

# Microcontroladores

## Laboratorio Sesión 11

Semestre: 2021-2

Profesor: Kalun José Lau Gan

1

### Preguntas previas:

- Tengo unos inconvenientes de que no me arroja la onda para mover el servomecanismo empleando el Timer0.
  - Se ha constatado que en lugar de bajar la bandera TMROIF luego de producirse un desborde se la esta alzando. La bandera no debe de colocarse manualmente a uno, es el mismo modulo quien lo hace ante un desborde, lo que se debe de hacer es bajar la bandera manualmente para que pueda detectarse otro evento de desborde en el modulo.
- He conseguido una PSU de 5V 2A, funcionará con los circuitos que implemente? No se quemará ya que solicitó 5V 1A?
  - El amperaje mostrado en las especificaciones técnicas de la PSU es la corriente máxima que puede entregar, va a depender del consumo que tenga tu circuito, es decir, si tu circuito consume solo 100mA, la PSU solo entregará dicho monto de corriente.

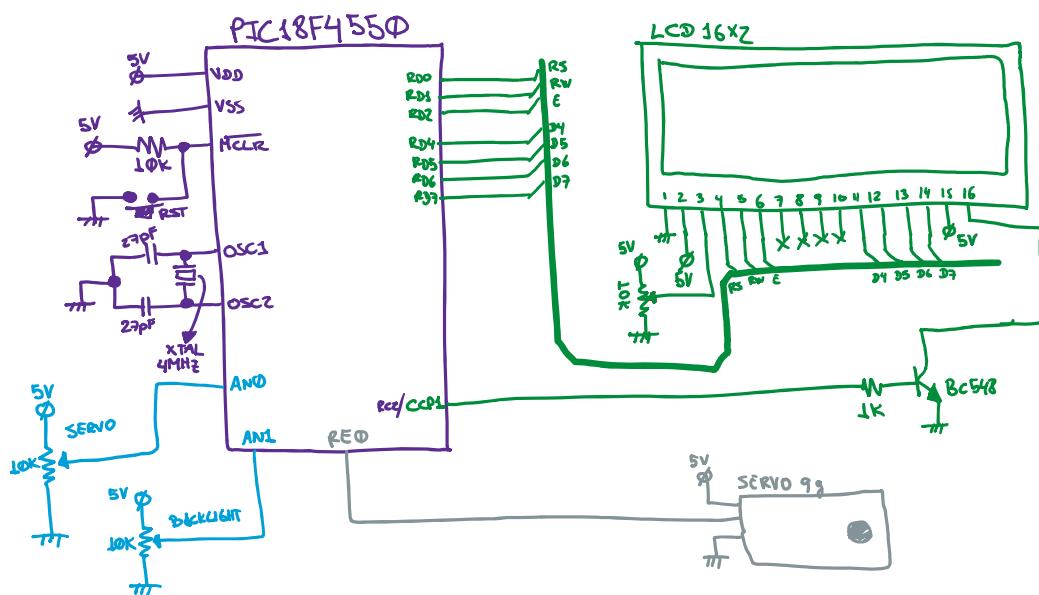
2

## Agenda:

- Interface a servomecanismos
- Generación de PWM con CCPx
- Interacción entre el conversor A/D con diferentes periféricos internos del microcontrolador.
- Ejemplos de aplicación:
  - Ajuste de la intensidad de la luz de fondo del LCD a través de un potenciómetro
  - Movimiento del servo a través de un potenciómetro

3

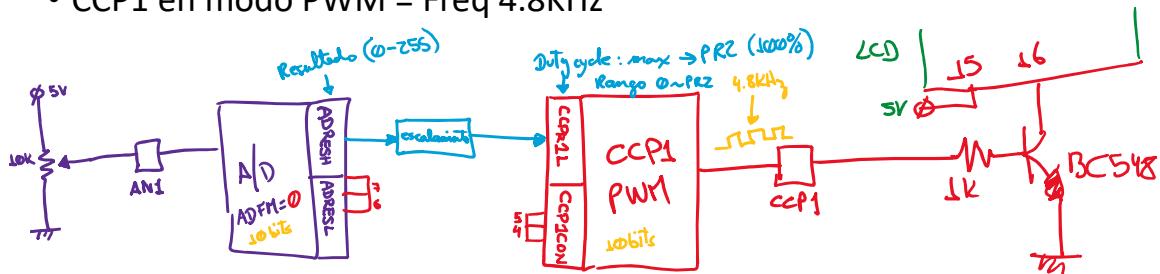
Implementar el siguiente circuito



4

## Análisis:

- Atacar el problema por partes:
  - Conversor A/D con el control de luz de fondo (AN.1 con CCP1-PWM)
  - Conversor A/D con el servo (AN.0 con cuenta inicial de Timer0 ó Timer3 para temporizar TON y TOF de la señal)
- Habilitar dos canales analógicos en el A/D:
  - ADCON1
- Conversor A/D va a funcionar con justo izquierda (ADFM=0)
- CCP1 en modo PWM = Freq 4.8KHz



