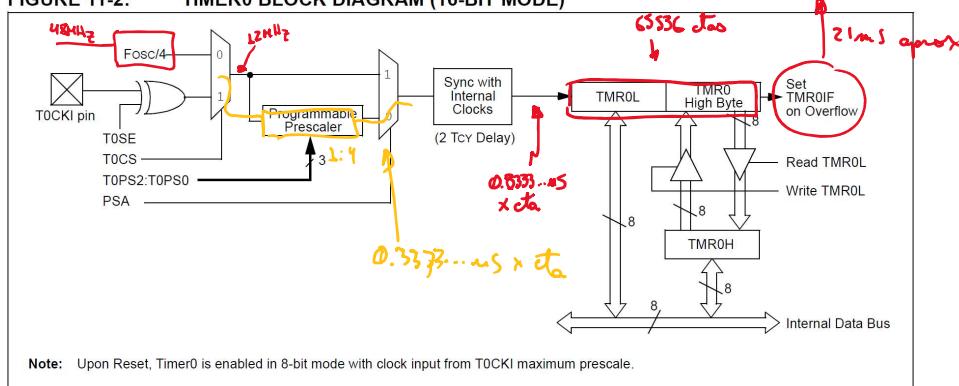
32

Usando el Timer0 en modo 16 bits para la temporización:

- El timer0 debe de generar tanto TON como TOF de la señal que va hacia el servo
- Configuración para 2ms

$\text{Psc 1:4} \left\{ \begin{array}{l} \text{Para: } 1 \mu\text{s} \rightarrow 3000 \text{ ms} \\ \text{Para: } 2 \mu\text{s} \rightarrow 6000 \text{ ms} \end{array} \right\} \text{TON} \quad \begin{array}{l} 62536 \\ 59536 \\ 11536 \\ 8836 \end{array}$
 $\text{Psc 1:4} \left\{ \begin{array}{l} \text{Para: } 18 \mu\text{s} \rightarrow 5400 \text{ ms} \\ \text{Para: } 17 \mu\text{s} \rightarrow 5100 \text{ ms} \end{array} \right\} \text{TOFF} \quad \begin{array}{l} 62536 \\ 59536 \\ 11536 \\ 8836 \end{array}$

FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



33

Usando el Timer0 en modo 16 bits para la temporización:

- El timer0 debe de generar tanto TON como TOF de la señal que va hacia el servo

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W	R/W-1
TMR0ON	0	0	X	0	0	0	0

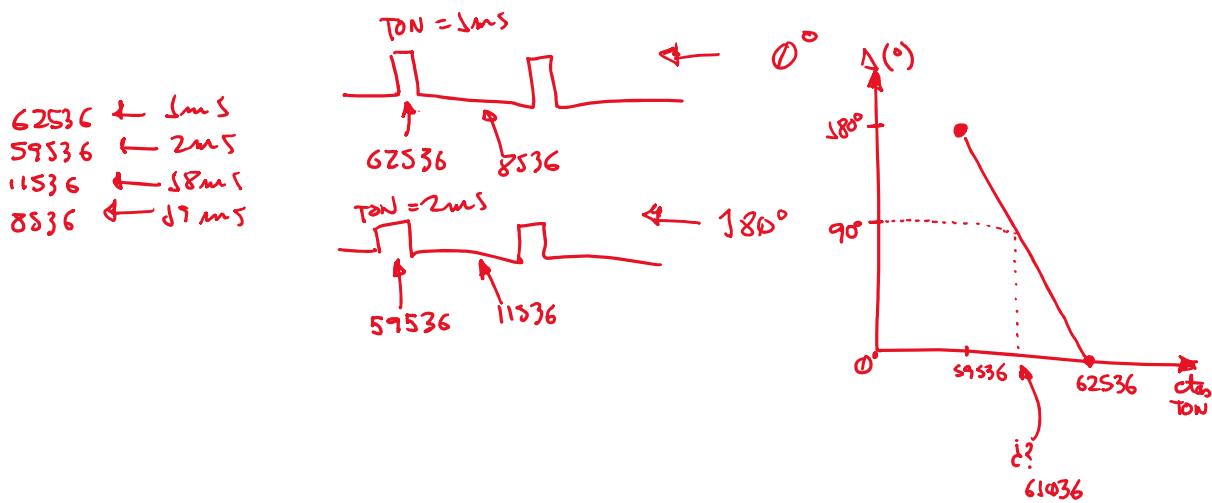
Legend:
R = Readable bit
W = Writable bit
-n = Value at POR
'1' = Bit is set
U = Unimplemented bit, read as '0'
'0' = Bit is cleared
x = Bit is unknown

bit 7	TMR0ON: Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0						
bit 6	T08BIT: Timer0 8-Bit/16-Bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter						
bit 5	TOCS: Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKO)						
bit 4	T0SE: Timer0 Source Edge Selected bit 1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin						
bit 3	PSA: Timer0 Prescaler Assignment bit 1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.						
bit 2-0	TOPS2:TOPS0: Timer0 Prescaler Select bits 111 = 1:256 Prescale value 110 = 1:128 Prescale value 101 = 1:64 Prescale value 100 = 1:32 Prescale value 011 = 1:16 Prescale value 010 = 1:8 Prescale value 001 = 1:4 Prescale value 000 = 1:2 Prescale value						

