

Microcontroladores

Laboratorio Sesión 10

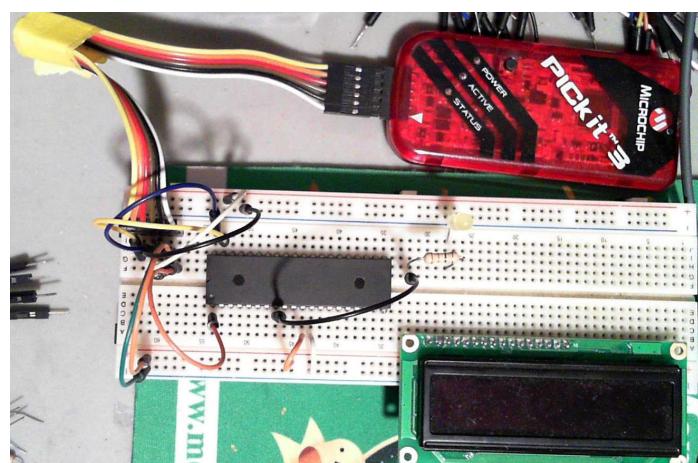
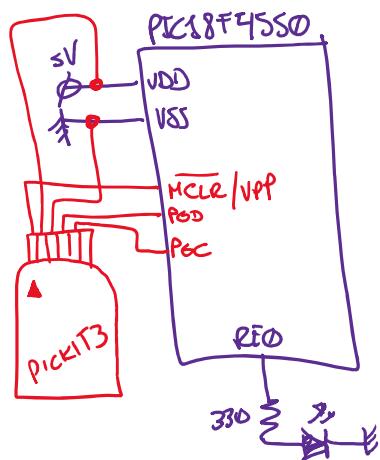
Semestre: 2022-1

Profesor: Kalun José Lau Gan

1

Preguntas previas:

- No funciona mi circuito 😞



2

Preguntas previas:

- No funciona mi circuito 😞

```

6 // CONFIG1L
7 #pragma config PLLDIV = 1           //
8 #pragma config CPUDIV = OSC1_PLL2//
9 #pragma config USBDIV = 1           //
10
11 // CONFIG1H
12 #pragma config FOSC = INTOSCIO_EC//
13 //##pragma config FOSC = XTPLL_XT
14 #pragma config FCMEN = OFF          //
15 #pragma config IESO = OFF          //
16
17 // CONFIG2L
18 #pragma config PWRT = ON           //
19 #pragma config BOR = OFF           //
20 #pragma config BORV = 3             //
21 #pragma config VREGEN = OFF         //
22
23 // CONFIG2H
24 #pragma config WDT = OFF           //
25 #pragma config WDTPS = 32768        //
26
27 // CONFIG3H
28 #pragma config CCP2MX = ON          //
29 #pragma config PBADEN = OFF         //
30 #pragma config LPT1OSC = OFF         //
31 #pragma config MCLRE = OFF          //
32
33 // CONFIG4L
34 #pragma config SIVREN = ON          //
35 #pragma config LVP = OFF            //
36 #pragma config ICPRT = OFF          //
37 #pragma config XINST = OFF          //

```

```

1 #include "cabecera.h"
2 #include <xc.h>
3 #define _XTAL_FREQ 8000000UL
4
5 void configuracion(void){
6     OSCCONbits.IRCF2 = 1;
7     OSCCONbits.IRCF1 = 1;
8     OSCCONbits.IRCFO = 1;    //FOSC (INTOSCIO_EC) = 8MHz
9     ADCON1= 0x0F;           //Todos los puertos en digitales
10    TRISBbits.RE0 = 0;      //RE0 como salida
11 }
12
13 void main(void) {
14     configuracion();
15     while(1){
16         LATEnbits.LE0 = 1;
17         __delay_ms(100);
18         LATEnbits.LE0 = 0;
19         __delay_ms(100);
20     }
21 }
22

```

3

Preguntas previas:

- Me compila todo bien sin errores pero no funciona el circuito y tampoco la simulación. ¿Qué puede ser?
 - Hay que tener en cuenta que una compilación correcta no significa que el algoritmo (aspecto lógico del programa o el planteamiento) este correcto. Revisar primero el hardware y revisar también el proyecto desarrollado tanto en algoritmo como en el MPLABX

4

Agenda:

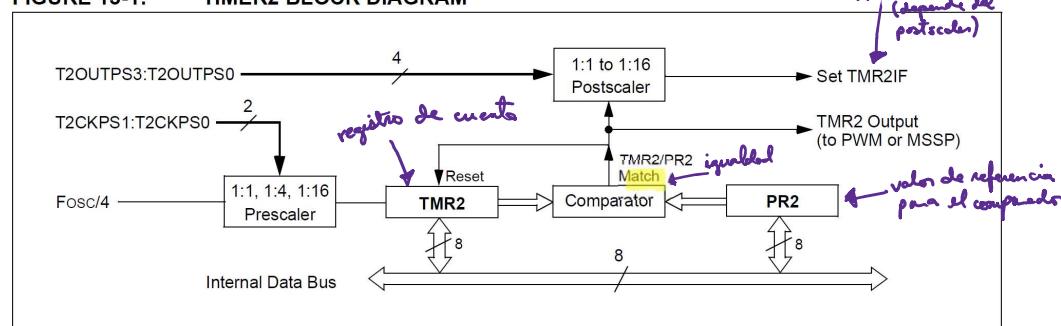
- Timer 2
- PWM con CCPx (Capture / Compare / PWM)

5

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, por lo tanto en las aplicaciones no es necesario hacer precarga de cuenta
- 8 bits de resolución, cuenta ascendente
- Solo hay una opción de fuente de reloj: Fosc/4
- Nunca se desborda!

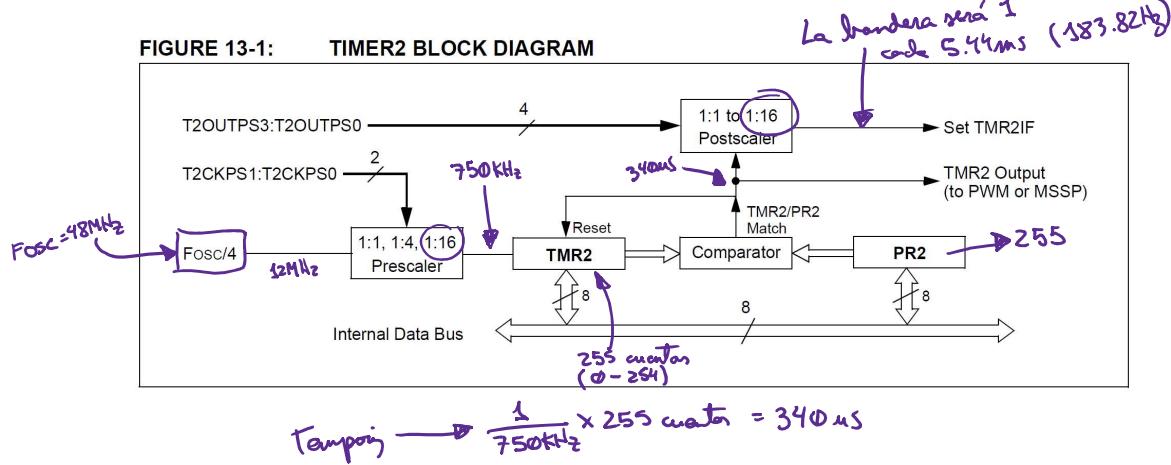
FIGURE 13-1: TIMER2 BLOCK DIAGRAM



6

Temporizado máximo en el Timer 2 si Fosc=48MHz

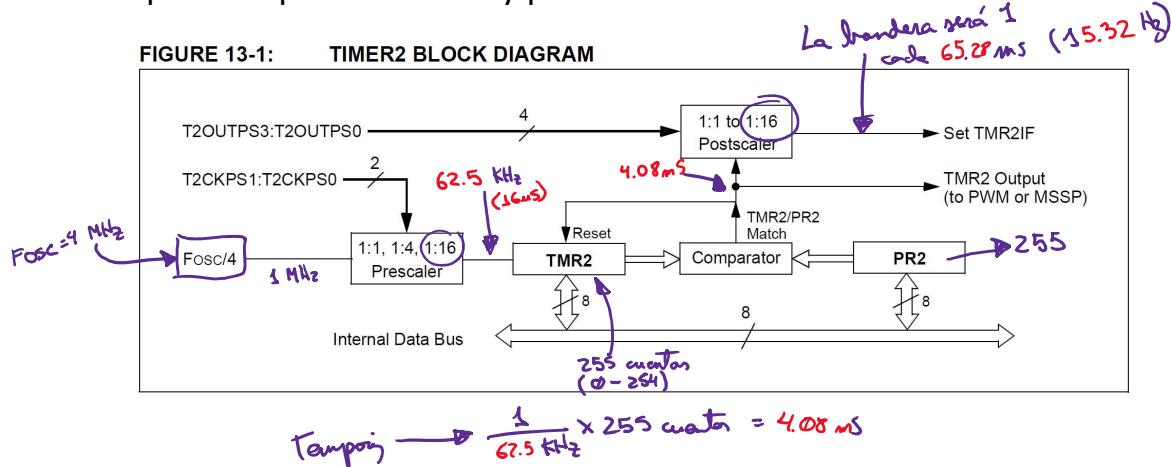
- Empleando prescaler 1:16 y postscaler 1:16:



7

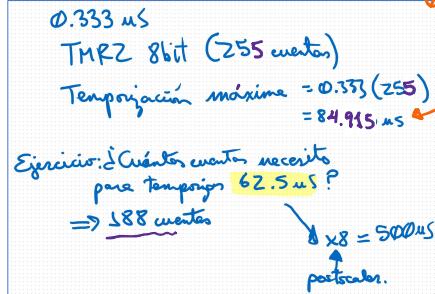
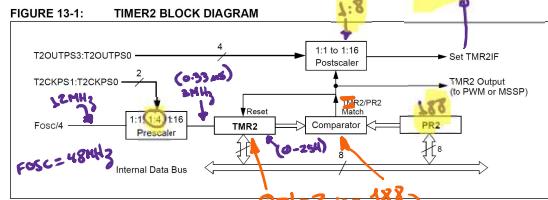
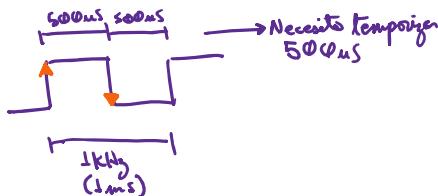
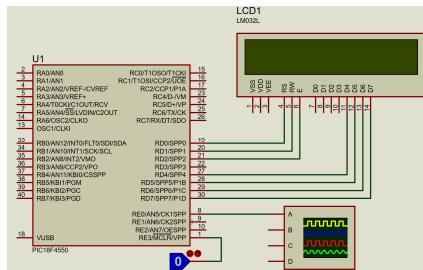
Temporizado máximo en el Timer 2 si Fosc=4MHz

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



8

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



9

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

T2CON
0x39

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	0	1	1	1	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 $0111 = 1:8$
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

PR2 = 188

10

Aclaración en POSTSCALER de Timer2:

bit 6-3

T2OUTPS3:T2OUTPS0: Timer2 Output Postscale Select bits

0000 = 1:1 Postscale

0001 = 1:2 Postscale

⋮ ♂?

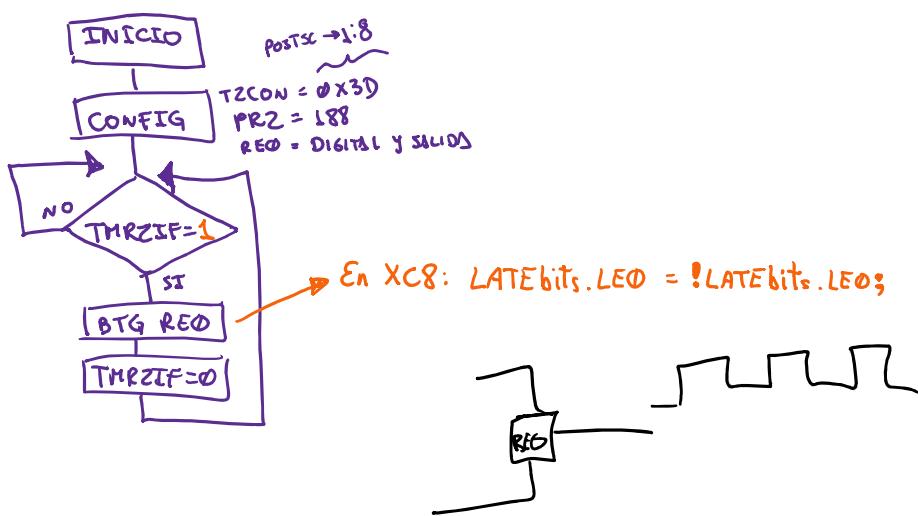
1111 = 1:16 Postscale

$\text{0010} = 1:3$
 $\text{0011} = 1:4$
 $\text{0100} = 1:5$
 $\text{0101} = 1:6$
 $\text{0110} = 1:7$

$\text{0111} = 1:8$ ✓
 $\text{1000} = 1:9$
⋮

11

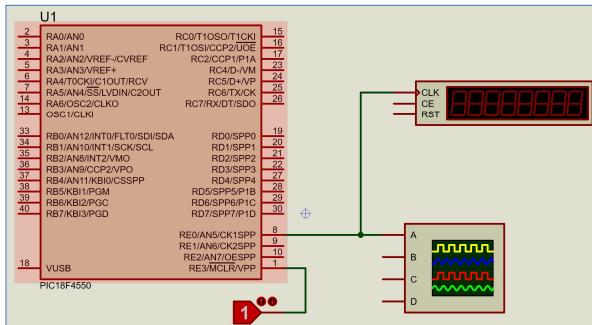
Diagrama de flujo:



12

Código en XC8

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



```

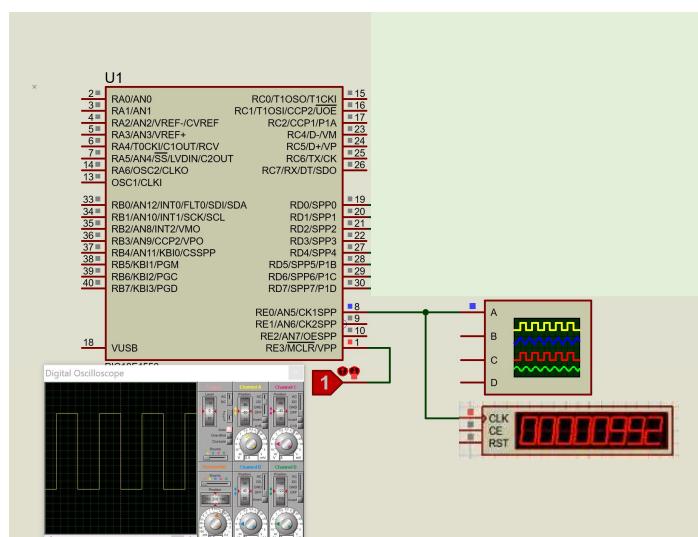
13  #define _XTAL_FREQ 48000000UL //Frecuencia
14
15 [-] void tmr2_conf(void){  
    T2CON = 0x00; // 1:8 Postscale  
    PR2 = 188; // 3D  
}
16 [-] }  
17 [-]   
18 [-] }  
19
20 [-] void main(void){  
    tmr2_conf();  
    ADCON1 = 0x0F;  
    TRISEbits.RE0 = 0;  
    while(1){  
        while(PIR1bits.TMR2IF == 0);  
        LATEbits.LE0 = !LATEbits.LE0;  
        PIR1bits.TMR2IF = 0;  
    }  
}
21 [-]   
22 [-]   
23 [-]   
24 [-]   
25 [-]   
26 [-]   
27 [-]   
28 [-]   
29 [-]

```

J87 ó J86

13

Simulación



14

Mejora: Empleando interrupciones

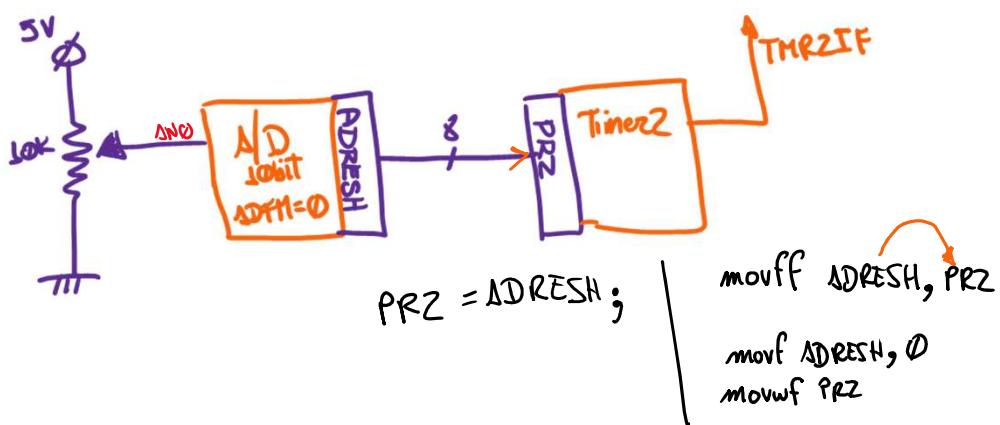
- Tener en cuenta el nuevo formato para la declaración de la función de interrupción en XC8
- La onda cuadra resultante no tiene 1KHz requerido por el enunciado por lo que se deberá modificar el valor de PR2 del Timer2 para ajustar a la frecuencia solicitada.
- Modificando PR2:
 - 187: 998Hz
 - 186: 1003Hz

```

13 #define _XTAL_FREQ 48000000UL //Frecuencia del
14
15 void tmr2_conf(void){
16     T2CON = 0x00;
17     PR2 = 186; // Valor modificado
18 }
19
20 void int_conf(void){
21     INTCONbits.GIE = 1;
22     INTCONbits.PIE1 = 1;
23     PIE1bits.TMR2IE = 1;
24 }
25
26 void main(void){
27     tmr2_conf();
28     int_conf();
29     ADCON1 = 0x0F;
30     TRISEbits.RE0 = 0;
31     while(1);
32 }
33
34 void __interrupt() High_ISR(void){
35     LATBbits.LE0 = !LATBbits.LE0;
36     PIR1bits.TMR2IF = 0;
  