5

## Análisis

- La configuración de CCP1 para PWM de F 4.8Khz se trató en la sesión anterior de laboratorio:

### Ejemplo con simulación:

- Generar una onda PWM empleando el CCP1 con frecuencia de 4.8Khz y 50% de Duty Cycle (FOSC = 48MHz):

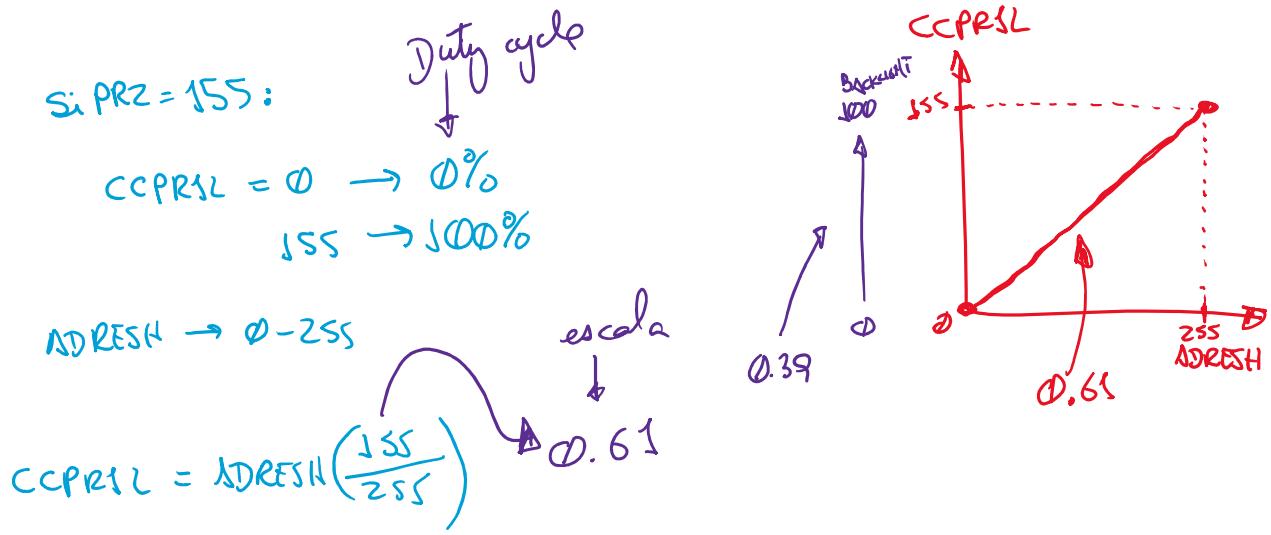
$$\text{PWM Period} = [(PR2 + 1) \cdot 4 \cdot Tosc \cdot (TMR2 Prescale Value)]$$

$PR2 = 155$  con PSC 1:16 } Conf periodo  
 $CCP1RL = 78$  ← Conf. duty cycle

6

## Consideración:

- Proceso de escalamiento: CCPR1L no debe exceder PR2



7

## Análisis:

- Configuración del A/D:
  - Dos canales: AN.0 y AN.1

$$ADCON1 = 0 \times 01$$

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

| bit 7   | U-0   | U-0  | R/W <sup>(1)</sup> | R/W <sup>(0)</sup> | R/W <sup>(1)</sup> | bit 0 |
|---|---|------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|-------|
| bit 7   | —   | —    | VCFG1              | VCFG0              | PCFG3              | PCFG2              | PCFG1              | PCFG0              | PCFG3              | PCFG2              | PCFG1 |
| <b>Legend:</b>  |   |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
| R = Readable bit<br>W = Writable bit<br>U = Unimplemented bit, read as '0'<br>-n = Value at POR<br>'1' = Bit is set<br>'0' = Bit is cleared<br>x = Bit is unknown |   |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
| bit 7-6   | Unimplemented: Read as '0'  |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
| bit 5   | VCFG1: Voltage Reference Configuration bit (VREF- source)<br>1 = VREF- (AN2)<br>0 = VSS |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
| bit 4   | VCFG0: Voltage Reference Configuration bit (VREF+ source)<br>1 = VREF+ (AN3)<br>0 = VDD |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
| bit 3-0   | PCFG3:PCFG0: A/D Port Configuration Control bits:                                       |      |                    |                    |                    |                    |                    |                    |                    |                    |       |
|   | PCFG3:<br>PCFG0   | AN12 | AN11               | AN10               | AN9                | AN8                | AN7 <sup>(2)</sup> | AN6 <sup>(2)</sup> | AN5 <sup>(2)</sup> | AN4                | AN3   |
| 0000 <sup>(1)</sup>   | A   | A    | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0001  | A   | A    | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0010  | A   | A    | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0011  | D   | A    | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0100  | D   | D    | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0101  | D   | D    | D                  | A                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0110  | D   | D    | D                  | D                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 0111 <sup>(1)</sup>   | D   | D    | D                  | D                  | A                  | A                  | A                  | A                  | A                  | A                  | A     |
| 1000  | D   | D    | D                  | D                  | D                  | A                  | A                  | A                  | A                  | A                  | A     |
| 1001  | D   | D    | D                  | D                  | D                  | D                  | A                  | A                  | A                  | A                  | A     |
| 1010  | D   | D    | D                  | D                  | D                  | D                  | D                  | A                  | A                  | A                  | A     |
| 1011  | D   | D    | D                  | D                  | D                  | D                  | D                  | D                  | A                  | A                  | A     |
| 1100  | D   | D    | D                  | D                  | D                  | D                  | D                  | D                  | D                  | A                  | A     |
| 1101 <sup>(1)</sup>   | D   | D    | D                  | D                  | D                  | D                  | D                  | D                  | D                  | A                  | A     |
| 1110  | D   | D    | D                  | D                  | D                  | D                  | D                  | D                  | D                  | D                  | A     |
| 1111  | D   | D    | D                  | D                  | D                  | D                  | D                  | D                  | D                  | D                  | D     |

