

# Microcontroladores

Semestre: 2022-1

Profesor: Kalun José Lau Gan

Semana 4: Módulo Timer0

1

## ¿Preguntas previas?

- El PC (program counter) solo se mencionó un ejemplo (decodificador de display de 7 segmentos). ¿Hay alguna otra utilidad?
  - El PC es un registro contador incremental el cuál almacena la dirección de memoria de la siguiente instrucción a ejecutar, si modificas dicho registro al término de la ejecución de la instrucción actual se irá a la dirección dada ahí.
  - Utilidad práctica es hacer saltos dentro de la memoria de programa pero no es muy conveniente ya que tenemos instrucciones dedicadas a ello (GOTO, BRA, CALL).
- Acerca del LB1:
  - Los grupos de asignación se asignan de manera aleatoria al inicio de la sesión de laboratorio y se habilitará el acceso del instructivo en PDF en el AV.

2

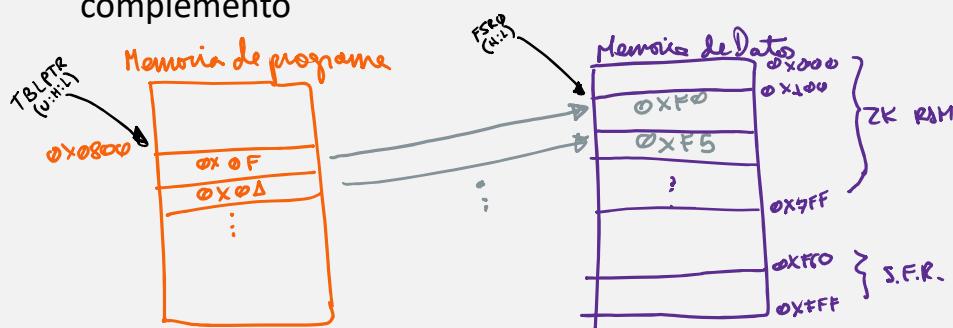
## Preguntas previas:

¿?

3

## Ejemplo sobre manipulación de datos entre memoria de programa y memoria de datos

- Memoria de programa dirección 0x0800
  - 0x0F, 0x0A, 0x08, 0x10, 0xFF, 0x3D, 0x4B, 0x15
- En la memoria de datos en la dirección 0x100 escribir los datos que están en la memoria de programa en dirección 0x0800 pero en complemento



```

inicio:    movlw 0x08
          movwf TBLPTRH
          movlw 0x00
          movwf TBLPTRL
          lfsr 0, 0x100
          movlw .8
          cpfseq TBLPTRL
          goto aunno
          goto yatermine
          TBLRD*
          comf TABLAT, w
          movwf INDFO
          incf TBLPTRL, f
          incf FSROL
          goto loop
yatermine:nop
end
loop:
aunno:
  
```

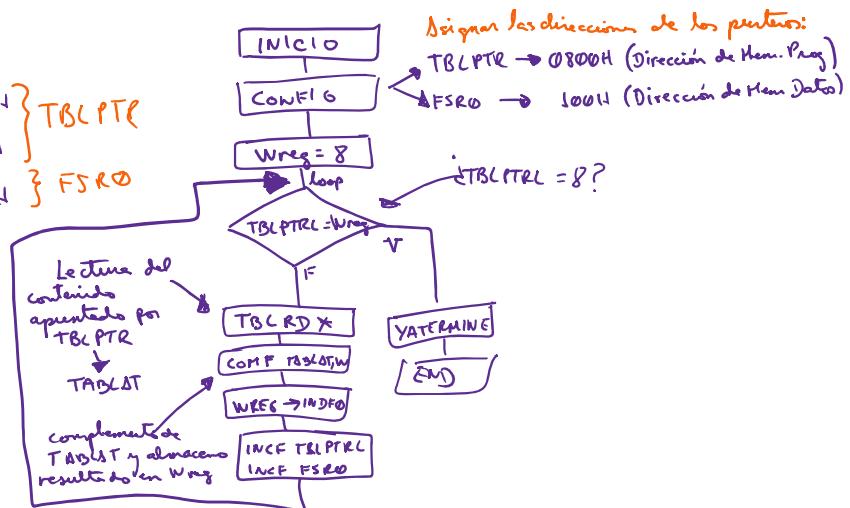
4

## Aplicando ingeniería reversa para obtener el algoritmo en diagrama de flujo a partir del código XC8 PIC Assembler

```

início:    moviw 0x08
            movwf TBLPTRH
            movlw 0x00
            movwf TBLPTRL
            lfsr 0, 0x100
            movlw .8
            cpfseq TBLPTRL
            goto aunno
            goto yatermine
            TBLRD*
            comf TABLAT, w
            movwf INDFO
            incf TBLPTRL, f
            incf FSROL
            goto loop
            yatermine:nop
            end

```



5

## Sobre algoritmos:



6

## Agenda:

- El modulo Timer 0
- Aplicaciones con temporizadores
- Multiplexación de displays de siete segmentos
- Algoritmo para la obtención de los dígitos de centena, decena y unidad de un registro.

