

Microcontroladores

Semana 11

Por Kalun José Lau Gan

1

¿Preguntas previas?

- Puede explicarnos mas sobre interrupciones vectorizadas en el XC8 con el Curiosity Nano?
 - Hoy atenderemos este punto nuevamente
- ¿Cómo puedo construir un reloj con el Curiosity Nano?
 - Empleando el Timer1 ó algún módulo RTC externo (DS1307, DS3231 ó similares)
- ¿MPLAB Xpress puede correr fuera de una PC?
 - Como se evidenció la semana pasada, MPLAB Xpress al ser una plataforma que corre en un navegador web, se puede emplear cualquier máquina que tenga ello (un web browser), pudiendo ser una PC desktop corriendo Windows, Linux, MacOS, Android. Hasta desde un celular puedes desarrollar una aplicación con el Curiosity Nano!

2

Preguntas previas

- ¿Veremos manipulación de servos?
 - Si, hoy atenderemos ese tema
- ¿Cuáles son los temas pendientes en el curso?
 - Comunicación serial I2C
 - Comunicación serial UART
 - Protocolos e interfaces propietarias (sensores DHT11, 1-wire)
 - Otras interfaces y protocolos (RS485, CAN)
 - PCB
 - Modelado 3D y manufactura aditiva
 - Micros de 32bits (ESP32, Rpi-Pico, STM32, PIC32)
- ¿Cuáles son las evaluaciones pendientes en el curso?
 - Según el sílabo: LB3 (Sem12), PC2(Sem14), DD(Sem15) y TF(Sem16)

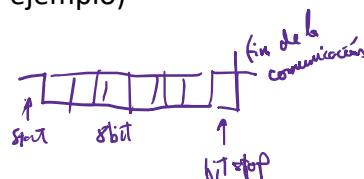
3

Preguntas previas

- ¿Cómo se transmiten los datos en el UART?
 - Usando el registro U1TXB (U1TXB por ejemplo)

```

U1TXB = 'H';
while (U1ERRR&bitc.TXMTIF == 0);
U1TXB = 'O';
while (U1ERRR&bitc.TXMTIF == 0);
U1TXB = 'L';
while (U1ERRR&bitc.TXMTIF == 0);
U1TXB = 'A';
while (U1ERRR&bitc.TXMTIF == 0);
  
```



```

unsigned int bateria = 16399;
// determinar: primero H y luego L
U1TXB = (bateria >> 8) & 0x00FF;
while (U1ERRR&bitc.TXMTIF == 0);
U1TXB = bateria & 0x00FF;
while (U1ERRR&bitc.TXMTIF == 0);
  
```

4

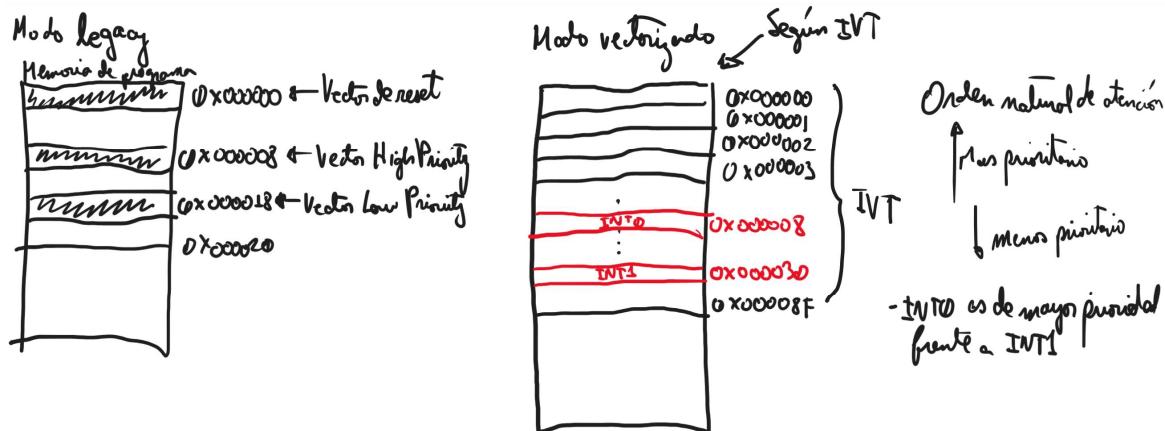
Agenda:

- Resumen de interrupciones vectorizadas
- El Timer1
- Aplicación de reloj en tiempo real con Timer1
- El servomecanismo
- Aplicación de replicación de movimiento entre un potenciómetro y un servomecanismo

5

Interrupciones vectorizadas a detalle

- Hay una tabla de interrupciones vectorizadas (IVT)
- Cada fuente de interrupción que posee el microcontrolador se encuentra en una dirección en particular en la IVT



6

Interrupciones vectorizadas a detalle

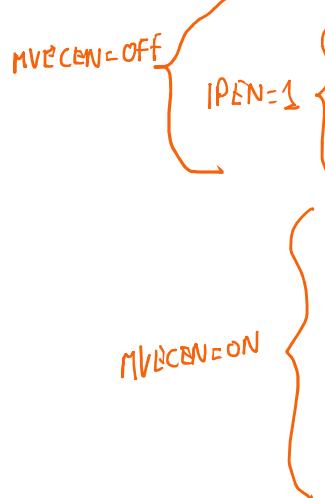
- La IVT:
- La atención de las interrupciones en el orden natural: menor valor de dirección mayor prioridad.