A = Analog input

D = Digital I/O

8

## Código parte backlight

```

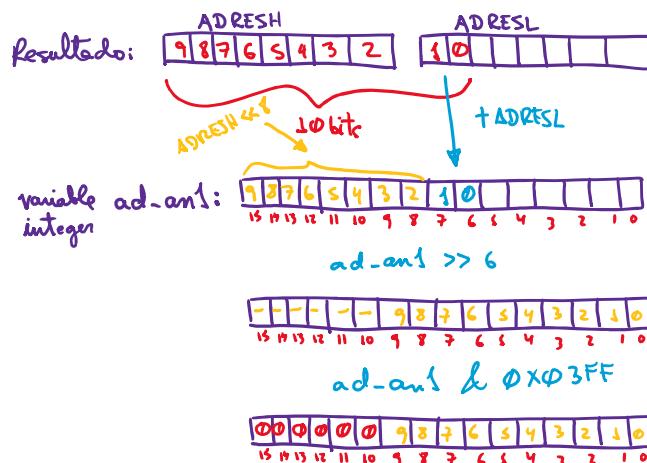
6 //declaracion de variables globales
7 unsigned int millar = 0;
8 unsigned int centena = 0;
9 unsigned int decena = 0;
10 unsigned int unidad = 0;
11 float x_var = 0;
12
13 void configuro(void){}
14
15 void CCP1_config(void){
16     PR2 = 155;
17     CCP1R1 = 78;
18     TRISChbits.RC2 = 0;
19     T2CON = 0x07;
20     CCPICON = 0x0C;
21 }
22
23 void ADC_config(void){
24     ADCON2 = 0x24; //ADFM=0 (just izq)
25     ADCON1 = 0x0D; //A0 y A1
26     ADCON0 = 0x05; //A1 seleccionado, AD On
27 }
28
29 void lcd_init(void){
30     TRISE = 0x00;
31     LCD_CONFIG();
32     __delay_ms(15);
33     BORRAR_LCD();
34     CURSOR_HOME();
35     CURSOR_ONOFF(OFF);
36 }
37
38 void convierte(unsigned int numero){
39     millar = (numero % 10000) / 1000;
40     centena = (numero % 1000) / 100;
41     decena = (numero % 100) / 10;
42     unidad = numero % 10;
43 }
44
45 void main(void){
46     configuro();
47     CCP1_config();
48     ADC_config();
49     lcd_init();
50     POS_CURSOR(1,0);
51     ESCRIBE_MENSAJE("Lab Semana 11", 13);
52     while(1){
53         ADCON0bits.GODONE = 1;
54         while(ADCON0bits.GODONE == 1);
55         x_var = ADRESH * 0.61;
56         CCP1R1 = x_var;
57         x_var = ADRESH * 0.39;
58         convierte(x_var);
59         POS_CURSOR(2,0);
60         ESCRIBE_MENSAJE("Backlight: ", 11);
61         ENVIA_CHAR(centena + 0x30);
62         ENVIA_CHAR(decena + 0x30);
63         ENVIA_CHAR(unidad + 0x30);
64         ENVIA_CHAR('%');
65     }
66 }