7

## El Timer 0

- Información extraída del datasheet del PIC18F4550
- Diagrama de bloques:

FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)

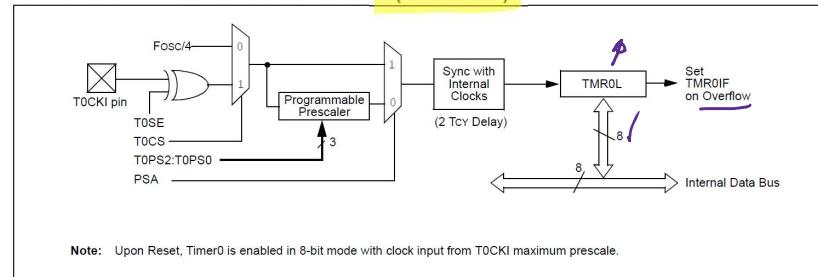
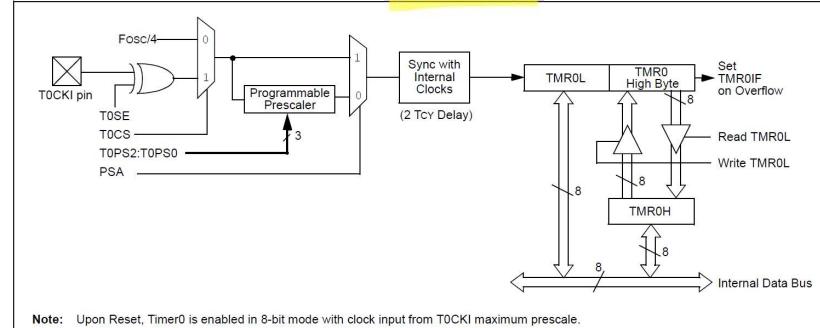


FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



8

## El módulo Timer 0

- (Ref. Item 11 de la hoja técnica del microcontrolador PIC18F4550)
- Temporizador de cuenta ascendente
- Resolución 8 bits (0-255) ó **16 bits (0-65535)**
- Las cuentas del Timer0 se alojan en:
  - TMROH:TMROL (16 bits)
  - TMROL (8 bits)
- **Tener en consideración el procedimiento estricto sobre el tratamiento de la cuenta en modo 16 bits.**
- Diversas fuentes de reloj: interno (FOSC/4) o externo (pin6 TOCKI)
- Divisor de frecuencia al reloj de entrada (1:2 – 1:256)
- El desborde se produce cuando la cuenta esta en el valor mas alto y se recibe un pulso de reloj, ocasionando que la cuenta pase a 0 y levantándose la bandera de desborde (TMROIF=1)
- Al activarse TMROIF=1 se debe de bajar manualmente la bandera para que se pueda detectar un nuevo desborde (simplemente haciendo bcf INTCON, 2, siendo el bit 2 el TMROIF).
- Al desbordarse puede emitir interrupción (TMROIF = 1), revisar interrupciones y sus 10 registros implicados
- Se usa el registro T0CON (SFR 0FD5H) para configurar el Timer0 (por defecto T0CON=0FFH)

9

## Timer 0 – Modos de trabajo

- Modo temporizador (reloj interno para la cuenta: FOSC/4)
  - Ej. Generador de ondas cuadradas, base de tiempo para la multiplexación de los displays de siete segmentos, cuentas regresivas cortas, efectos de desplazamiento en displays, LED blinkers, etc
  - **No se usa para aplicaciones en tiempo real (relojes, cronómetros)**
- Modo contador (empleando pin externo TOCKI para la cuenta)
  - Ej. Velocímetro para bicicleta, medidor de RPMs de un motor

10

## El Timer0 – Registro T0CON

- Tener en cuenta que los valores por defecto en un PoR son '1' en cada bit del registro
- TOSE es ignorado si TOCS = 0

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	TOCS	TOSE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0
<b>Legend:</b>							
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	<b>TMR0ON:</b> Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0						
bit 6	<b>T08BIT:</b> 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	<b>TOCS:</b> Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKO)						
bit 4	<b>TOSE:</b> Timer0 Source Edge Select bit 1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin						
bit 3	<b>PSA:</b> 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	<b>T0PS2:T0PS0:</b> 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						

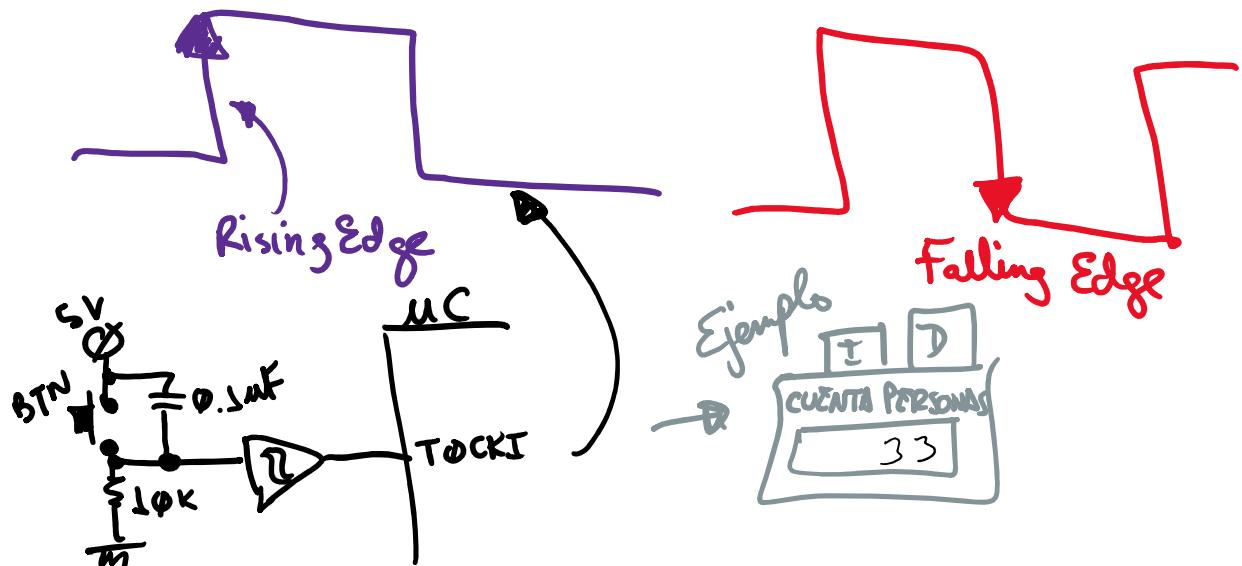
11

## Procedimiento para ingresar una cuenta inicial al Timer 0 en modo 16 bits

1. Si por ejemplo se quiere ingresar el número 5536 como cuenta inicial, convertirlo a hexadecimal (DEC 5536 = HEX 15A0H)
2. Se ingresa el dato de 8 bit mas significativo a TMR0H, en el ejemplo 15H hacia TMR0H.
3. Se ingresa el dato de 8 bit menos significativo a TMR0L, haciendo esto se sube en simultáneo el TMR0H al registro de cuentas del Timer0, en el ejemplo 0A0H hacia TMR0L.
4. Recordar que luego del desborde se deberá ingresar nuevamente la cuenta inicial para preservar el temporizado de manera continua.
5. No olvidar que luego de un desborde se levanta la bandera y ésta hay que bajarla manualmente para que el módulo pueda activarla ante un nuevo desborde