Vector Number	Interrupt source	Vector Number (cont.)	Interrupt source (cont.)
0x0	Software Interrupt	0x3E	PWM3RINT
0x1	HUV/D (High/Low-Voltage Detect)	0x3F	PWM3GINT
0x2	OSF (Oscillator Fail)	0x40	U2RX
0x3	CSW (Clock Switching)	0x41	U2TX
0x4		0x42	U2E
0x5	CLC1 (Configurable Logic Cell)	0x43	U2
0x6		0x44	TMR5
0x7	IOC (Interrupt-On-Change)	0x45	TMR5G
0x8	INT0	0x46	CCP2
0x9	ZCD (Zero Crossing Detection)	0x47	SCAN
0xA	AD (ADC Conversion Complete)	0x48	U3RX
0xB	ACT (Active Clock Tuning)	0x49	U3TX
0xC	CMI (Comparator)	0x4A	U3E
0xD	SMT1 (Signal Measurement Timer)	0x4B	U3
0xE	SMT1PRA	0x4C	-
0xF	SMT1PWA	0x4D	CLC4
0x10	ADT	0x4E - 0x4F	-
0x11 - 0x13	DMA1SCNT (Direct Memory Access)	0x51	INT2
0x15	DMA1DCNT	0x52	CLC5
0x16	DMA1OR	0x53	CWG2 (Complementary Waveform Generator)
0x17	DMA1TA	0x54	NC02
0x18	SP11RX (Serial Peripheral Interface)	0x55	DMA2SCNT
0x19	SP11TX	0x56	DMA3OR
0x1A	SP11	0x57	DMA3A
0x1B	TMR2	0x58	CCP3
0x1C	TMR1	0x59	CLC6
0x1D	TMR0	0x5A	DN303
0x1E	CCP1 (Capture/Compare/PWM)	0x5B	TMR4
0x1F	TMR0	0x5C	DM4ASCNT
0x20	UIRX	0x5D	DM4ACNT
0x21	UITX	0x5E	DM4AR
0x22	UI	0x5F	DM4AA
0x23	U1	0x60	U4RX
0x24 - 0x25	-	0x61	U4TX
0x26	PWM1BRINT	0x62	U4E
0x27	PWM1GINT	0x63	U4
0x28	SP10RX	0x64	DM4SCNT
0x29	SP10TX	0x65	DM4SCNT
0x2A	SP10	0x66	DM4S0R
0x2B	-	0x67	DM4SA
0x2C	TMR3	0x68	U5RX
0x2D	TMR2	0x69	U5TX
0x2E	PWM0RINT	0x6A	U5E
0x2F	PWM2GINT	0x6B	U5
0x30	INT1	0x6C	DM6SCNT
0x31	CLC2	0x6D	DM6DCNT
0x32	CWG1 (Complementary Waveform Generator)	0x6E	DM60R
0x33	NC01 (Numerically Controlled Oscillator)	0x6F	DM6A
0x34	DMA3SCNT	0x70	-
0x35	DMA2DCNT	0x71	CLC7
0x36	DMA2OR	0x72	CM2
0x37	DMA2A	0x73	NC03
0x38	I2C1RX	0x74 - 0x77	-
0x39	I2C1TX	0x78	NVM
0x3A	I2C1	0x79	CLC8
0x3B	I2C1E	0x7A	CRC (Cyclic Redundancy Check)
0x3C	-	0x7B	TMR6
0x3D	CLC3	0x7C - 0x8F	-

