

Microcontroladores Laboratorio Sesión 10

Semestre: 2021-1

Profesor: Kalun José Lau Gan

1

Agenda:

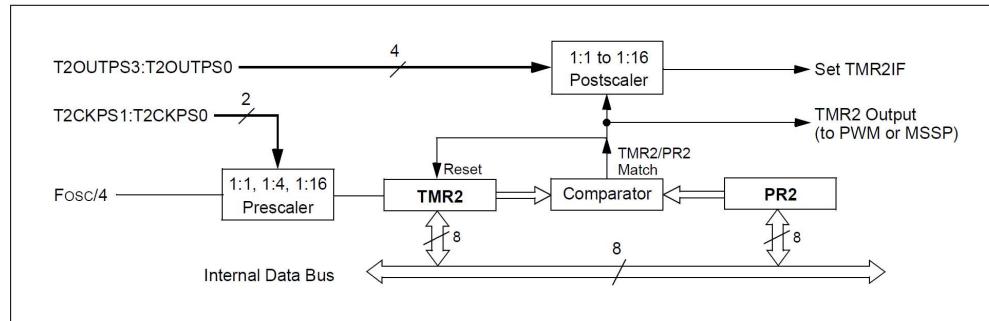
- Timer 2
- PWM con CCP

2

Timer 2:

- Temporizador con prescaler (divide la frecuencia de entrada) y postscaler (para contabilizar la cantidad de igualdades antes de levantar la bandera TMR2IF)
- Posee un comparador entre la cuenta actual (TMR2) y un valor de referencia (PR2), se emitirá una señal de igualdad (match) si $TMR2 = PR2$.
- Ante un evento de igualdad ($TMR2 = PR2$) la cuenta del Timer2 se reinicia
- 8 bits de resolución
- Solo hay una opción de fuente de reloj: Fosc/4
- Nunca se desborda!