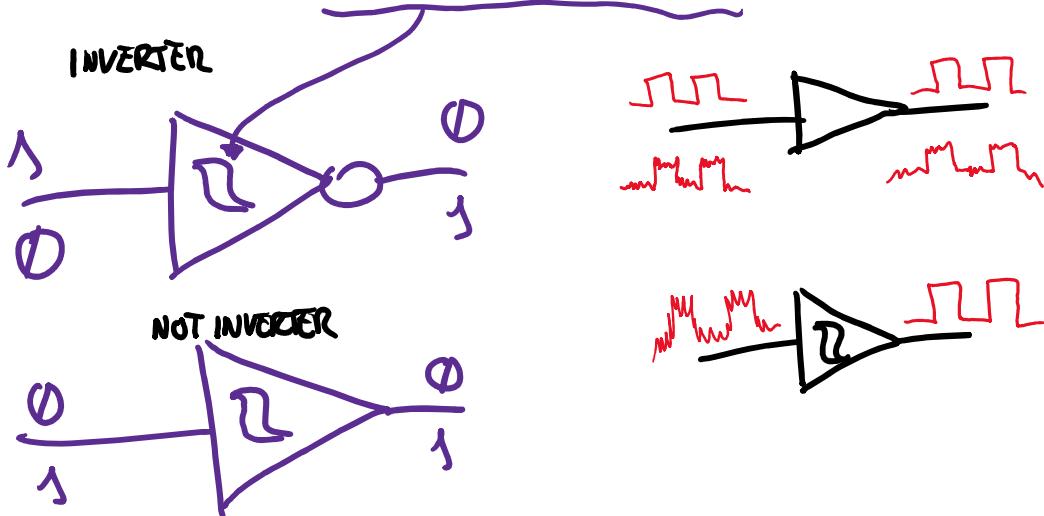
12

## Recordando Rising Edge vs Falling Edge



13

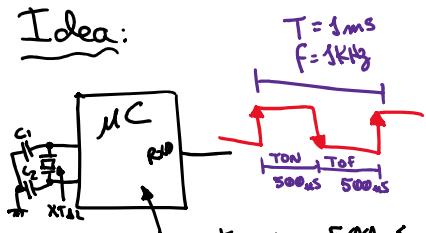
## Recordando el SCHMITT TRIGGER



14

Ejemplo: Generar una onda cuadrada de 1KHz DC 50 % empleando el Timer0 modo 8 bits:

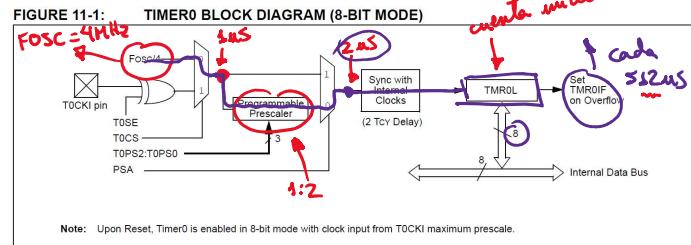
Idea:



Tiene que temporizar 500μs para que bastele RX0

Lo va a hacer el Timer0

¿Cómo hago para que el Timer0 temporice 500μs?



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

cuenta inicial = 6 cada 500μs

Resumen:

1: Modo 8 bit - temporizador ( $F_{osc}/4$ )

2: Prescaler 1:2

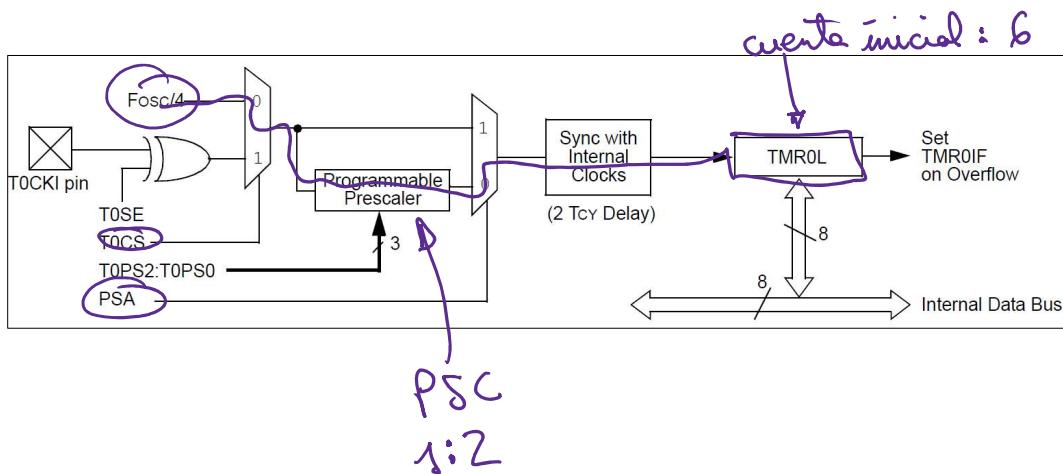
3: Cuenta inicial de 6 }  $\rightarrow TMR0L$

$\Rightarrow T0CON = \emptyset \times C0$

$C0H$

15

Continuación...