```

9

## Notas adicionales del A/D:

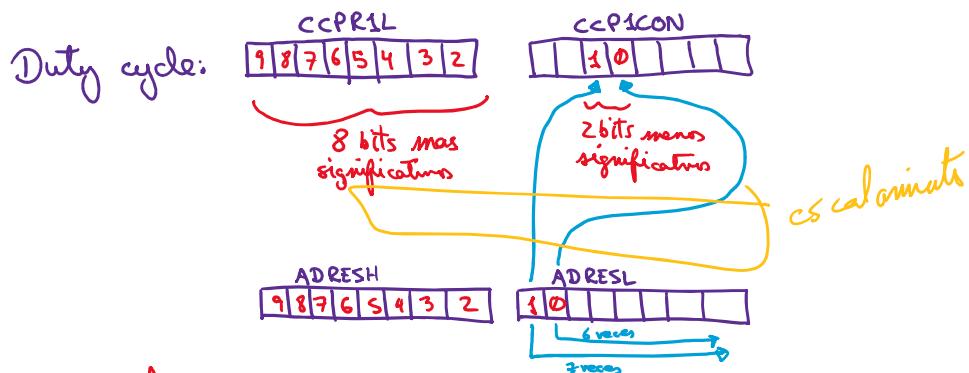
Cómo hacer operaciones con ADFM=0



10

## Notas adicionales sobre CCPx en modo PWM

Detalle de los dos bits menos significativos del Duty Cycle del PWM en el CCP



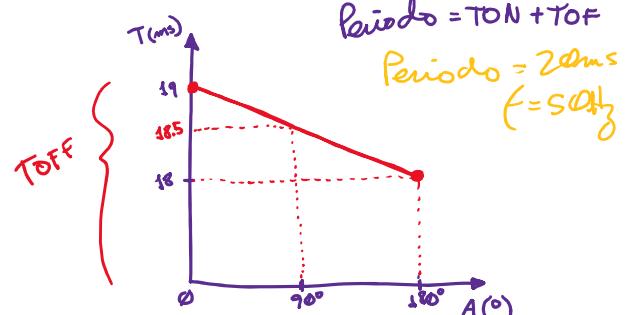
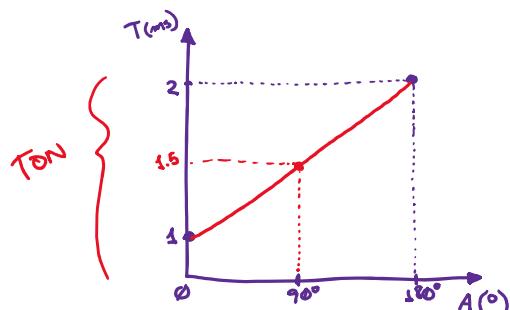
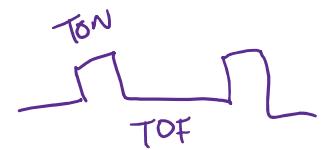
*Note: Análisis errado puesto que CCPR1L como máximo debe de ser PR2*

11

## Etapa de generación de señal al servo

Análisis:

- Configuración del Timer0 modo 16 bits:



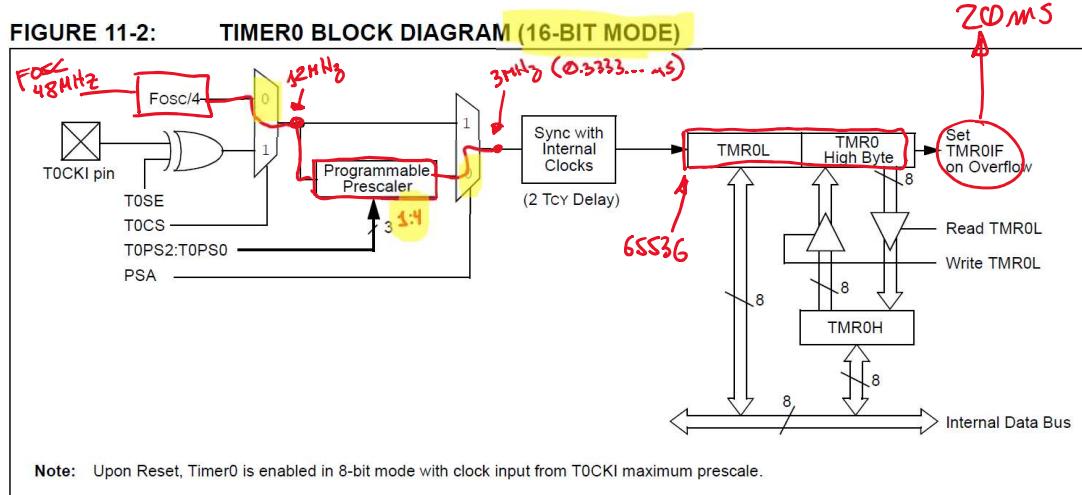
- Debes temporizar con el Timer0 para TON y para TOF por separado

12

## Etapa de generación de señal al servo

### Análisis:

- Configuración del Timer0 modo 16 bits y Fosc 48MHz:



13

## Etapa de generación de señal al servo

### Análisis:

- Configuración del Timer0 modo 16 bits: Temporización 20ms
- Usando prescaler 1:4 y Fosc 48MHz

$$\begin{aligned} 65536 &\rightarrow 21.84533 \text{ ms} \\ X &\rightarrow 20 \text{ ms} \\ X = \frac{65536 \times 20}{21.84533} &= 60000 \leftarrow \begin{array}{l} \text{Total de cuentas} \\ \text{que debe de hacer} \\ \text{Timer 0} \end{array} \\ \Rightarrow \text{Cuenta inicial} : 65536 - 60000 &= 5536 \end{aligned}$$

$T = 20 \text{ ms}$

$\downarrow T_{ON}$        $\uparrow T_{OFF}$   
 1-2ms      18-19ms

$\omega \times 15$        $\omega \times 10$   
 $\text{TMR0H} : \text{TMR0L}$

14

## Etapa de generación de señal al servo

### Análisis:

- Configuración del Timer0 modo 16 bits: Temporización 20ms

Según lo anterior, debo de calcular para 1ms-2ms (TON)

Servo: 20ms — 60000

0° → 1ms — X

$$X = \frac{60000}{20}$$

$$X = 3000$$

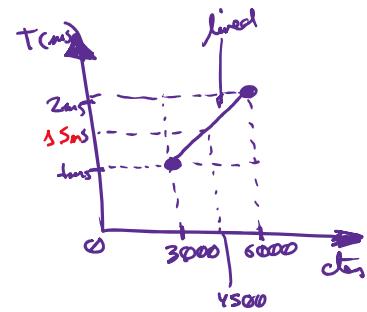
1ms

180° → 20ms — 60000

2ms — X

$$X = \frac{60000(2)}{20}$$

$$X_{2ms} = 60000$$



cuenta inicial: }  
 $65536 - X$  }  
 1ms → 62536  
 2ms → 59536

15

## Etapa de generación de señal al servo

### Análisis:

- Configuración del Timer0 modo 16 bits: Temporización 20ms

Según lo anterior, debo de calcular para 18ms-19ms (TOF)