FIGURE 13-1: TIMER2 BLOCK DIAGRAM

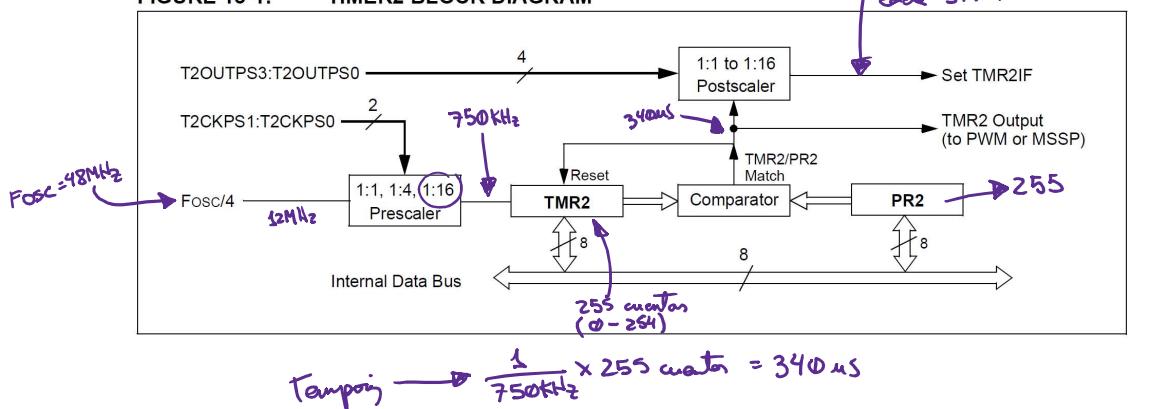


3

Temporizado máximo en el Timer 2 si Fosc=48MHz

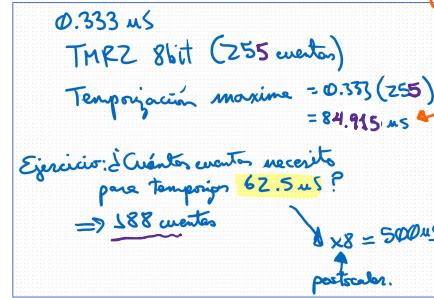
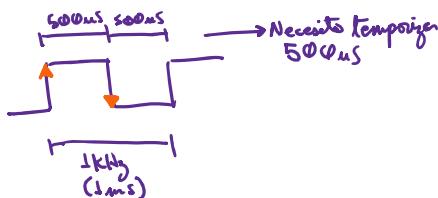
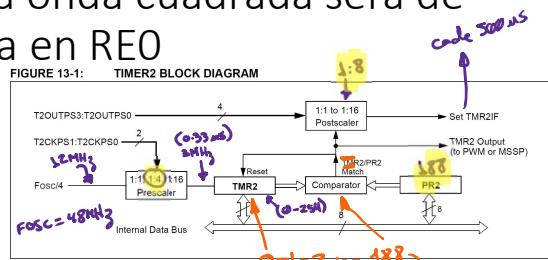
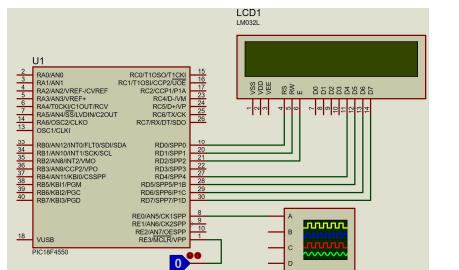
- Empleando prescaler 1:16 y postscaler 1:16:

FIGURE 13-1: TIMER2 BLOCK DIAGRAM



4

Ejemplo: Desarrollar un generador de onda cuadrada empleando el Timer2, dicha onda cuadrada será de 1KHZ DC 50% y será emitida en RE0



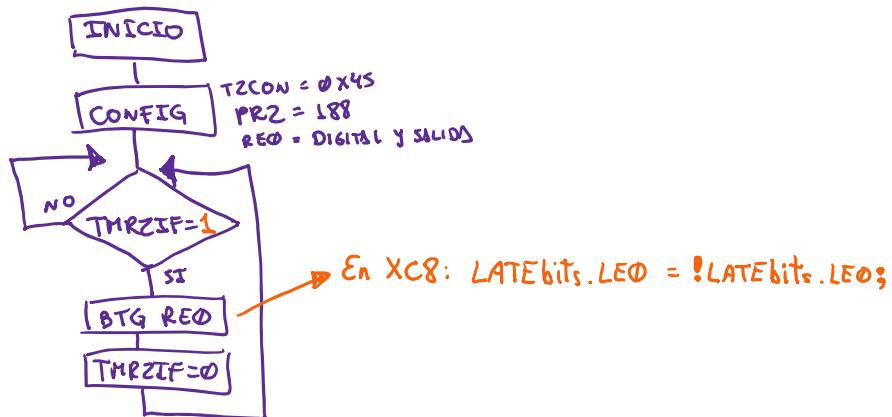
5

Ejemplo: Desarrollar un generador de onda cuadrada empleando el Timer2, dicha onda cuadrada será de 1KHZ DC 50%

REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7	1	0	0	0	1	0	1
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	Unimplemented: Read as '0'						
bit 6-3	T2OUTPS3:T2OUTPS0: Timer2 Output Postscale Select bits						
	0000 = 1:1 Postscale						
	0001 = 1:2 Postscale						
	.						
	1000 = 1:8						
	.						
	1111 = 1:16 Postscale						
bit 2	TMR2ON: Timer2 On bit						
	1 = Timer2 is on						
	0 = Timer2 is off						
bit 1-0	T2CKPS1:T2CKPS0: Timer2 Clock Prescale Select bits						
	00 = Prescaler is 1						
	01 = Prescaler is 4						
	1x = Prescaler is 16						

$$\underline{1887 = 188}$$

Diagrama de flujo:



7

Código en XC8

- Nótese que no se está empleando interrupciones para detectar el evento de TMR2.