16

## Configurar el TOCON:

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-4	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	TOPS2	T0PS1	0PS0
bit 7							bit 0

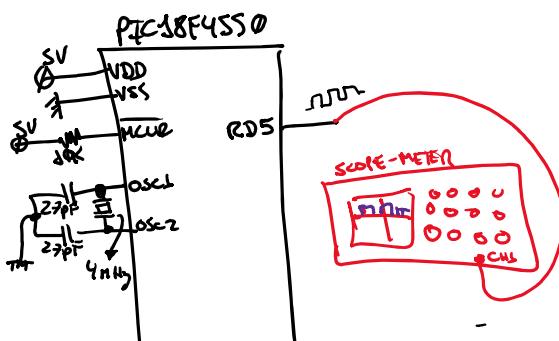
<b>Legend:</b>	
R = Readable bit	W = Writable bit
-n = Value at POR	'1' = Bit is set
	'0' = Bit is cleared
	X = Bit is unknown

- 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 → **T0CS:** Timer0 Clock Source Select bit  
1 = Transition on T0CKI pin  
0 = Internal instruction cycle clock (CLKO) *(Fosc/4)*
- bit 4 → **T0SE:** Timer0 Source Edge Select 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

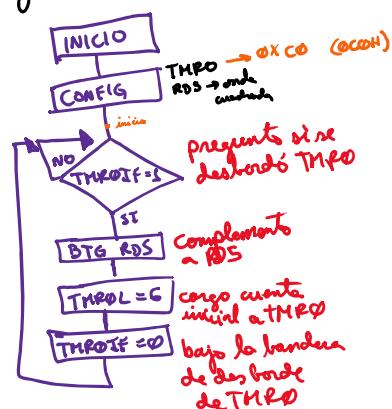
17

Ejemplo: Generar una onda cuadrada de 1KHz empleando el Timer0 modo 8 bits:

Circuito:

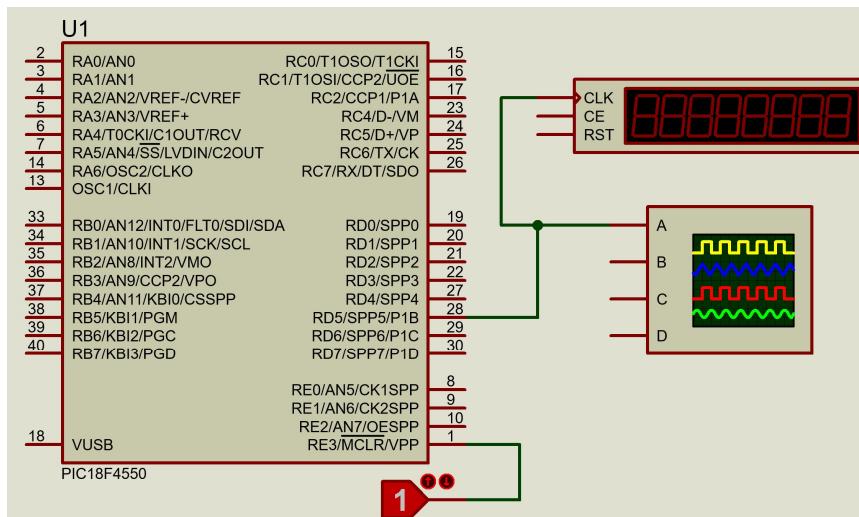


Algoritmo:



18

## Hardware



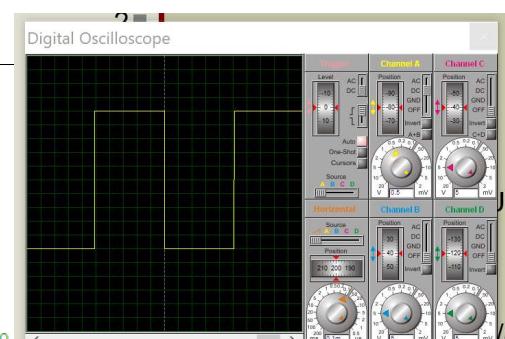
19

## Propuesta de código en MPASM de lo anterior

```

17      org 0x0000
18      goto init_conf
19
20      org 0x0020
21      init_conf:
22          movlw 0xC0
23          movwf T0CON      ;Timer0 ON 8bit, FOSC/4, 1:2PSC
24          bcf TRISD, 5    ;RD5 como salida
25      loop:
26          btfs INTCON, TMR0IF    ;Pregunto si se desbordó TMR0
27          goto loop           ;No se desbordó
28          btg LATD, 5         ;Basculo RD5
29          movlw .6
30          movwf TMR0L        ;Cargo valor inicial de 6 a TMR0
31          bcf INTCON, TMR0IF ;Bajamos la bandera de desborde de TMR0
32          goto loop
33          end

```



20

## Propuesta de código en XC8 PIC ASM de lo anterior

```

1  PROCESSOR 18F4550
2  #include "cabecera.inc"
3
4  PSECT principal, class=CODE, reloc=2, abs
5
6  principal:
7      ORG 0000H
8      goto configuro
9      ORG 0020H
10
11 configuro:
12     bcf TRISD, 5           ;RD5 como salida
13     movlw 0COH             ;Timer0 ON, modo 8bit, 1:2PSC, FOSC/4
14     movwf TOCON
15
16 loop:   movlw 6
17     movwf TMROL            ;Cuenta inicial de 6
18     otro:   btfs INTCON, 2 ;Pregunta si TMR0IF=1
19     goto otro
20     btg LATD, 5            ;Basculamos RD5 para que genere la onda cuadrada
21     bcf INTCON, 2          ;Bajando la bandera TMR0IF
22     goto loop
23
24 end principal

```



21

## Observaciones

- No sale exacto en las pruebas, esto debido a que no se ha contemplado el tiempo en que se demora en ejecutar las instrucciones
- Se tiene que hacer una compensación haciendo que el TMR0 cuente menos cuentas (periodo menor de temporizado) y se ajusta empleando nops
- Haciendo este tipo de compensaciones nos acercaremos a la frecuencia solicitada pero no será exacto.
- Esto nos hace pensar que no se va a poder hacer sistemas en tiempo real.