Servo: 20ms — 60000

180° → 18 ms — X

$$X = \frac{60000(18ms)}{20ms}$$

$$X = 54000$$

18ms

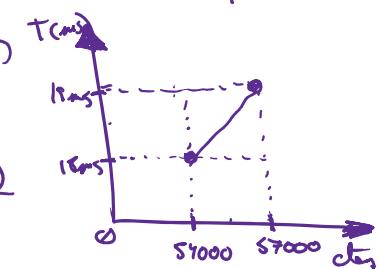
0° → 20ms — 60000

19ms — X

$$X = \frac{60000(19ms)}{20ms}$$

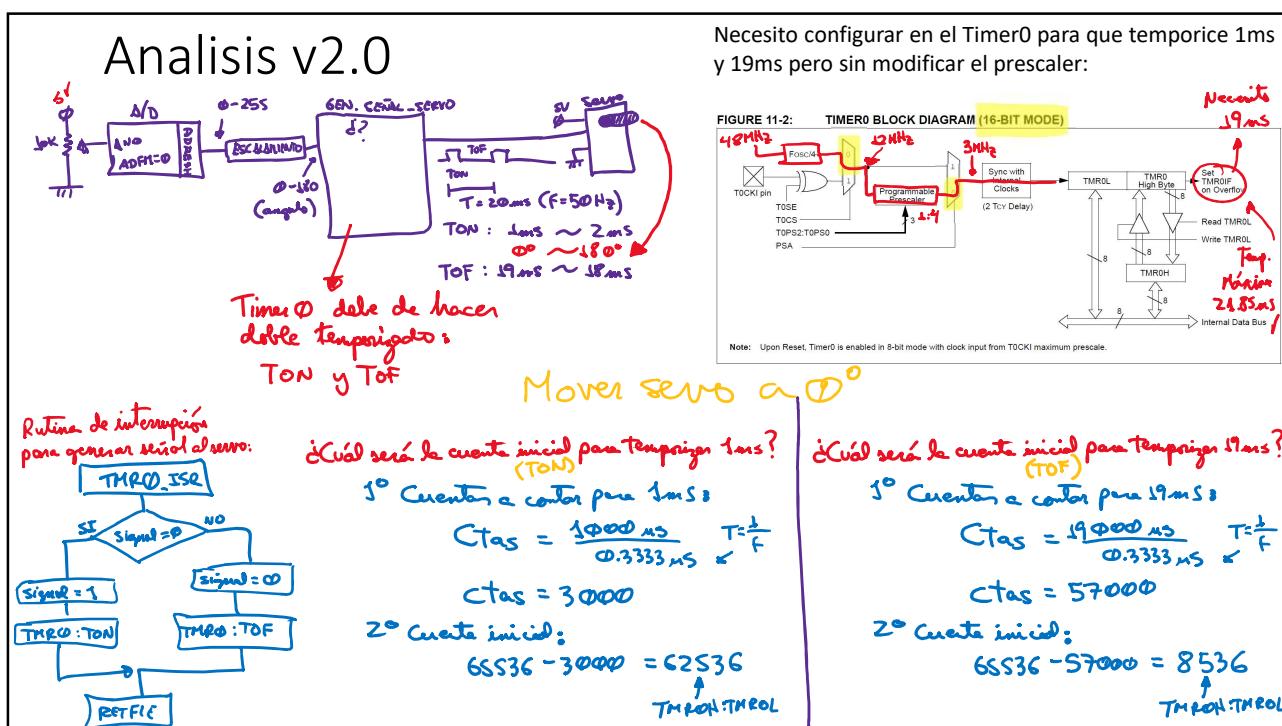
$$X = 57000$$

19ms



cuenta inicial: }  
 $65536 - X$  }  
 18ms : 11536  
 19ms : 8536

16



17

## Etapa de generación de señal al servo

- Código para enviar señal al servo desde RE0 para que se mueva a 0°:

```

5 void configuro(void) {
6     ADCON1 = 0x0F;
7     TRISEbits.RE0 = 0;
8     LATEbits.LE0 = 0;
9 }
15
16 void TMRO_config(void) {
17     TOCON = 0x81;
18     INTCON = 0xA0;
19 }
20
21 void main(void) {
22     configuro();
23     TMRO_config();
24     while(1);
25 }
  
```

Para 1ms: Cuenta inicial TON = 62536  
 en hexadecimal: 0xF448

Para 19ms: Cuenta inicial TOF = 8536      TMROH  
 en hexadecimal: 0x2158

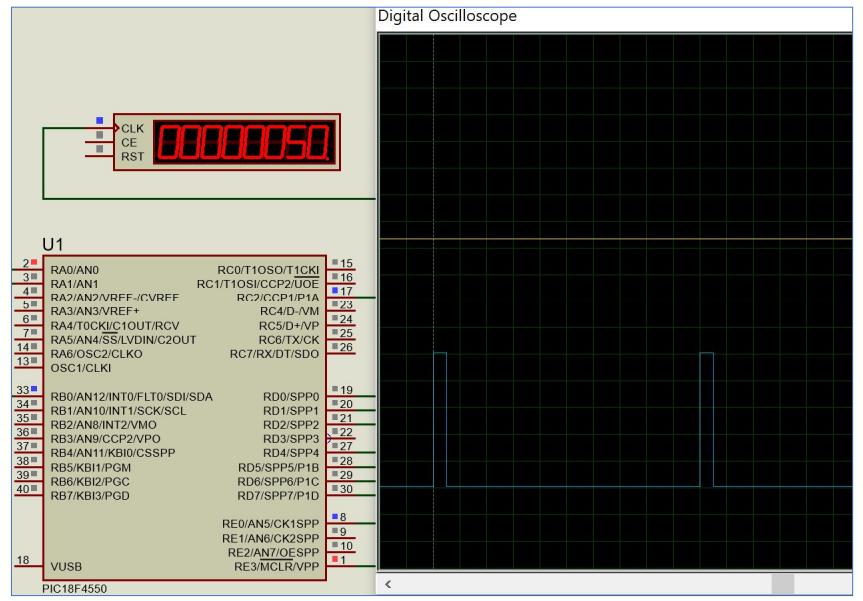
```

27 void __interrupt() TMRO_ISR(void){
28     if(PORTEbits.RE0 == 0){
29         LATEbits.LE0 = 1;
30         TMROH = 0xF4; } 1ms
31         TMROL = 0x48;
32     }
33     else{
34         LATEbits.LE0 = 0;
35         TMROH = 0x21; } 19ms
36         TMROL = 0x58;
37     }
38     INTCONbits.TMR0IF = 0;
39 }
  
```

18

## Etapa de generación de señal al servo

- Simulación en Proteus



19

## Etapa de generación de señal al servo

- Código para enviar señal al servo desde RE0 para que se mueva entre 0° y 180° según RB0

```

5 void configuro(void) {
6     ADCON1 = 0x0F;
7     TRISBbits.RE0 = 0;
8     LATBbits.LE0 = 0;
9 }

```

```

27 void __interrupt() TMR0_ISR(void){
28     if(PORTBbits.RB0 == 0){
29         if(PORTEbits.RE0 == 0){
30             LATEbits.LE0 = 1;
31             TMROH = 0xF4;
32             TMROL = 0x48; } 5ms
33         } else{
34             LATEbits.LE0 = 0;
35             TMROH = 0x21;
36             TMROL = 0x58; } 15ms
37     }
38     else{
39         if(PORTEbits.RE0 == 0){
40             LATEbits.LE0 = 1;
41             TMROH = 0xE8;
42             TMROL = 0x90; } 2ms
43         } else{
44             LATEbits.LE0 = 0;
45             TMROH = 0x2D;
46             TMROL = 0x10; } 18ms
47     }
48 }
49 INTCONbits.TMR0IF = 0;
50 }
51 }
52 }
53 }