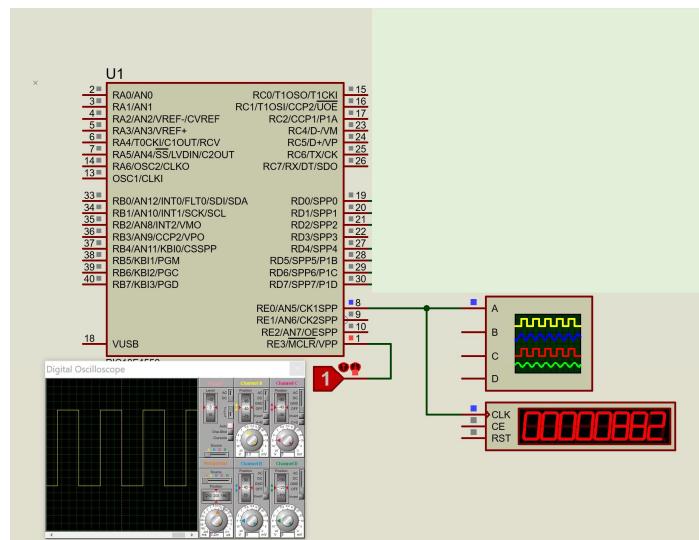
```

1  #pragma config PLLDIV = 1      // PLL
2  #pragma config CPUDIV = OSC1_PLL2 // Syst
3  #pragma config FOSC = XTPLL_XT // Oscil
4  #pragma config PWRT = ON       // Power
5  #pragma config BOR = OFF        // Brown
6  #pragma config WDT = OFF        // Watch
7  #pragma config CCP2MX = ON      // CCP2
8  #pragma config PBADEN = OFF     // PORTE
9  #pragma config MCLRE = ON       // MCLR
10 #pragma config LVP = OFF        // Singl
11
12 #include <xc.h>
13 #define _XTAL_FREQ 48000000UL //Frecue
14
15 void tmr2_conf(void){
16     T2CON = 0x45;
17     PR2 = 188;
18 }
19
20 void main(void){
21     tmr2_conf();
22     ADCON1 = 0x0F;
23     TRISEbits.REO = 0;
24     while(1){
25         while(PIR1bits.TMR2IF == 0);
26         LATEbits.LE0 = !LATEbits.LE0;
27         PIR1bits.TMR2IF = 0;
28     }
29 }

```

8

Simulación



9

Mejora: Empleando interrupciones

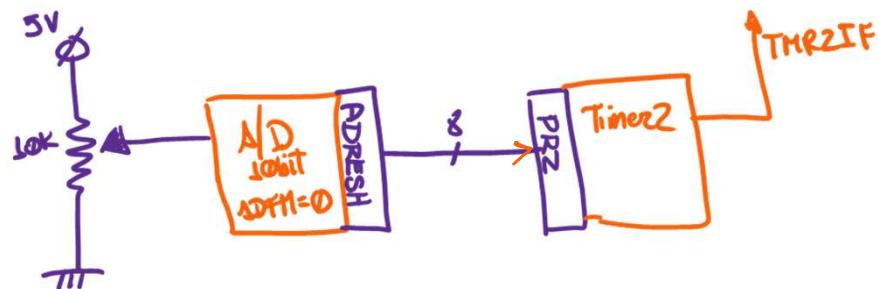
- Tener en cuenta el nuevo formato para la declaración de la función de interrupción en XC8
- La onda cuadra resultante no tiene 1KHz requerido por el enunciado por lo que se deberá modificar el valor de PR2 del Timer2 para ajustar a la frecuencia solicitada.

```

13 #define _XTAL_FREQ 48000000UL //Frecuencia del
14
15 void tmr2_conf(void){
16     T2CON = 0x45;
17     PR2 = 188;
18 }
19
20 void int_conf(void){
21     INTCONbits.GIE = 1;
22     INTCONbits.PEIE = 1;
23     PIE1bits.TMR2IE = 1;
24 }
25
26 void main(void){
27     tmr2_conf();
28     int_conf();
29     ADCON1 = 0x0F;
30     TRISBbits.RE0 = 0;
31     while(1);
32 }
33
34 void __interrupt(high_priority) High_ISR(void){
35     LATEbits.LE0 = !LATEbits.LE0;
36     PIR1bits.TMR2IF = 0;
37 }
```