7

Interrupciones vectorizadas a detalle

- Plantilla de funciones de interrupciones:



```

//----- un solo fuente de interrupcion
void __interrupt() INT_ISR(void){
//-
}

//----- con prioridades en modo legacy
void __interrupt(high_priority) INT_HP_ISR(void){
//-
}
08H

void __interrupt(low_priority) INT_LP_ISR(void){
//-
}
LP

//----- con prioridades de orden natural segun IVT
void __interrupt(irq(IRQ_INT0) INT0_ISR(void){
//-
}

void __interrupt(irq(IRQ_INT1) INT1_ISR(void){
//-
}

void __interrupt(irq(IRQ_INT2) INT2_ISR(void){
//-
}

void __interrupt(irq(IRQ_TMR0) TMR0_ISR(void){
//-
}

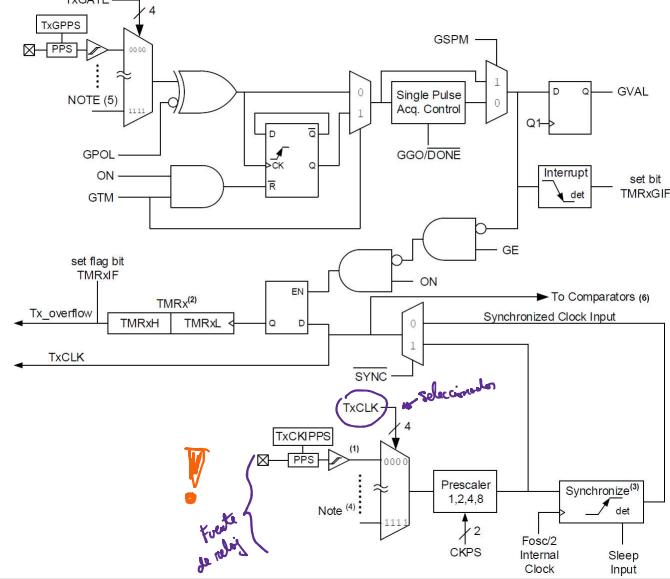
```

8

El Timer 1 en el PIC18F57Q43

- Módulo temporizador de 16 bits
- Empleado para aplicaciones de RTC ✓
- Función de gate
- Funciona con el CCP1 (modo captura ó modo comparación)
- Múltiples fuentes de reloj

Figure 25-1. Timer1 Block Diagram



9

El Timer 1 en el PIC18F57Q43

- Selección de la fuente de reloj (registro T1CLK)

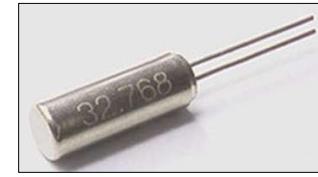


Timer Clock Source Selection Register									
Bit	7	6	5	4	3	2	CS[4:0]	1	0
Access				R/W	R/W	R/W	CS[4:0]	R/W	R/W
Reset				0	0	0		0	0
Bits 4:0 – CS[4:0] Timer Clock Source Selection									
Table 25-4. Timer Clock Sources									
CS	Timer1	Timer3	Timer5						
11111-10110		Reserved							
10101		CLC8_OUT							
10100		CLC7_OUT							
10011		CLC6_OUT							
10010		CLC5_OUT							
10001		CLC4_OUT							
10000		CLC3_OUT							
01111		CLC2_OUT							
01110		CLC1_OUT							
01101	TMR5_OUT	TMR5_OUT							
01100	TMR3_OUT	Reserved							
01011		TMR1_OUT							
01010		TMR0_OUT							
01001		CLKREF_OUT							
01000		EXTOSC							
00111		SOSC							
00110		MFINTOSC (31.25 kHz)							
00101		MFINTOSC (500 kHz)							
00100		LFINTOSC							
00011		HFINTOSC							
00010		Fosc							
00001		Fosc/4							
00000	Pin selected by T1CKIPPS	Pin selected by T3CKIPPS	Pin selected by T5CKIPPS						

Reset States: POR/BOR = 00000
All Other Resets = uuuuu

10

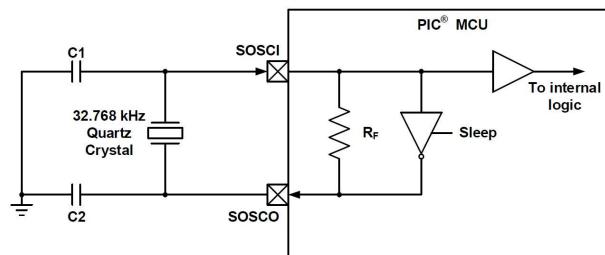
El Timer 1 en el PIC18F57Q43



- Oscilador secundario

12.1.1.5 Secondary Oscillator

The Secondary Oscillator (SOSC) is a separate external oscillator block that can be used as an alternate system clock source or as a Timer clock source. The SOSC is optimized for 32.768 kHz, and can be used with either an external quartz crystal connected to the SOSCI and SOSCO pins, or with an external clock source connected to the SOSCI pin as shown in the figures below.



11

El CCP1 en el PIC18F57Q43

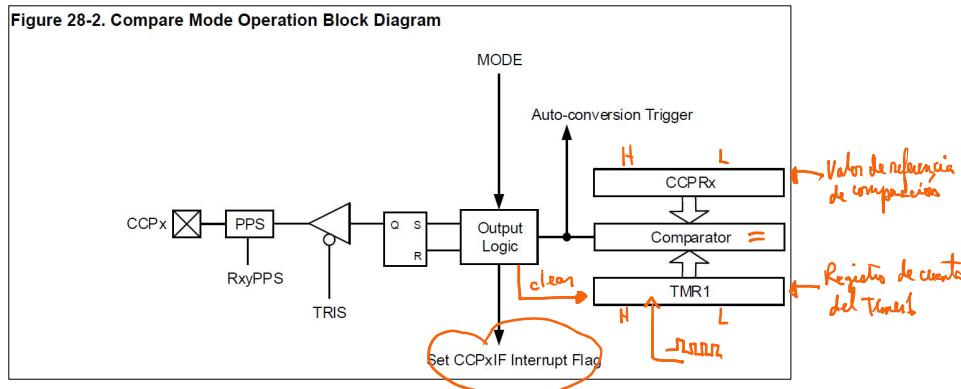
- Módulo
captura/comparación/PWM

CCP Mode	Timer Resource
Capture	Timer1, Timer3 or Timer5
Compare	
PWM	Timer2, Timer4 or Timer6