22

## Código en XC8 PIC Assembler con compensación de tiempo para obtener la señal cuadrada de 1KHz:

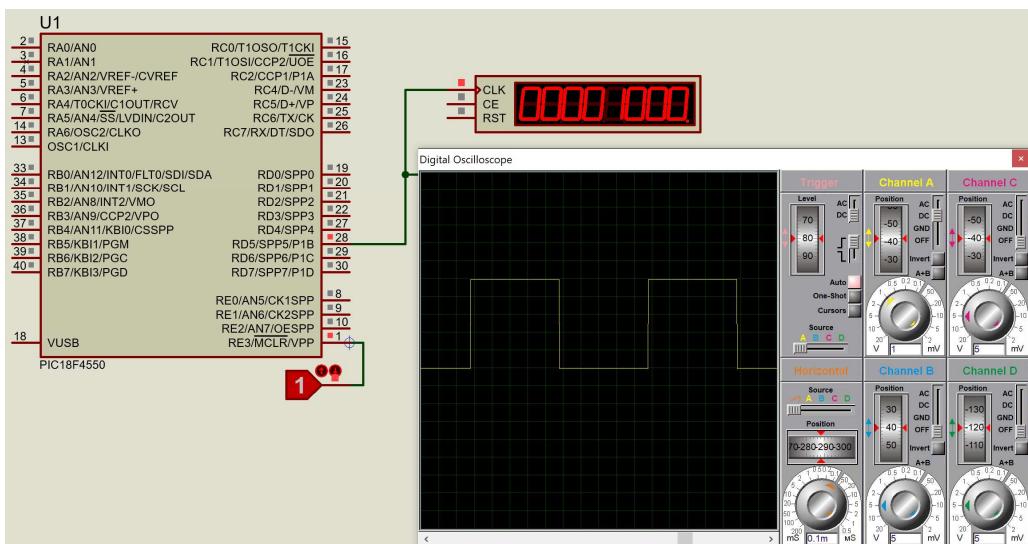
```

1      PROCESSOR 18F4550
2      #include "cabecera.inc"
3
4      PSECT mastercode,class=CODE,reloc=2,abs
5      ORG 00000H      ;Vector de reset
6      mastercode: goto configuracion
7
8      ORG 00020H      ;Zona de programa de usuario
9      configuracion:
10     movlw 0C0H
11     movwf TOCON      ;Tmr0 FOSC/4, modo 8 bits, psc 1:2
12     bcf TRISD, 5    ;RD5 como salida
13     inicio:
14     btfs INTCON, 2  ;Pregunto si TMR0 se desbordó
15     goto inicio
16     btg LATD, 5     ;Basculo RD5
17     movlw 11
18     nop
19     nop
20     movwf TMROL      ;Cargo cuenta inicial de 11 a TMR0
21     bcf INTCON, 2  ;Bajo la bandera TMR0IF
22     goto inicio
23
24     END mastercode

```

23

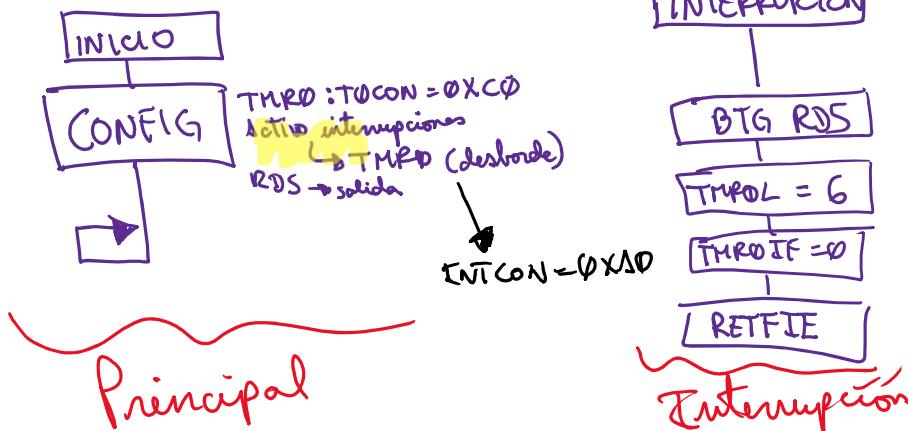
## Simulación en Proteus con el código en XC8 PIC Assembler (compensación añadida)



24

# Mejora del generador de onda cuadrada de 1KHz empleando interrupciones

Diagramas de flujo:



25

Código en MPASM del ejemplo pero ahora con TMRO\_ISR

```

17      org 0x0000          ;Vector de RESET
18      goto init_conf
19
20      org 0x0008          ;Vector de interrupcion
21      goto TMRO_ISR
22
23      org 0x0020
24      init_conf:
25          movlw 0xC0
26          movwf TOCON          ;Timer0 ON 8bit, FOSC/4, 1:2PSC
27          movlw 0xA0
28          movwf INTCON         ;Activamos la interrupcion de desborde de TMRO
29          bcf TRISD, 5         ;RDS como salida
30      loop:
31          goto loop
32
33      TMRO_ISR:
34          btg LATD, 5          ;Basculo RDS
35          movlw .12
36          movwf TMROL          ;Cargo valor inicial de 6 a TMRO
37          bcf INTCON, TMROIF   ;Bajamos la bandera de desborde de TMRO
38          retfie
39          end