```

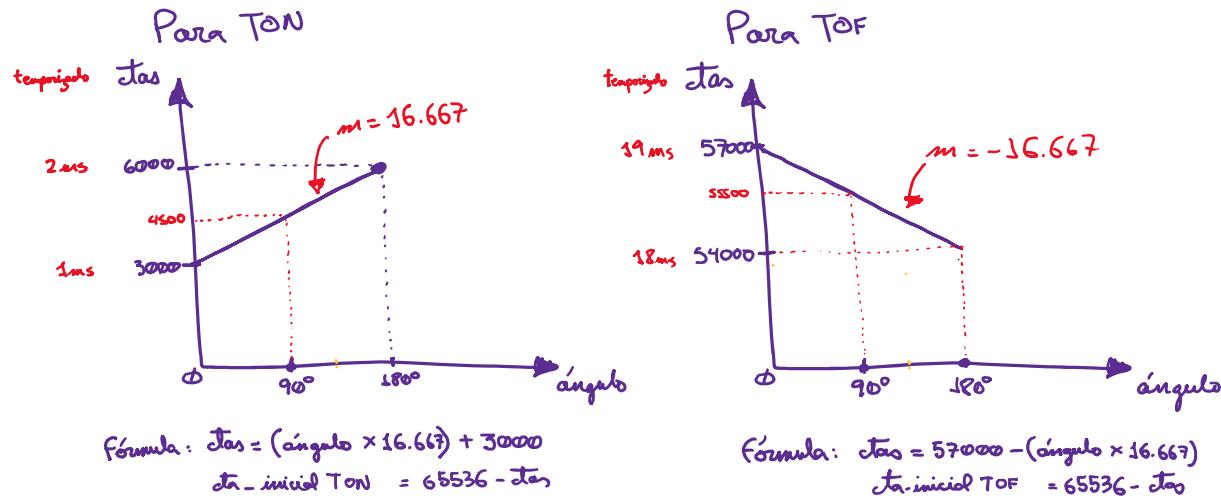
*Servo:*

0°      5ms      15ms      2ms      18ms

20

## Análisis v3.0

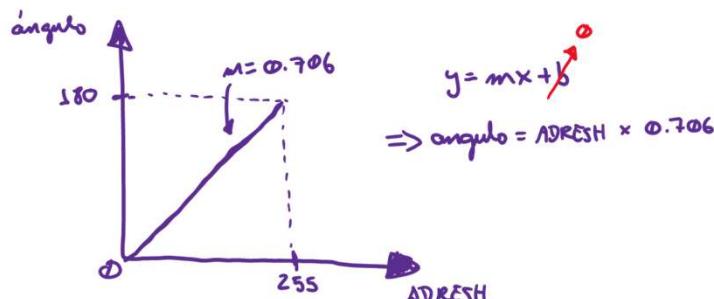
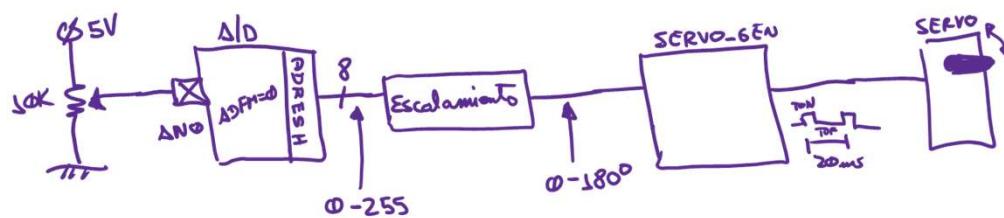
- Gráficas de ángulos vs periodo de temporizado tanto en Ton como en Tof



21

## Análisis v3.0

- Escalamiento entre lectura de AN0 y ángulo del servo:



22

## Análisis v3.0

- Ecuaciones finales para cuentas iniciales de TON y TOF con respecto a ADRESH (del AN0)