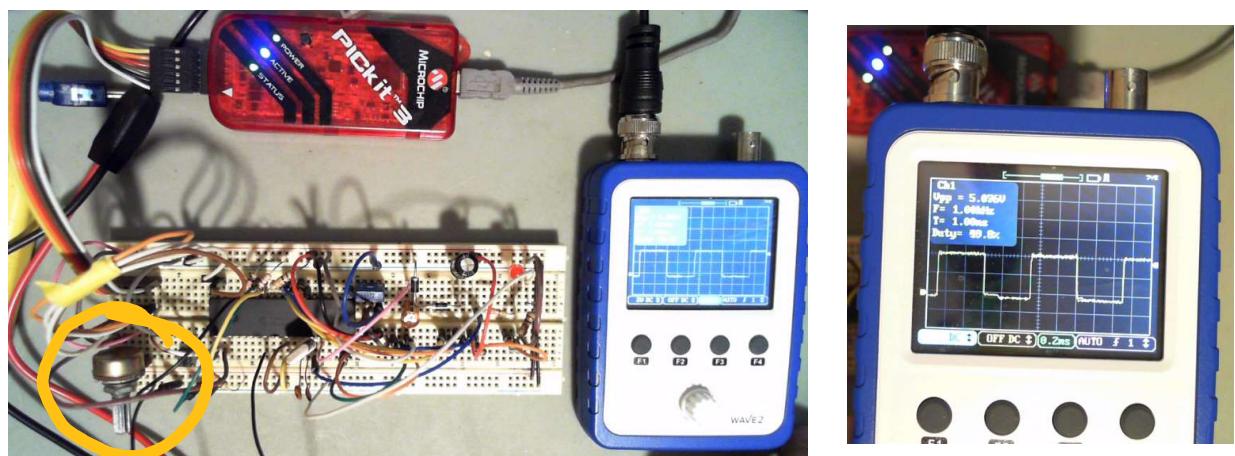
10

Mejora: Utilizando un potenciómetro en AN0 para ajustar el valor de PR2 del Timer2



11

Mejora: Utilizando un potenciómetro en AN0 para ajustar el valor de PR2 del Timer2



12

Mejora: Utilizando un potenciómetro en AN0 para ajustar el valor de PR2 del Timer2

- En la implementación física del circuito hemos notado que hay mucho ruido producto del potenciómetro conectado a AN0
- Se ha implementado un filtro de promedio de 20 valores para dicha entrada.

```

49 void main(void) {
50     init_conf();
51     lcd_conf();
52     POS_CURSOR(1,0);
53     ESCRIBE_MENSAJE(" SqrWave Gen ",15);
54     while(1){
55         unsigned char x;
56         promedio = 0;
57         for(x=0;x<20;x++){
58             ADCON0bits.GODONE = 1;
59             while(ADCON0bits.GODONE == 1);
60             promedio = promedio + ADRESH;
61         }
62         promedio = promedio / 20;
63         PR2 = promedio;
64         POS_CURSOR(2,0);
65         ESCRIBE_MENSAJE("PR2:",4);
66         convierte(PR2);
67         ENVIA_CHAR(centena+0x30);
68         ENVIA_CHAR(decena+0x30);
69         ENVIA_CHAR(unidad+0x30);
70     }
71 }
```

13

Mejora: Añadimos la funcionalidad del LCD para visualizar el valor que se asigna a PR2

- Según la simulación, el valor mas cercano que se obtiene en PR2 al mover el potenciómetro es 166, con ello la señal cuadrada en RE0 es de 998Hz.