```

26

## Portado de código anterior a XC8 PIC Assembler:

```

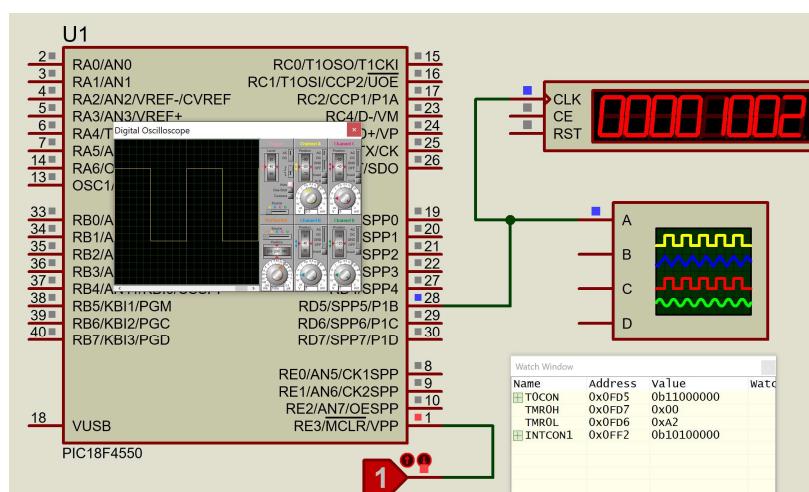
1      PROCESSOR 18F4550
2      #include "cabecera.inc"
3
4      PSECT maincode, class=CODE, reloc=2, abs
5      ORG 00000H
6      maincode:
7          goto configuro
8
9      ORG 00008H
10     goto TMR0_ISR
11
12     ORG 00020H
13     configuro:
14         bcf TRISD, 5           ;RD5 sea salida
15         movlw 0C0H
16         movwf TOCON           ;Timer0 ON, 8BIT, FOSC/4, 1:2PS
17         movlw 0A0H
18         movwf INTCON          ;Habilitando la interrupcion de desborde de TMR0
19
20     inicio:    nop
21         goto inicio
22
23     TMR0_ISR: btg LATD, 5       ;Basculacion en RD5
24         nop
25         movlw 12
26         movwf TMROL           ;Carga de cuenta inicial de 6 a TMRO
27         bcf INTCON, 2          ;Bajamos la bandera de overflow del TMRO
28         retfie
29
30     end maincode           ;NOTA: Hay que tomar en cuenta el tiempo de ejecución
31

```

27

## Simulación

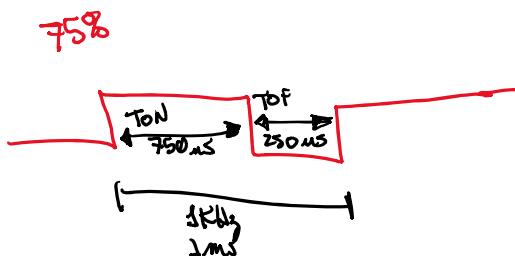
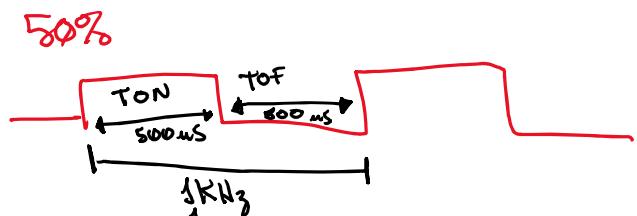
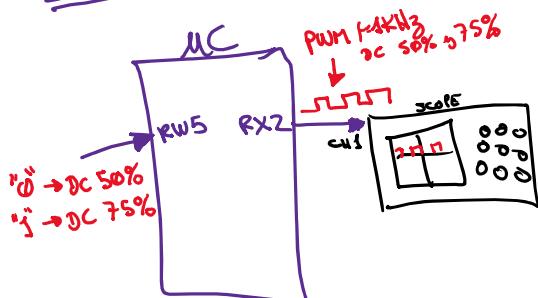
El CPU del microcontrolador esta en standby mientras que el Timer0 se encarga de hacer la generación de la señal cuadrada. Adicionalmente se deberá de compensar para obtener una mejor precisión en frecuencia.



28

Ejemplo: Desarrollar un generador de PWM con frecuencia 1KHz y con dos opciones de Duty Cycle 50% y 75%.

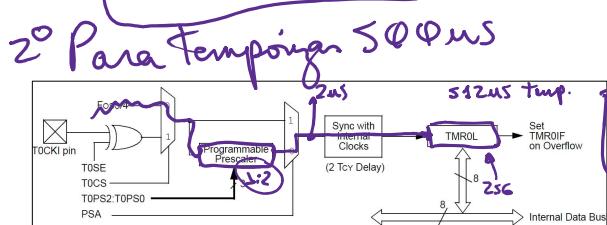
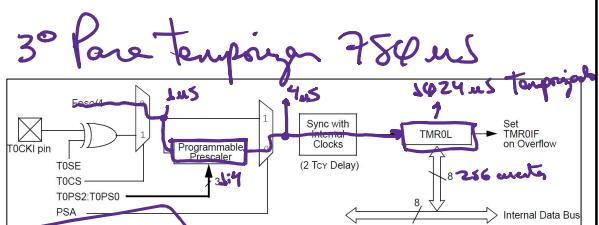
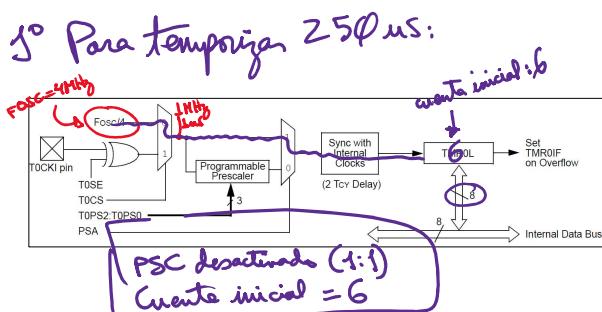
Idea:



Nota: Debemos de configurar tres temporizaciones en el Timer0:  
250μs, 500μs y 750μs.