$\text{t0con} = 0x81$

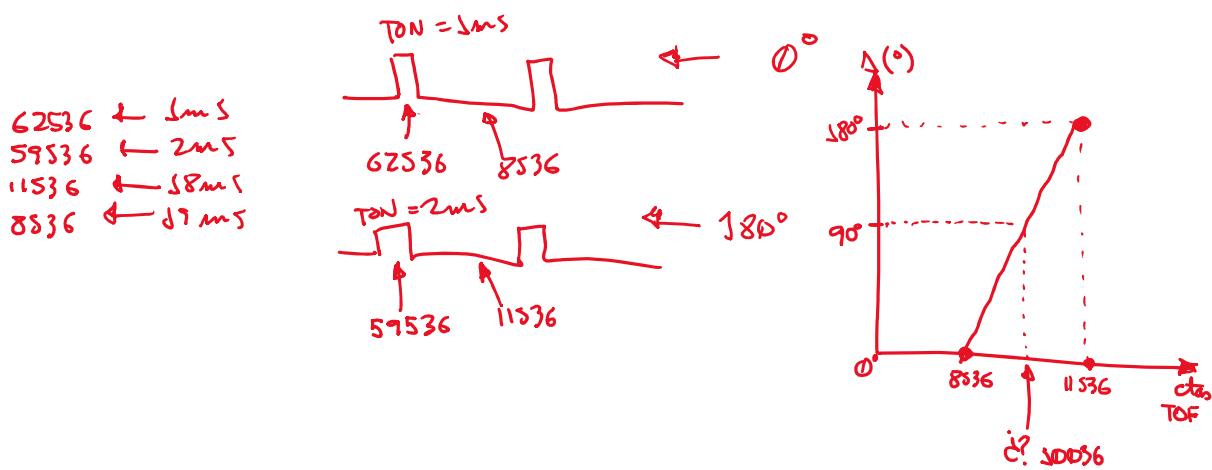
34

Usando el Timer0 en modo 16 bits para la temporización:



35

Usando el Timer0 en modo 16 bits para la temporización:



36

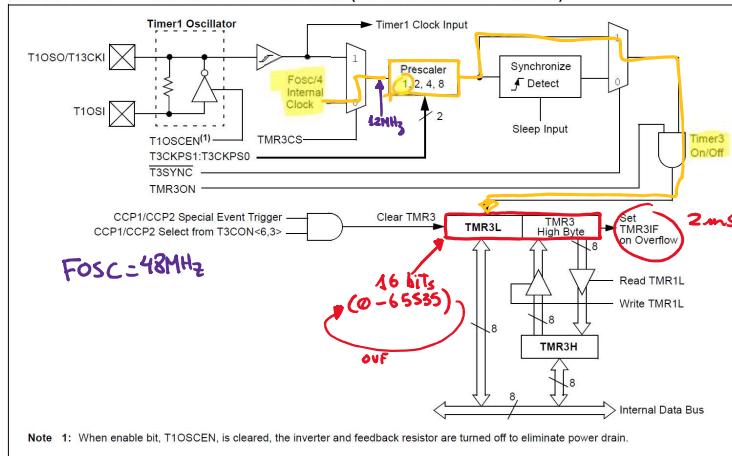
Empleando el Timer 3 para manipular el servo 0° y 180°

- Periodo de temporización:

- TON entre 1ms y 2ms ✓
- TOFF entre 18ms y 19ms ↗ pendiente

Temporización Máxima:
5.46 ms

FIGURE 14-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)



Para 2ms:

$$5.46 \text{ ms} - 65536 \\ 2 \text{ ms} - x \\ x = \frac{65536}{5.46 \text{ ms}} \\ x = 24000$$

la cantidad de
cuentos que TMR3
debe de realizar

⇒ Cuenta inicial:
41536
TMR3H:TMR3L
para 2ms.