12

El CCP1 en el PIC18F57Q43

- CCP1 en modo comparación
- Diagrama de bloques



13

El CCP1 en el PIC18F57Q43

- CCP1 en modo comparación
- Información importante:

28.3.2 Timer1 Mode for Compare

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See the “TMR1 - Timer1 Module with Gate Control” section for more information on configuring Timer1.



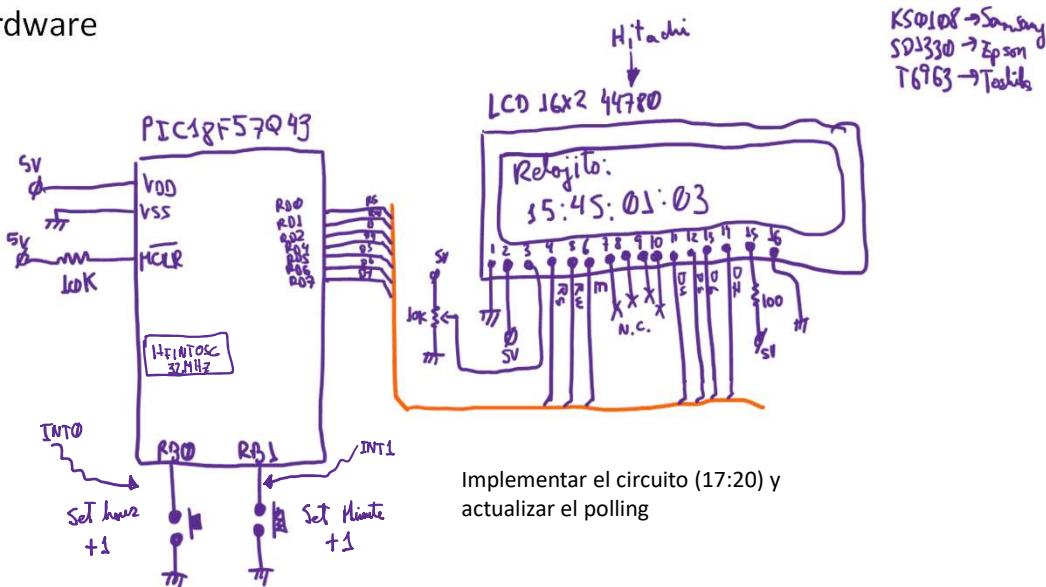
Important: Clocking Timer1 from the system clock (F_{osc}) **must not be used in Compare mode**. For Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ($F_{osc}/4$) or from an external clock source.