$$cta\_inicial\_ton = 65536 - ((ADRESH \times 0.706) \times 16.667) + 3000$$

$$cta\_inicial\_tof = 65536 - (57000 - ((ADRESH \times 0.706) \times 16.667))$$

23

## Código propuesto:

- Solo se encuentra la lectura de AN0 y la aplicación de las formulas obtenidas para las cuentas iniciales de TON y TOF orientadas a generar la señal del servo:

```

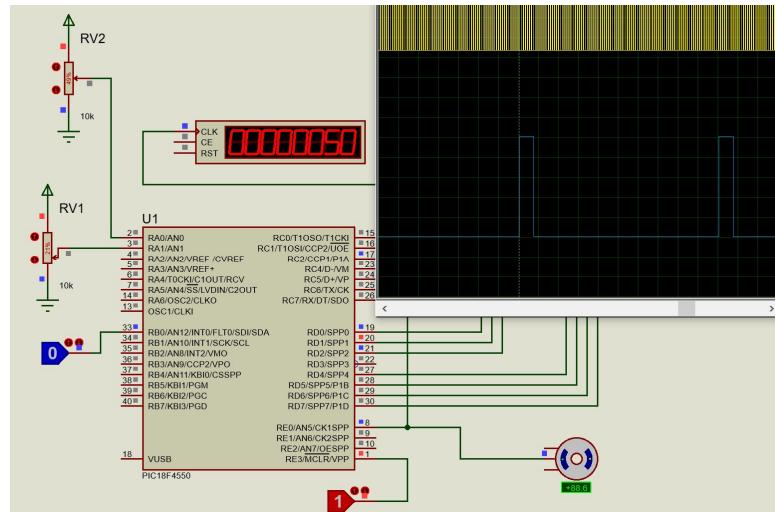
6 //declaracion de variables globales
7 float calculo_ton = 0;
8 float calculo_tof = 0;
9 unsigned int cta_inicial_ton = 0;
10 unsigned int cta_inicial_tof = 0;
11
12 void configuro(void){
13     TRISBbits.RE0 = 0;
14     LATBbits.LE0 = 0;
15 }
16
17 void ADC_config(void){
18     ADCON2 = 0x24;           //8stad fosc/4 adfm=0
19     ADCON1 = 0x0D;           //an0 y an1
20     ADCON0bits.ADON = 1;    //encendemos el ADC
21 }
22
23 void TMR0_config(void){
24     T0CON = 0x81;
25     INTCON = 0xA0;
26 }
```

```

29 void main(void) {
30     configuro();
31     TMR0_config();
32     ADC_config();
33     while(1){
34         ADCON0 = 0x03;          //tome una muestra en an0
35         while(ADCON0bits.GODONE == 1);
36         calculo_ton = ((ADRESH * 0.706) * 16.667) + 3000;
37         cta_inicial_ton = 65536 - calculo_ton;
38         calculo_tof = 57000 - ((ADRESH * 0.706) * 16.667);
39         cta_inicial_tof = 65536 - calculo_tof;
40     }
41 }
42
43 void __interrupt() TMR0_ISR(void){
44     if(PORTBbits.RE0 == 0){
45         LATBbits.LE0 = 1;
46         TMR0H = cta_inicial_ton >> 8;
47         TMR0L = cta_inicial_ton & 0x00ff;
48     }
49     else{
50         LATBbits.LE0 = 0;
51         TMR0H = cta_inicial_tof >> 8;
52         TMR0L = cta_inicial_tof & 0x00ff;
53     }
54     INTCONbits.TMR0IF = 0;
55 }
```

24

## Simulación

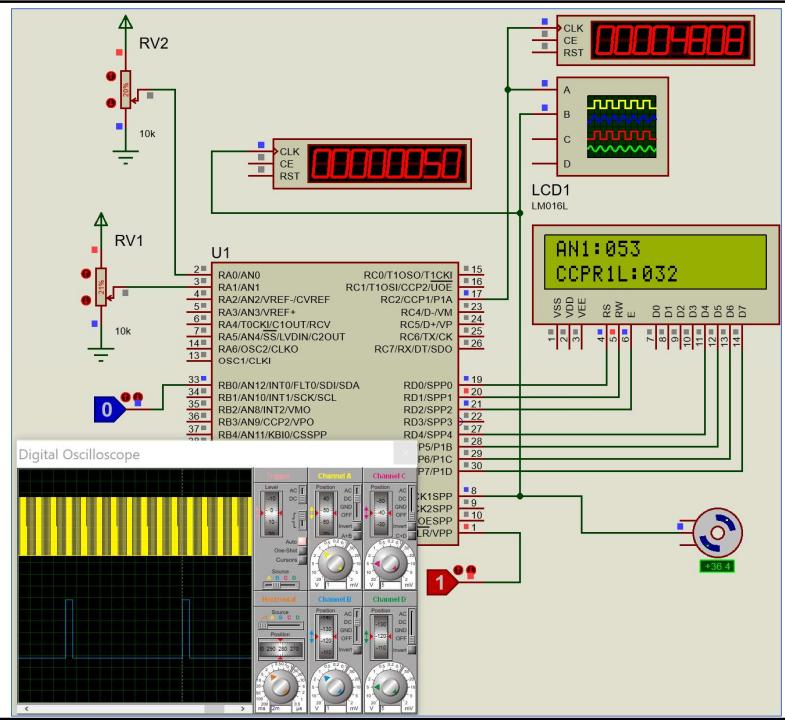


25

## Código final completo

26

Simulación completa:



27

Fin de la sesión!

- Semana 12: LB3

28