```

15

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



16

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

The circuit diagram shows a PIC18F4550 microcontroller connected to an LM044L operational amplifier. The op-amp is configured as a voltage-controlled voltage source (VCVS) with a 10K resistor from its output to ground. The non-inverting input (pin 13) is connected to an analog-to-digital converter (ADC) input (RA0/AN0). A 10K potentiometer (RV1) is connected between the inverting input (pin 14) and ground. The reference voltage for the ADC is +1.55V. The digital oscilloscope displays a square wave signal on the screen, with the x-axis showing time in ms.

```

#define _XTAL_FREQ 48000000UL

void configuro(void){
    ADCON2 = 0x24; //8TAD FOSC
    ADCON1 = 0x0E; //AN0
    ADCON0 = 0x01; //ADC ON
    T2CON = 0x3D;
    PR2 = 186;
    TRISEbits.RE0 = 0;
    INTCONbits.GIE = 1;
    INTCONbits.PIE1 = 1;
    PIE1bits.TMR2IE = 1;
}

void main(void) {
    configuro();
    while(1){
        ADCON0bits.GODONE = 1; //I
        while(ADCON0bits.GODONE == 1); //E
        PR2 = ADRESH;
    }
}

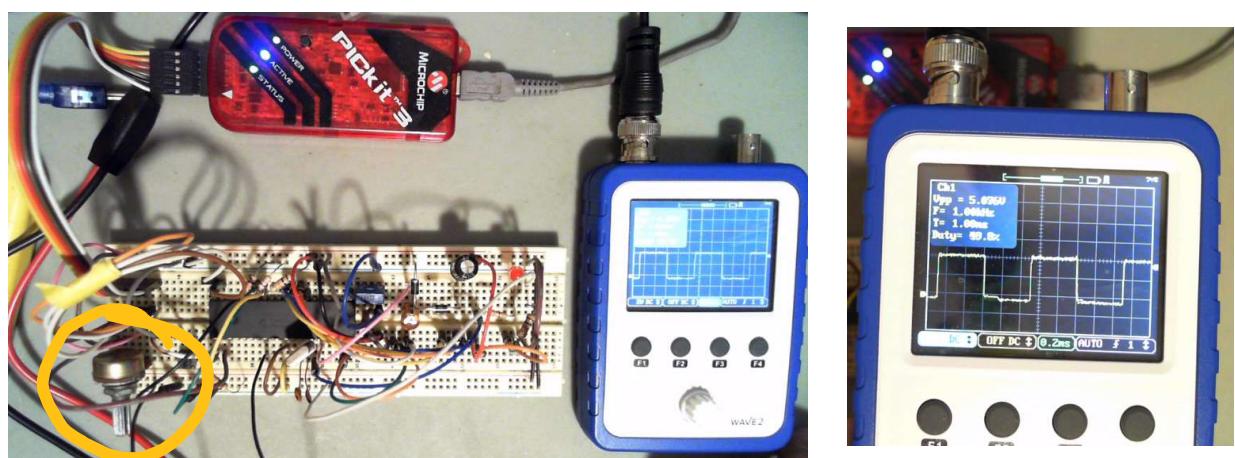
void _interrupt() TMR2_ISR(void){
    // LATEbits.LE0 = !LATEbits.LE0; //bas
    // PIR1bits.TMR2IF = 0; //baj
    asm("btg LATE, 0"); //b
    asm("bcf PIR1, 1"); //b
}

```

- Simulación: moviendo el POT se modificará el periodo de la onda en REO

17

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



18

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.

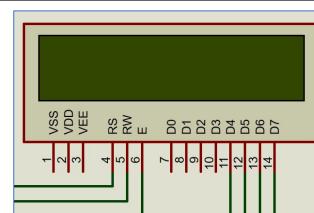
```

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 }
```

19

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 187, con ello la señal cuadrada en RE0 es de 998Hz.



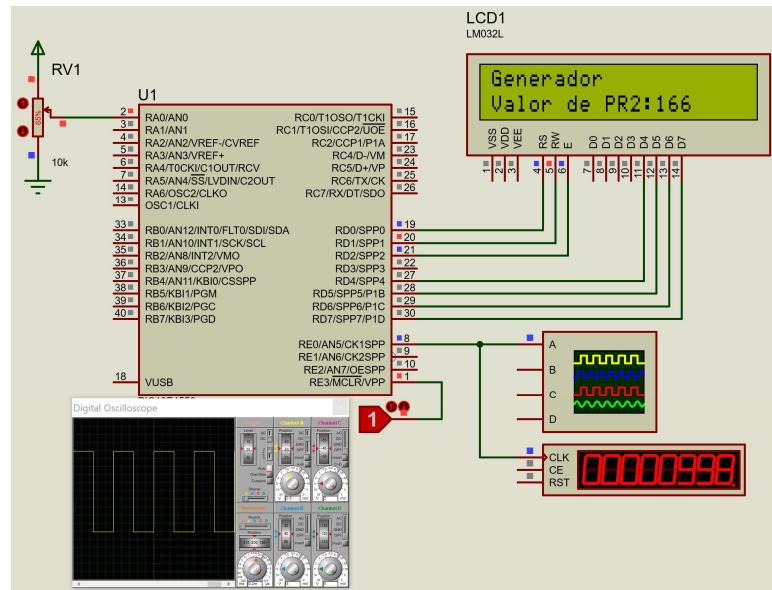
```

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 = 0x00; //D
41     PR2 = 187; //187 & 186
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
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; //Out
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     LATEbits.LE0 = !LATEbits.LE0;
81     PIR1bits.TMR2IF = 0;
82 }
```