$$f_{osc} \rightarrow 32.768 \text{ KHz}$$

14

Ejercicio: Reloj empleando el Timer1

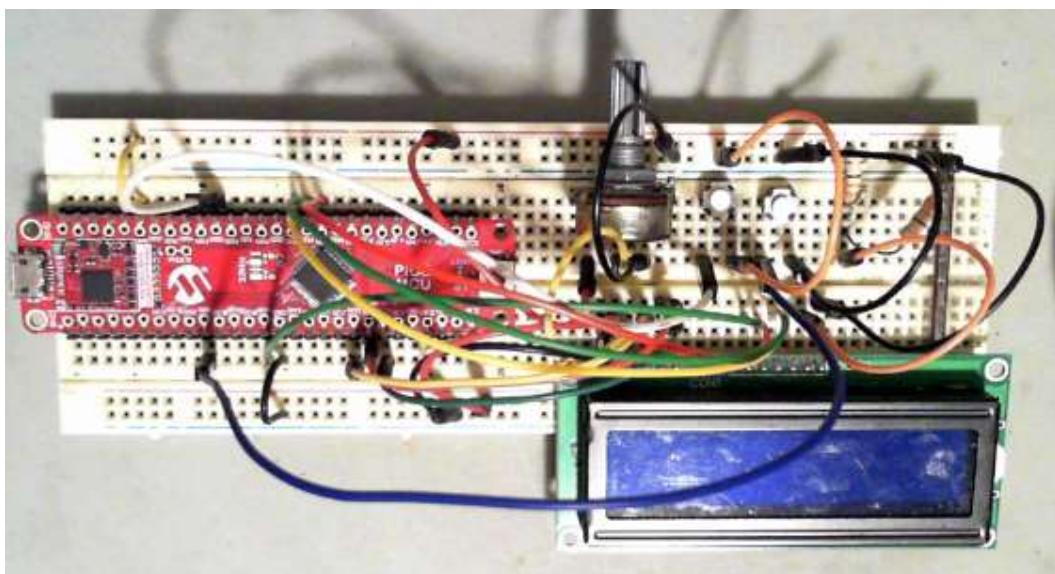
- Hardware



15

Ejercicio: Reloj empleando el Timer1

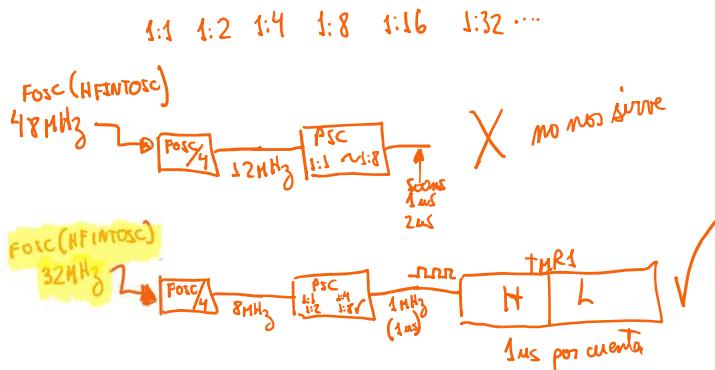
- Hardware



16

Ejercicio: Reloj empleando el Timer1

- A tener en cuenta con respecto a la frecuencia del reloj principal

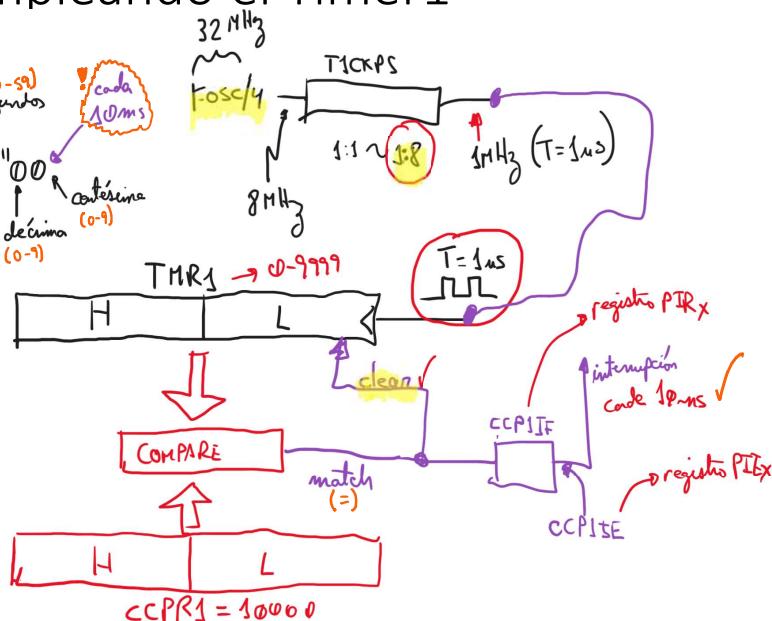


17

Ejercicio: Reloj empleando el Timer1

Analizando:

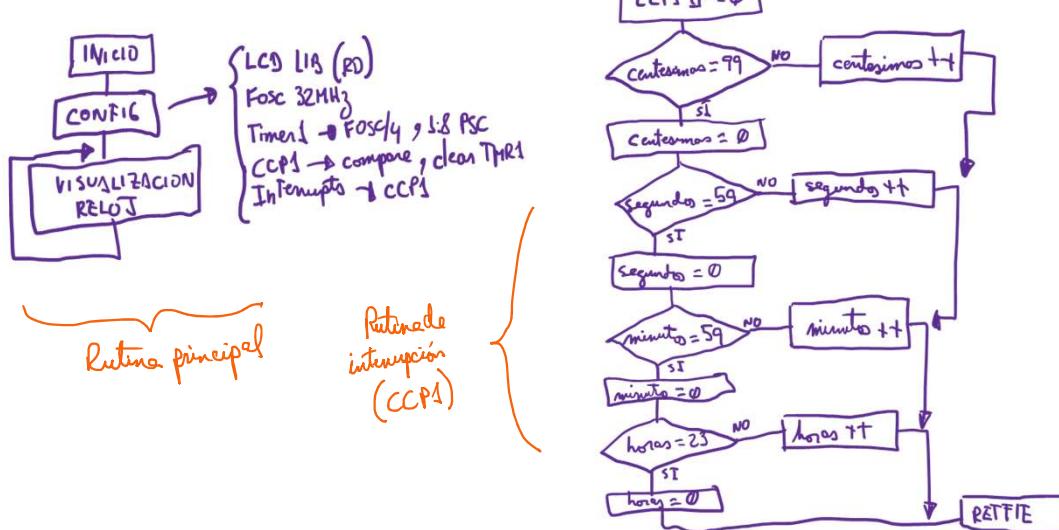
- Timer 1 conectado a CCP1
- CCP1 en modo comparador evento especial de disparo
- FOSC (HFINTOSC) debe de configurarse a 32MHz
- Prescaler del Timer1: 1:8
- Valor de referencia de comparación del CCP1: 10000
- Cuando cuentas del Timer1 llegue a 10000 ocurrirá una igualdad y será reiniciada la cuenta, por lo tanto el renglón de cuentas será de 0000 a 9999
- Cuando ocurra el match se levantará la bandera CCP1F y generará una interrupción, esto sucederá de manera repetitiva y regular cada 10ms.