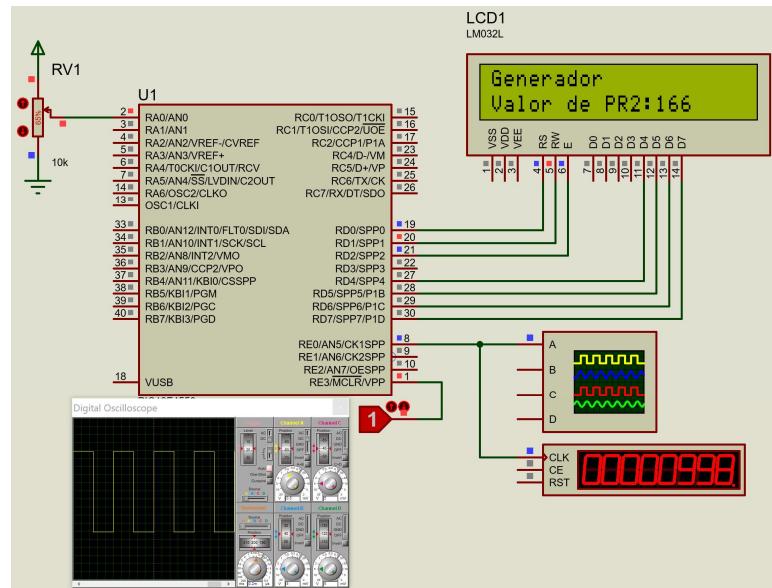
```

1 #pragma config PLLDIV = 1 // FLL Pres
2 #pragma config CPUDIV = OSC1_PLL2// System
3 #pragma config FOSC = XTPLL_XT // Oscillat
4 #pragma config FWRT = ON // Power-up
5 #pragma config BOR = OFF // Brown-out
6 #pragma config WDT = OFF // Watchdog
7 #pragma config CCP2MX = ON // CCP2 MUX
8 #pragma config PBADEN = OFF // PORTB A/
9 #pragma config MCORE = ON // MCLR Pin
10 #pragma config LVP = OFF // Single-S
11
12 #include <xc.h>
13 #include "LCD.h"
14 #define _XTAL_FREQ 48000000UL //Frecuencia
15
16 unsigned int var_gen = 0;
17
18 unsigned int millar = 0;
19 unsigned int centena = 0;
20 unsigned int decena = 0;
21 unsigned int unidad = 0;
22
23 void convierte(unsigned int numero){
24     millar = numero /1000;
25     centena = (numero % 1000) / 100;
26     decena = (numero % 100) / 10;
27     unidad = numero % 10;
28 }
29
30 void lcd_init(void) {
31     TRISD = 0x00; //Puerto RD
32     LCD_CONFIG();
33     _delay_ms(15);
34     BORRAR_LCD();
35     CURSOR_HOME();
36     CURSOR_ONOFF(OFF);
37 }
38
39 void tmr2_conf(void){
40     T2CON = 0x45;
41     PR2 = 164;
42 }
43
44 void int_conf(void){
45     INTCONbits.GIE = 1;
46     INTCONbits.PEIE = 1;
47     PIEbits.TMR2IE = 1;
48 }
49
50 void ad_conf(){
51     ADCON2 = 0x24; //ADM0=0 (ju
52     ADCON1 = 0x0E; //Canal AN0
53     ADCON0 = 0x01; //Canal AN0 s
54 }
55
56 void main(void){
57     lcd_init();
58     tmr2_conf();
59     ad_conf();
60     int_conf();
61     //ADCON1 = 0x0F;
62     TRISEbits.RE0 = 0;
63     POS_CURSOR(1,0);
64     ESCRIBE_MENSAJE("Generador",9);
65     while(1){
66         ADCON0bits.GODONE = 1;
67         while(ADCON0bits.GODONE == 1); //In
68         PR2 = ADRESH;
69         var_gen = ADRESH;
70         convierte(var_gen);
71         POS_CURSOR(2,0);
72         ESCRIBE_MENSAJE("Valor de PR2:",13);
73         ENVIA_CHAR(centena+0x30);
74         ENVIA_CHAR(decena+0x30);
75         ENVIA_CHAR(unidad+0x30);
76     }
77 }
78
79 void __interrupt(high_priority) High_ISR(void) {
80     LATEnbits.LE0 = !LATEnbits.LE0;
81     PIR1bits.TMR2IF = 0;
82 }
```