20

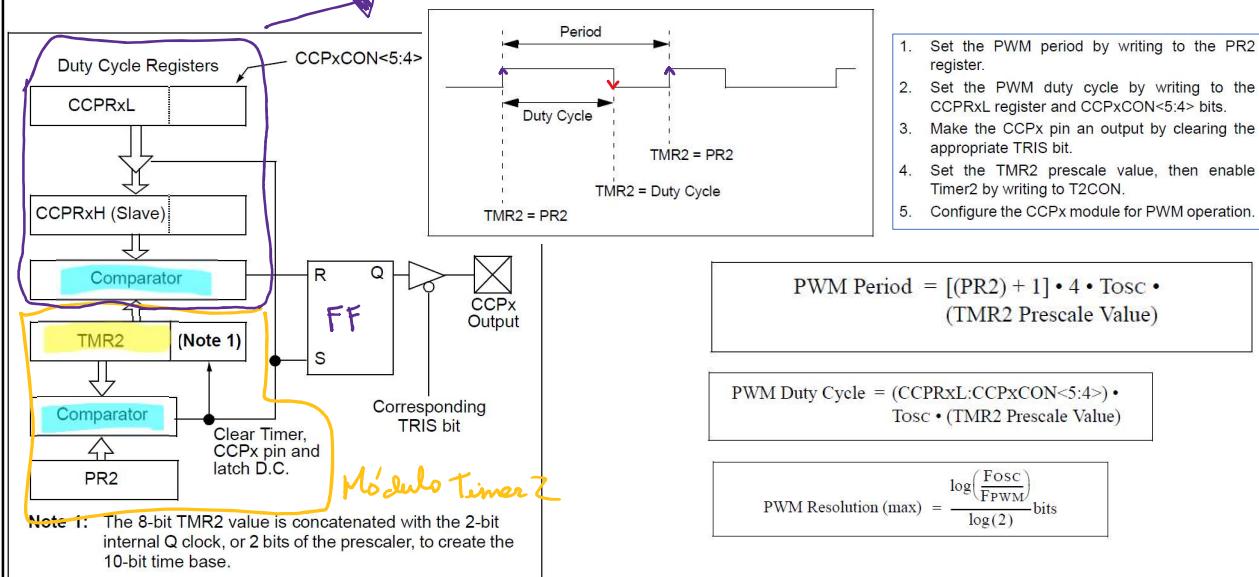
Mejora: Añadimos la funcionalidad del LCD para visualizar el valor que se asigna a PR2