18

Ejercicio: Reloj empleando el Timer1

- Algoritmo



19

Ejercicio: Reloj empleando el Timer1

- Cuatro tareas principales a cumplir en respecto a la base de tiempo:
 - HFINTOSC a 32MHz
 - Configurar el Timer1
 - Configurar el CCP1 (compare mode) y establecer valor de referencia de comparación a 10000
 - Configurar las interrupciones

20

Ejercicio: Reloj empleando el Timer1

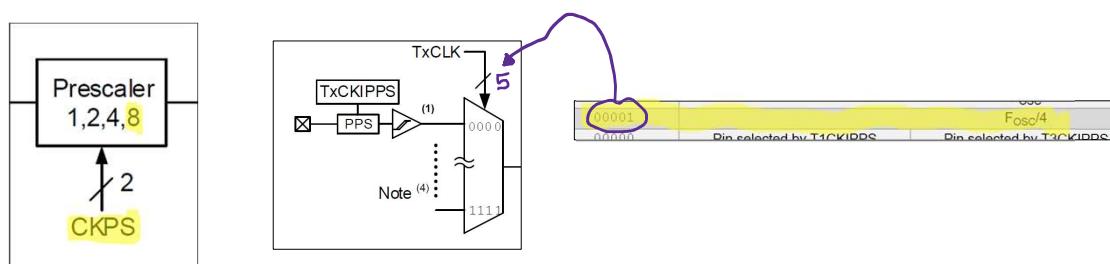
- Configuración de la fuente de reloj:
HFINTOSC a 32MHz:
 - OSCCON1 = 0x60;
 - OSCFRQ = 0x06;
 - OSCEN = 0x40

21

Ejercicio: Reloj empleando el Timer1

- Configuración del Timer1:

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x031C	TMR1	7:0				TMR1[7:0]				
		15:8				TMR1[15:8]				
0x031E	T1CON	7:0			CKPS[1:0]					
0x031F	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	SYNC	RD16	ON
0x0320	T1GATE	7:0					GSS[5:0]			
0x0321	T1CLK	7:0						CS[4:0]		



22

Ejercicio: Reloj empleando el Timer1

- Registro T1CLK

$T1CLK = 0x\text{D}1$

↑
Source Fosc/4

Name: TxCLK
Address: 0x321,0x32D,0x339

Timer Clock Source Selection Register

Bit	7	6	5	4	3	2	1	0
Access	R/W							
Reset	1	0	0	0	0	0	0	0

Bits 4:0 – CS[4:0] Timer Clock Source Selection
Table 25-4. Timer Clock Sources

CS	Clock Source		
	Timer1	Timer3	Timer5
11111-10110		Reserved	
10101	CLC8_OUT		
10100	CLC7_OUT		
10011	CLC6_OUT		
10010	CLC5_OUT		
10001	CLC4_OUT		
10000	CLC3_OUT		
01111	CLC2_OUT		
01110	CLC1_OUT		
01101	TMR5_OUT	TMR5_OUT	Reserved
01100	TMR3_OUT	Reserved	TMR3_OUT
01011	Reserved	TMR1_OUT	TMR1_OUT
01010		TMR0_OUT	
01001		CLKREF_OUT	
01000		EXTOSC	
00111		SOSC	
00110		MFINTOSC (31.25 kHz)	
00101		MFINTOSC (500 kHz)	
00100		LFINTOSC	
00011		HFINTOSC	
00010		Fosc	
00001		Fosc/4	
00000	Pin selected by T1CKIPPS	Pin selected by T3CKIPPS	Pin selected by T5CKIPPS

23

Ejercicio: Reloj empleando el Timer1

- Registro T1CON:

$T1CON = 0x\text{3}3$

prescaler 1:8
timer on

Name: TxCON
Address: 0x31E,0x32A,0x336

Timer Control Register

Bit	7	6	5	4	3	2	1	0
Access	R/W	R/W	R/W	R/W	R/W	X/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 5:4 – CKPS[1:0] Timer Input Clock Prescaler Select
Reset States: POR/BOR = 00
All Other Resets = uu

Value	Description
11	1:8 Prescaler value
10	1:4 Prescaler value
01	1:2 Prescaler value
00	1:1 Prescaler value

Bit 2 – SYNC Timer External Clock Input Synchronization Control
Reset States: POR/BOR = 0
All Other Resets = u