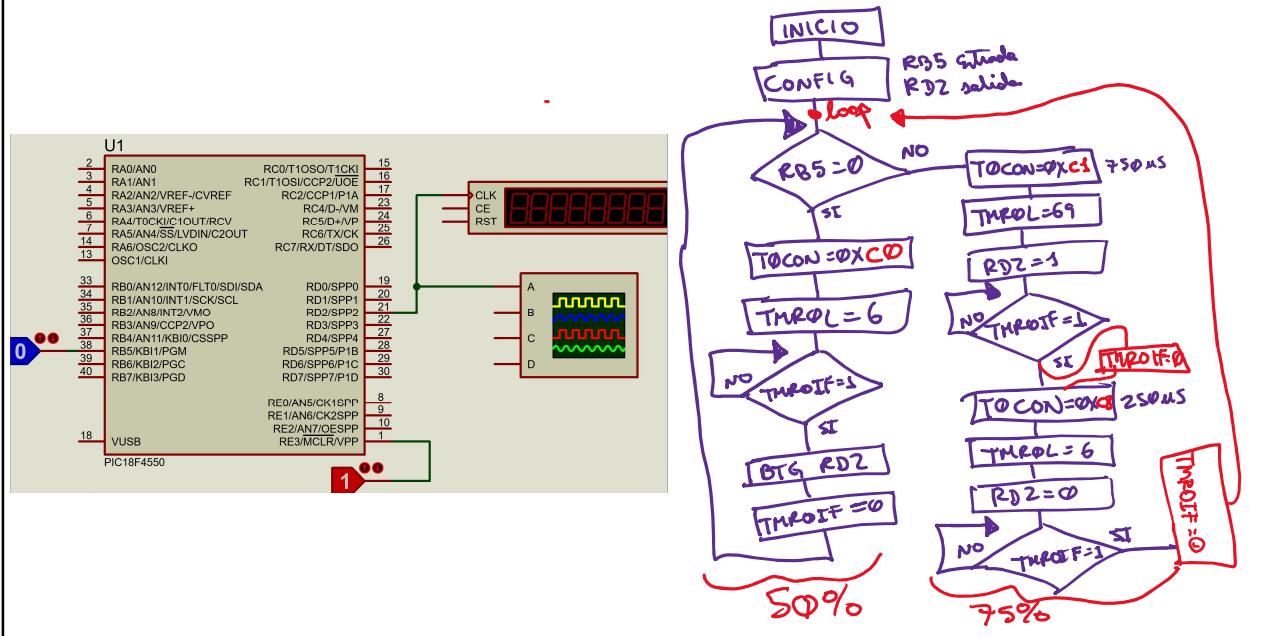
29

Cont. del ejemplo: Configuración del Timer0 según tiempos de temporización requeridos



30

## Cont. del ejemplo: Hardware y desarrollo del diagrama de flujo



31

## Cont. del ejemplo: código en MPASM

```

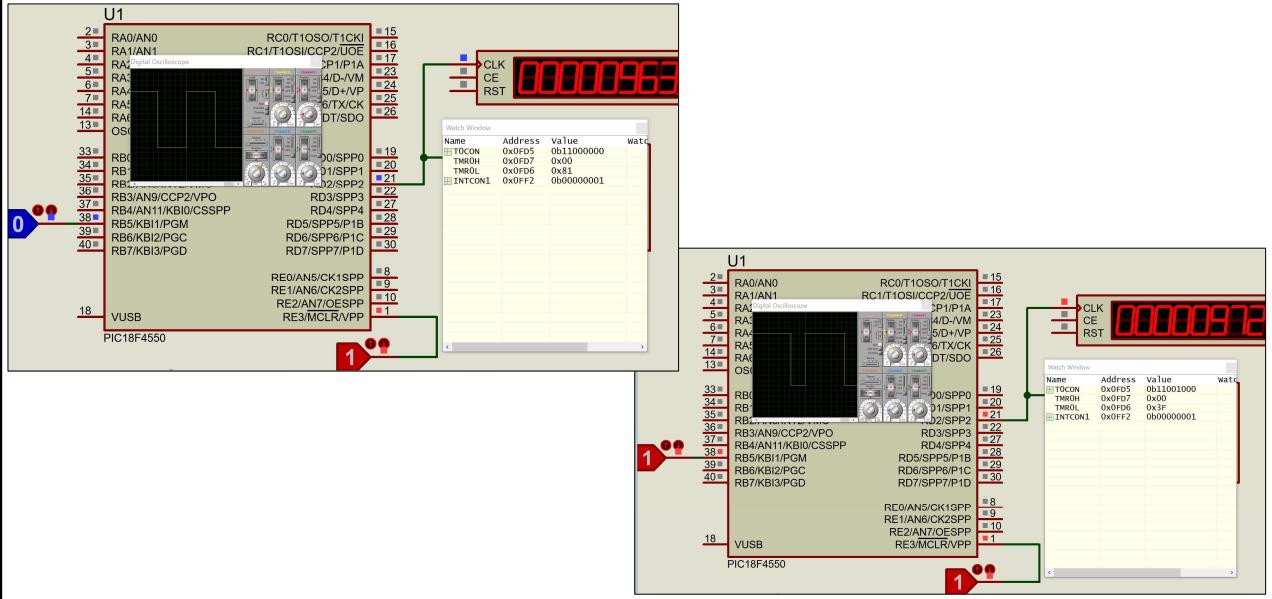
17     org 0x0000
18     goto init_conf
19
20     org 0x0020
21     init_conf:
22         bcf TRISD, 2
23     loop:
24         btfsc PORTB, 5
25         goto setentaycinco
26         movlw 0xC0
27         movwf T0CON
28         movlw .6
29         movwf TMR0L
30     otrol:
31         btfss INTCON, TMROIF
32         goto otrol
33         btf LATD, 2
34         bcf INTCON, TMROIF
35         goto loop
36     setentaycinco:
37         movlw 0xC1
38         movwf T0CON
39         movlw .69
40         movwf TMR0L
41         bsf LATD, 2
42     otro2:
43         btfss INTCON, TMROIF
44         goto otro2
45         bcf INTCON, TMROIF
46         movlw 0xC8
47         movwf T0CON
48         movlw .6
49         movwf TMR0L
50         bcf LATD, 2
51     otro3:
52         btfss INTCON, TMROIF
53         goto otro3
54         bcf INTCON, TMROIF
55         goto loop
56         end