- Simulación



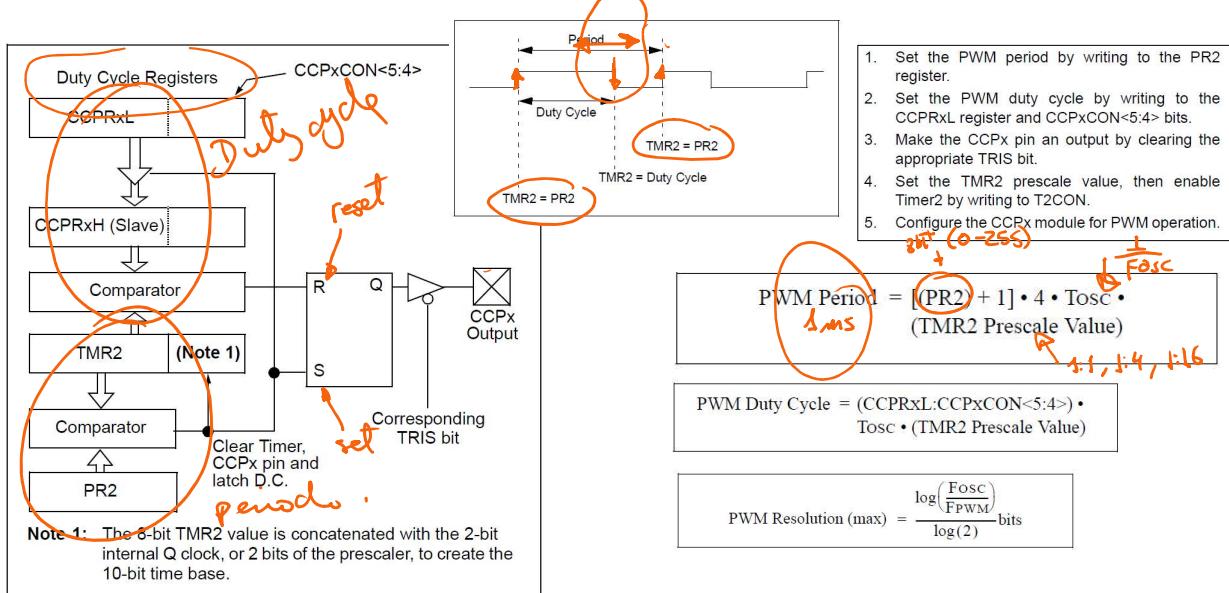
21

PWM con el CCP (extraído de la hoja técnica)



22

PWM con el CCP (extraído de la hoja técnica)



23

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

$$F = \frac{1}{T}$$

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

Conclusion: Con Fosc=48MHz no se puede generar PWM a 2.35KHz

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

$$PR2 = 318.15$$

Registros de 8bit $\Rightarrow 318$ Nooooo

24

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

- Para dar una solución anterior podemos bajar el Fosc a 32MHz (configurando CPUDIV = OSC2_PLL3)
- Trabajando a Fosc = 32MHz obtenemos que PR2 = 212 teniendo en cuenta PSC 1:16 por lo que si se podría generar la PWM solicitada.
- Para DC 50% establecemos CCPR1L = 106

25

Si necesito una onda PWM con freq de 10KHz 50% DC con Fosc 48MHz:

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

1:3, 1:4 o 1:16

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

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

$PR2 = 74$ ⚠️ Si se puede usar

Si DC 50%

Condición

26

Si necesito una onda PWM con frecuencia de 25KHz
30% DC trabajando a Fosc 48MHz:

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

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

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

$$PR2 = 29$$

Condición

⚠ Si se puede usar

Si DC 30%

$$\Rightarrow CCPRL = 9 \text{ ó } 10$$

$$CCPRL = 29$$

$$= 100\%$$

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

27

Ejemplo: Si necesito una onda PWM con frecuencia de 17.6KHz 65%DC a FOSC 32MHz:

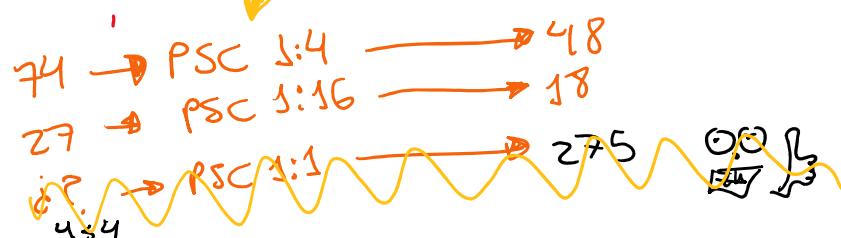
$$\text{PWM Period} = [(PR2 + 1) \cdot 4 \cdot Tosc \cdot (\text{TMR2 Prescale Value})]$$

PR2?

No se puede usar 3:3

Duty Cycle:

CCPRL:



28

Ejemplo con simulación:

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

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

$PR2 = 155$ con PSC 1:16 } Conf. periodo
 $CCPR1L = 78$ ← Conf. duty cycle

29

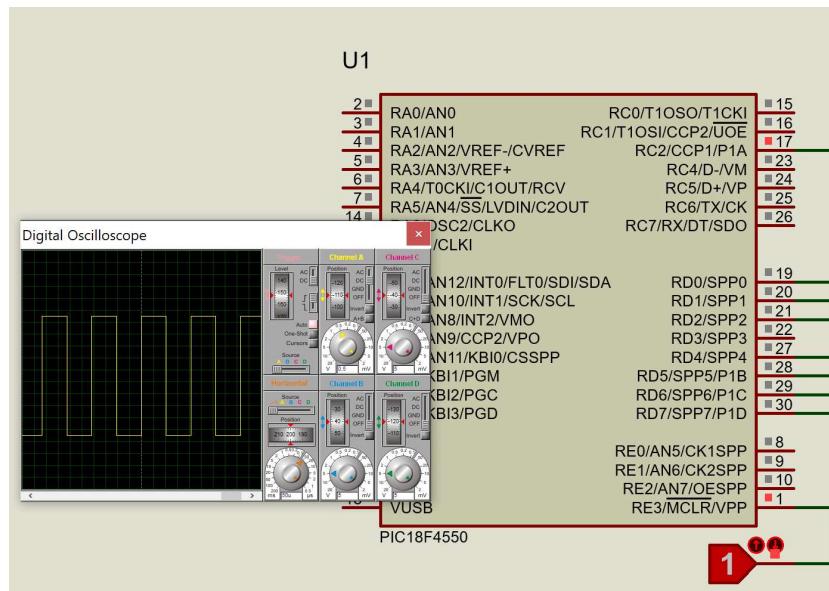
Código en XC8

```

23 #include <xc.h>
24 #define _XTAL_FREQ 48000000UL //Frecuencia de trabajo 48MHz
25
26 void configuracion() {
27     PR2 = 155; //Para el periodo (freq 4.8 KHz en PWM CCP)
28     CCPR1L = 78; //Para el duty cycle (50%)
29     TRISCbbits.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 }
```