Value	Condition	Description
x	CS = Fosc/4 or Fosc	This bit is ignored. Timer uses the incoming clock as is.
1	All other clock sources	Do not synchronize external clock input
0	All other clock sources	Synchronize external clock input with system clock

Bit 1 – RD16 16-Bit Read/Write Mode Enable
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Enables register read/write of Timer in one 16-bit operation
0	Enables register read/write of Timer in two 8-bit operations

Bit 0 – ON Timer On
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Enables Timer
0	Disables Timer

24

Ejercicio: Reloj empleando el Timer1

- Configuración del CCP1:

Address	Name	Bit Pos.	7	6	5	4	3	2	1	0
0x0340	CCPR1	L	7:0				CCPR[7:0]			
		H	15:8				CCPR[15:8]			
0x0342	CCP1CON		7:0	EN		OUT	FMT		MODE[3:0]	
0x0343	CCP1CAP		7:0						CTS[3:0]	

$\text{CCPR1H} = 0x27$ ✓ } Valor de referencia del comparador
 $\text{CCPR1L} = 0x10$ }
 ~~~~~ 10000 en decimal  
 $\text{CCPR1} = 10000; X$   
Modo de comparación

25

## Ejercicio: Reloj empleando el Timer1

- Registro CCP1CON:

CCP1CON = 0x81

| CCP Control Register |     |   |   |   |   |   |   |   |     |  |
|----------------------|-----|---|---|---|---|---|---|---|-----|--|
| Bit                  | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |     |  |
| Access               | R/W |   |   | R | X |   |   |   | R/W |  |
| Reset                | 0   |   |   | 0 |   | 0 | 0 | 0 | R/W |  |
|                      |     |   |   |   |   |   |   |   |     |  |

**Bit 7 – EN CCP Module Enable**

| Value | Description     |
|-------|-----------------|
| 1     | CCP is enabled  |
| 0     | CCP is disabled |

**Bit 5 – OUT CCP Output Data (read-only)**

**Bit 4 – FMT CCPxRH:L Value Alignment (PWM mode)**

| Value | Condition    | Description          |
|-------|--------------|----------------------|
| x     | Capture mode | Not used             |
| x     | Compare mode | Not used             |
| 1     | PWM mode     | Left aligned format  |
| 0     | PWM mode     | Right aligned format |

**Bits 3:0 – MODE[3:0] CCP Mode Select**

Table 28-5. CCPx Mode Select

| MODE Value | Operating Mode | Operation                                        | Set CCPxIF |
|------------|----------------|--------------------------------------------------|------------|
| 11xx       | PWM            | PWM operation                                    | Yes        |
| 1011       | Compare        | Pulse output; clear TMR1 <sup>(2)</sup>          | Yes        |
| 1010       |                | Pulse output                                     | Yes        |
| 1001       |                | Clear output <sup>(1)</sup>                      | Yes        |
| 1000       |                | Set output <sup>(1)</sup>                        | Yes        |
| 0111       |                | Every 16 <sup>th</sup> rising edge of CCPx input | Yes        |
| 0110       |                | Every 4 <sup>th</sup> rising edge of CCPx input  | Yes        |
| 0101       |                | Every rising edge of CCPx input                  | Yes        |
| 0100       |                | Every falling edge of CCPx input                 | Yes        |
| 0011       |                | Every edge of CCPx input                         | Yes        |
| 0010       | Compare        | Toggle output                                    | Yes        |
| 0001       |                | Toggle output; clear TMR1 <sup>(2)</sup>         | Yes        |
| 0000       | Disabled       |                                                  | —          |

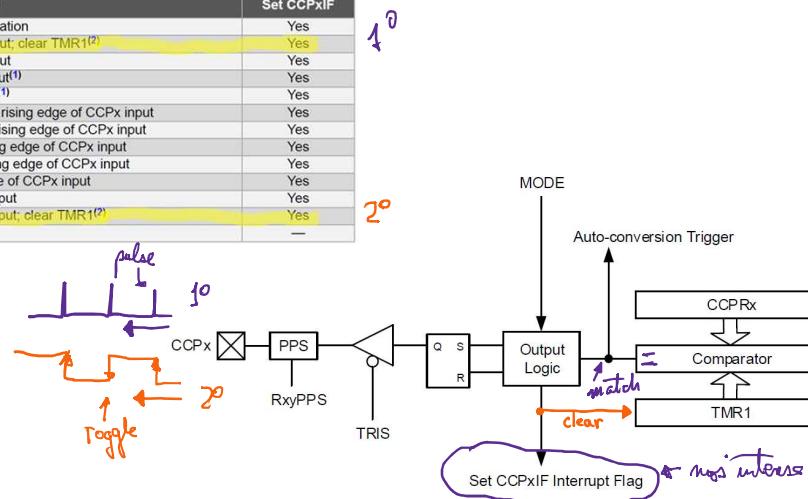
26

## Ejercicio: Reloj empleando el Timer1

- Registro CCP1CON:

Bits 3:0 – MODE[3:0] CCP Mode Select  
Table 28-5. CCPx Mode Select

| MODE Value | Operating Mode | Operation                                        | Set CCPxIF |
|------------|----------------|--------------------------------------------------|------------|
| 11xx       | PWM            | PWM operation                                    | Yes        |
| 1011       | Compare        | Pulse output; clear TMR1(2)                      | Yes        |
| 1010       |                | Pulse output                                     | Yes        |
| 1001       |                | Clear output <sup>(1)</sup>                      | Yes        |
| 1000       |                | Set output <sup>(1)</sup>                        | Yes        |
| 0111       |                | Every 16 <sup>th</sup> rising edge of CCPx input | Yes        |
| 0110       |                | Every 4 <sup>th</sup> rising edge of CCPx input  | Yes        |
| 0101       |                | Every rising edge of CCPx input                  | Yes        |
| 0100       |                | Every falling edge of CCPx input                 | Yes        |
| 0011       |                | Every edge of CCPx input                         | Yes        |
| 0010       |                | Toggle output                                    | Yes        |
| 0001       | Compare        | Toggle output; clear TMR1(2)                     | Yes        |
| 0000       | Disabled       |                                                  | —          |



27

## Ejercicio: Reloj empleando el Timer1

- Configuración de las interrupciones del CCP1:

|        |      |     |        |        |         |        |        |        |          |          |
|--------|------|-----|--------|--------|---------|--------|--------|--------|----------|----------|
| 0x04A1 | PIE3 | 7:0 | TMROIE | CCP1IE | TMR1GIE | TMR1IE | TMR2IE | SPI1IE | SPI1TXIE | SPI1RXIE |
|--------|------|-----|--------|--------|---------|--------|--------|--------|----------|----------|

1 → habilitar la int del CCP1

|        |         |     |          |      |      |  |  |         |         |         |
|--------|---------|-----|----------|------|------|--|--|---------|---------|---------|
| 0x04D6 | INTCON0 | 7:0 | GIE/GIEH | GIEL | IPEN |  |  | INT2EDG | INT1EDG | INT0EDG |
|--------|---------|-----|----------|------|------|--|--|---------|---------|---------|

↑ 1 → habilitar global de bits

|        |      |     |        |        |         |        |        |        |          |          |
|--------|------|-----|--------|--------|---------|--------|--------|--------|----------|----------|
| 0x04B1 | PIR3 | 7:0 | TMROIF | CCP1IF | TMR1GIF | TMR1IF | TMR2IF | SPI1IF | SPI1TXIF | SPI1RXIF |
|--------|------|-----|--------|--------|---------|--------|--------|--------|----------|----------|

bandera

28

**Ejercicio: Reloj empleando el Timer1**