37

Empleando el Timer 3 para manipular el servo

- Periodo de temporización:

- TON entre 1ms y 2ms ✓
- TOFF entre 18ms y 19ms ↗ pendiente

Para 1ms:

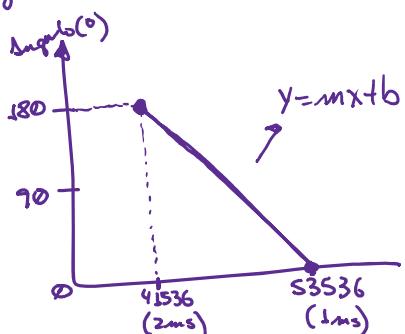
$$x = \frac{65536}{5.46 \text{ ms}} \\ x = 12000$$

⇒ cuenta inicial:

$$\frac{65536 - 12000}{53536}$$

TMR3H:TMR3L para 1ms

Ángulo vs cuenta inicial TMR3



38

Configuración de T3CON

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
bit 7	T3CKP2	T3CKP1	T3CKPS0	T3CKPS1	T3SYNC	TMR3CS	TMR3ON	bit 0
0	0	0	0	0	0	0	0	0

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7	RD16: 16-Bit Read/Write Mode Enable bit 1 = Enables register read/write of Timer3 in one 16-bit operation 0 = Enables register read/write of Timer3 in two 8-bit operations
bit 6, 3	T3CCP2:T3CCP1: Timer3 and Timer1 to CCPx Enable bits 1x = Timer3 is the capture/compare clock source for both CCP modules 01 = Timer3 is the capture/compare clock source for CCP2; Timer1 is the capture/compare clock source for CCP1 00 = Timer1 is the capture/compare clock source for both CCP modules
bit 5-4	T3CKPS1:T3CKPS0: Timer3 Input Clock Prescale Select bits 11 = 1:8 Prescale value 10 = 1:4 Prescale value 01 = 1:2 Prescale value 00 = 1:1 Prescale value
bit 2	T3SYNC: Timer3 External Clock Input Synchronization Control bit (Not usable if the device clock comes from Timer1/Timer3.) When TMR3CS = 1: 1 = Do not synchronize external clock input 0 = Synchronize external clock input When TMR3CS = 0: This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.
bit 1	TMR3CS: Timer3 Clock Source Select bit 1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge) 0 = Internal clock ($F_{osc}/4$)
bit 0	TMR3ON: Timer3 On bit 1 = Enables Timer3 0 = Stops Timer3

T3CON = 0x01

39

Código en XC8:

- Se reemplazó el `__delay_us()` de TON con Timer3
- Pendiente: hacer el mismo procedimiento para TOFF

```

1 #include "cabecera.h"
2 #include "LCD.h"
3 #include <xc.h>
4 #define _XTAL_FREQ 48000000UL
5
6 void configuracion(void){
7     ADCON1 = 0xF;           //Puertos E/S como digitales
8     TRISEbits.RE0 = 0;      //RE0 como salida
9     T3CON = 0x01;          //Configuracion Timer3
10 }
11
12 void lcd_init(void){
13     TRISD = 0x00;
14     __delay_ms(15);
15     LCD_CONFIG();
16     __delay_ms(15);
17     BORRAR_LCD();
18     CURSOR_HOME();
19     CURSOR_ONOFF(OFF);
20 }
```

```

22 void main(void) {
23     configuracion();
24     lcd_init();
25     POS_CURSOR(1,0);
26     ESCRIBE_MENSAJE("Servomecanismo",14);
27     while(1) {
28         if(PORTBbits.RB0 == 1){
29             LATEbits.LE0 = 1;
30             TMR3H = 0xA2;
31             TMR3L = 0x40;           //Angulo 180° Cuenta inicial 41536
32             while(PIR2bits.TMR3IF == 0);
33             LATEbits.LE0 = 0;
34             __delay_us(18250);
35             POS_CURSOR(2,0);
36             ESCRIBE_MENSAJE("Angulo:180",10);
37             ENVIA_CHAR(0xDF);
38             PIR2bits.TMR3IF = 0;
39         }
40     }
41     else{
42         LATEbits.LE0 = 1;
43         TMR3H = 0xD1;
44         TMR3L = 0x20;           //Angulo 0° Cuenta inicial 53536
45         while(PIR2bits.TMR3IF == 0);
46         LATEbits.LE0 = 0;
47         __delay_us(18750);
48         POS_CURSOR(2,0);
49         ESCRIBE_MENSAJE("Angulo: 0",10);
50         ENVIA_CHAR(0xDF);
51         PIR2bits.TMR3IF = 0;
52     }
53 }
```

40

Cuestionario:

- Ya que se ha analizado el Timer0 y el Timer3 como fuente de tiempo para obtener los periodos de un servo. Es posible manipular dos servos, uno con el Timer0 y otro con el Timer3 junto con el manejo adecuado de las interrupciones.

41

Fin de la sesión

- Destinar una hora el sábado y una hora el domingo para repasar el curso.

42