30

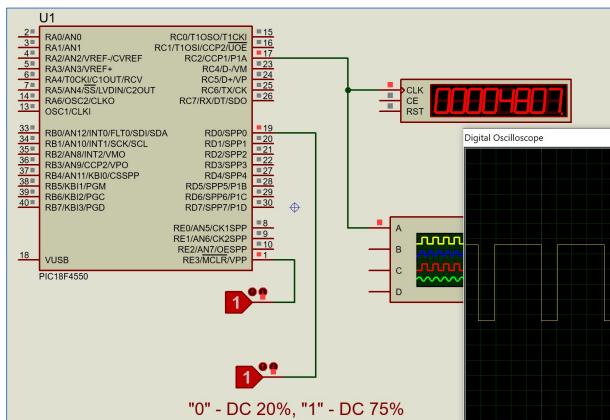
Simulación



31

Mejora del ejemplo anterior: Selección de duty cycle con RD0

- La entrada RD0 cambiará el valor de CCP1L relacionado con el Duty Cycle del PWM

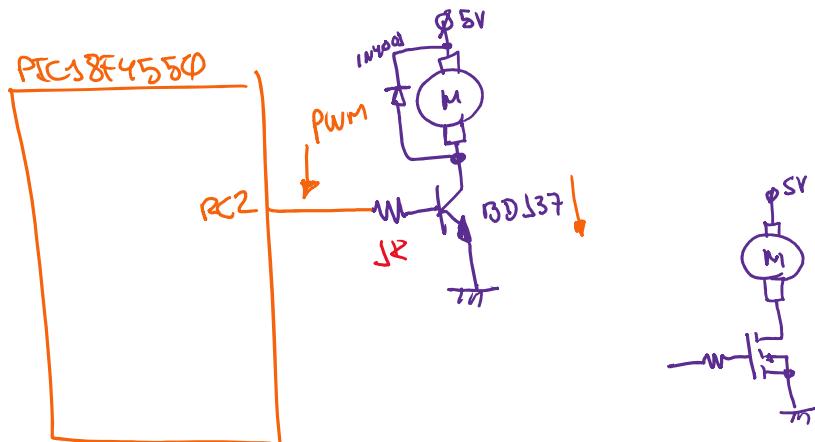


```

1 #include "cabecera.h"
2 #include <xc.h>
3 #define _XTAL_FREQ 48000000UL
4
5 void configuracion(void){
6     PR2 = 155;
7     CCP1L = 31;
8     TRISCbits.RC2 = 0;
9     T2CON = 0x07;
10    CCP1CON = 0x0C;
11 }
12
13 void main(void) {
14     configuracion();
15     while(1){
16         if (PORTDbits.RD0 == 1){
17             CCP1L = 116;
18         }
19         else{
20             CCP1L = 31;
21         }
22     }
23 }
```

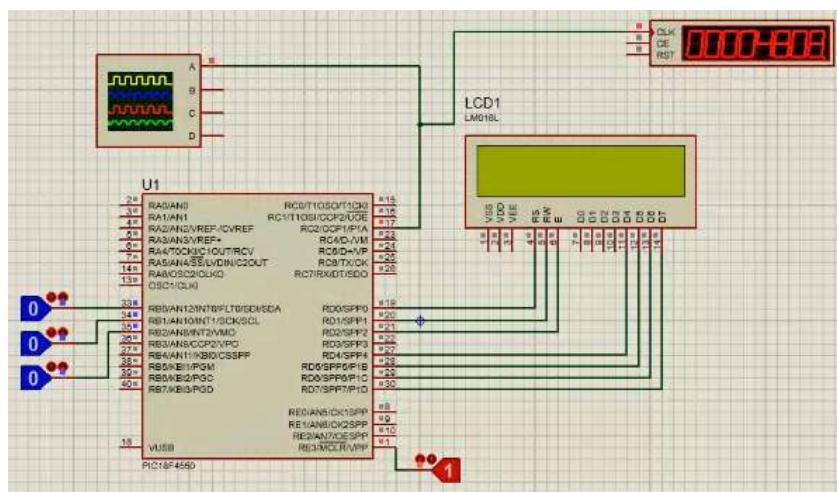
32

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



33

Simulación:



34

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).