14

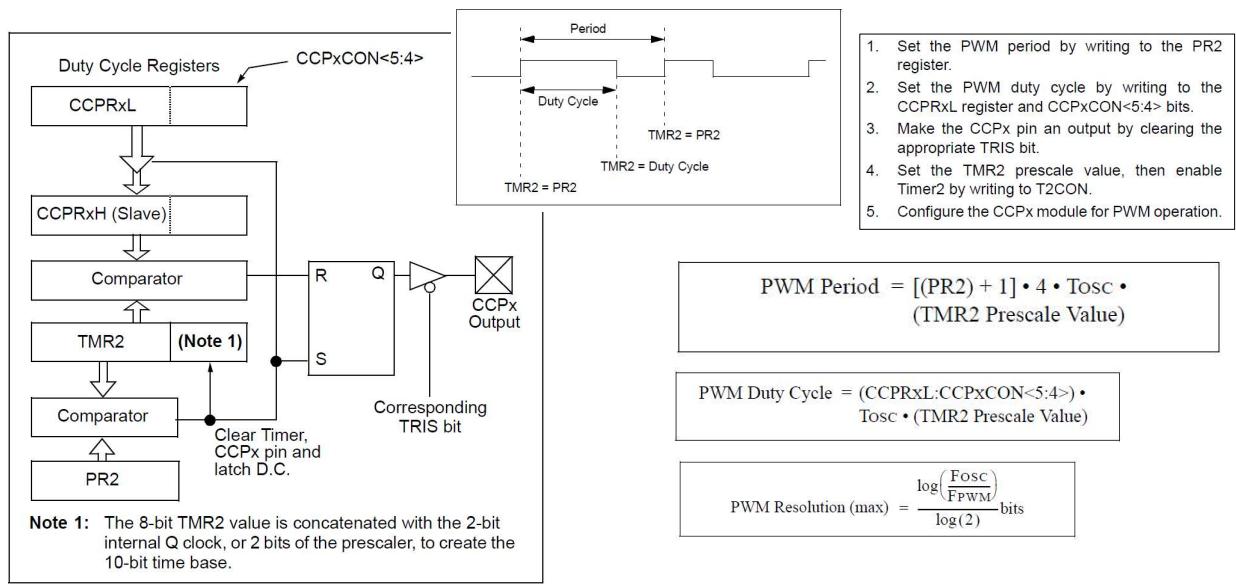
Mejora: Añadimos la funcionalidad del LCD para visualizar el valor que se asigna a PR2

- Simulación



15

PWM con el CCP (extraído de la hoja técnica)



16

Calcular el valor de PR2 para obtener una onda de 2.35KHz y DC 50% teniendo en cuenta Fosc 48MHz

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

Conclusión:
Con FOSC=48MHz
no se puede
generar PWM a 2.35KHz



$$\frac{1}{2350} = [(PR2) + 1] \cdot 4 \cdot \frac{1}{48E6} \cdot 16$$

$$PR2 = 318.15$$

Registro de 8bit $\Rightarrow 318$

¿Se puede
ingresar
en PR2?
Nooooo

17

Si necesito una onda PWM con freq de 10KHz:

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

$$\frac{1}{10K} = [(PR2 + 1)] \cdot 4 \cdot \left(\frac{1}{48E6}\right) \cdot 16$$

$$PR2 = [0 \sim 255]$$

$$PR2 = 74$$

Condición

Si DC 50%

$$\Rightarrow CCPRL = 37$$

Si se puede
usar

18

Si necesito una onda PWM con frecuencia de 25KHz:

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

$$\frac{1}{25\text{K}} = [(PR2 + 1) \cdot 4 \cdot \left(\frac{1}{48\text{MHz}}\right) \cdot (16)]$$

$$PR2 = [0 \sim 255]$$

$$PR2 = 29$$

Condición

⚠ Si se puede usar

Si DC 30%

$$\Rightarrow CCPRL = 9$$

$$CCPRL = 29$$

= 100%

$$CCPRL = 14 \rightarrow 50\%$$

19

Ejemplo: Si necesito una onda PWM con frecuencia de 4.8Khz:

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

PR2?

↳ 155 (PSC 1:16)

Duty Cycle:

$$CCPRL: 78$$

20

Código en XC8

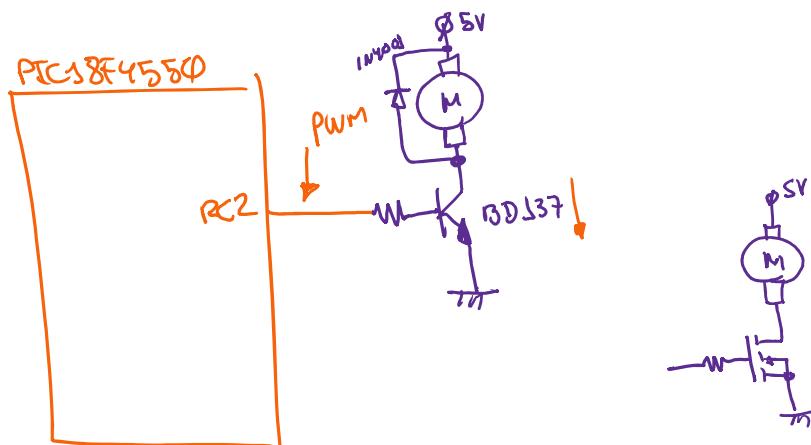
```

23 #include <xc.h>
24 #define _XTAL_FREQ 48000000UL //Frecuencia de trabajo 48MHz
25
26 void configuracion() {
27     PR2 = 155; //Para el periodo (freq 155KHz en PWM CCP)
28     CCPR1L = 78; //Para el duty cycle (50%)
29     TRISCBits.RC2 = 0; //Puerto RC2/CCP1 como salida
30     T2CON = 0x07; //Timer2 ON con PreSC 1:16
31     CCP1CON = 0x0C; //CCP1 en modo PWM
32 }
33
34 void main(void) {
35     configuracion();
36     while(1) {
37     }
38 }
```