```

**Asignación:** Portar el código mostrado hacia XC8 PIC Assembler

32

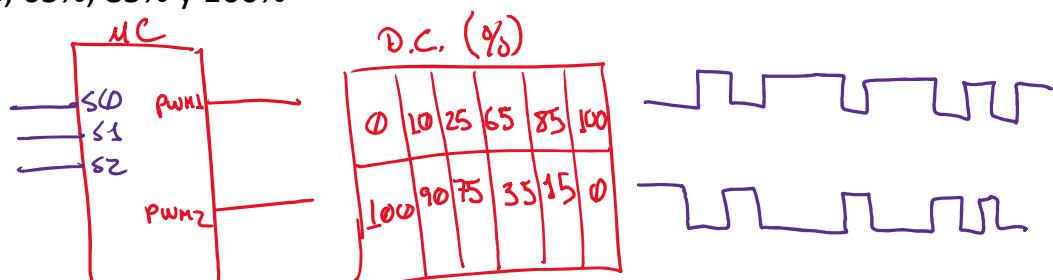
## Cont. del ejemplo: Simulación



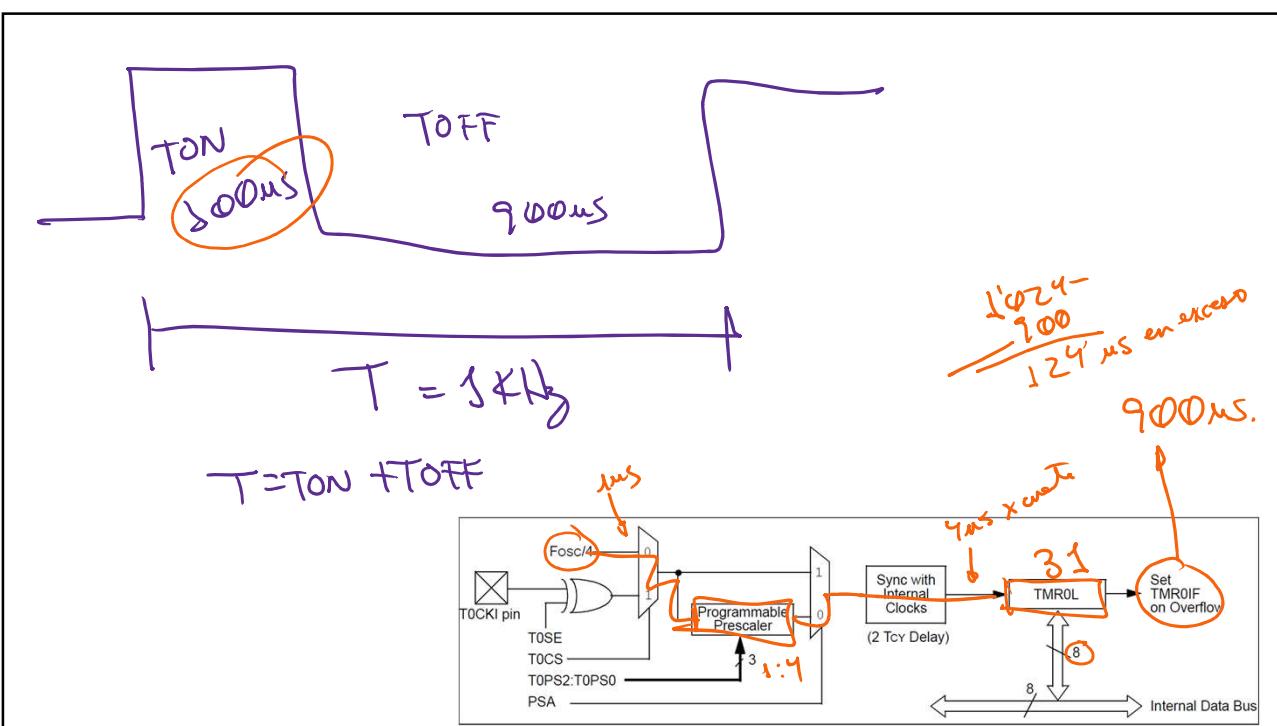
33

## Fin de la sesión

- Links adicionales:
  - Microchip Timer0 Tutorial part1: <http://ww1.microchip.com/downloads/en/devicedoc/51682a.pdf>
  - Microchip Timer0 Tutorial part2: <http://ww1.microchip.com/downloads/en/DeviceDoc/51702a.pdf>
- Ejercicio: Desarrollar un generador de PWM 2KHz con dos salidas complementarias y con opciones de dutycycle siguientes: 0%, 10%, 25%, 65%, 85% y 100%



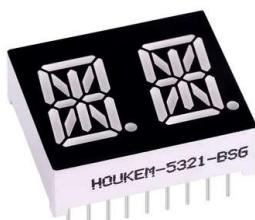
34



35

## Cuestionario:

- En un PoR. ¿En qué estado se encuentra el Timer0, encendido o apagado?
- Si Fosc = 24MHz. ¿Cuál es la temporización máxima del Timer0 en modo 16 bits?
- ¿Qué es lo que hace la instrucción BTG?
- Si Fosc = 12MHz. ¿Cuánto se demorará en ejecutar la instrucción CPFSGT?
- Hacer un circuito de conexión entre el microcontrolador PIC18F4550 y el siguiente dispositivo:



36