4.8

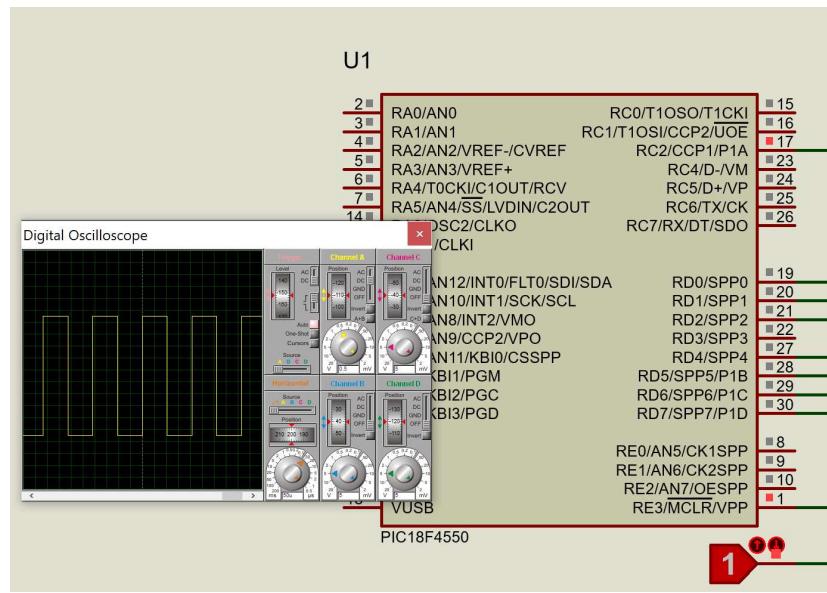
21

Ejemplo de aplicación: Control de las RPMs de un motor DC con PWM



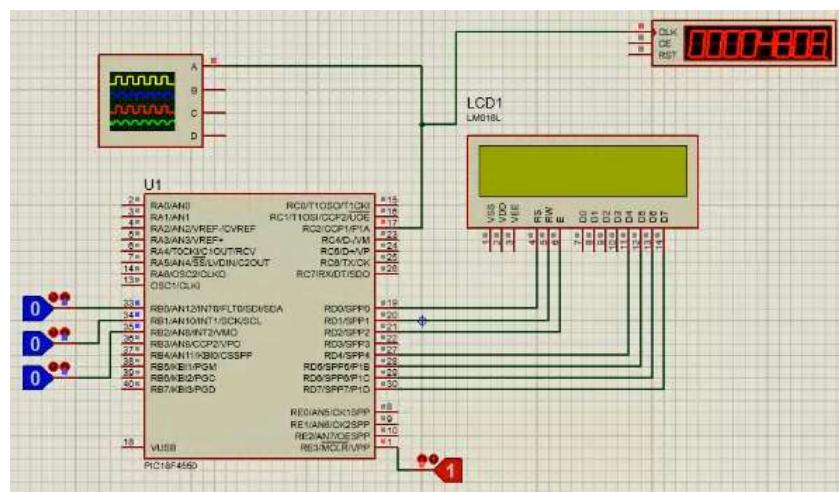
22

Simulación



23

Simulación:



24

Fin de la sesión!

- Modificar el ejemplo anterior para que el duty cycle sea manipulado por un potenciómetro conectado a AN0 (deberás de emplear el A/D del microcontrolador de tal modo que el resultado de la conversión se envié al registro de duty cycle del PWM).