```

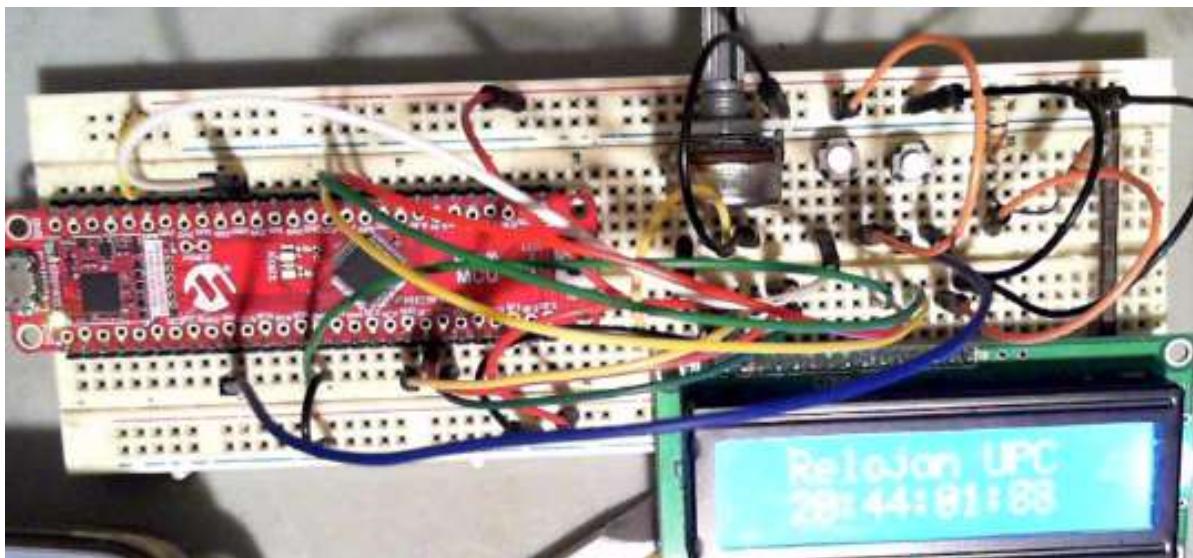
1 #include <xc.h>
2 #include "cabecera.h"
3 #include "LCD.h"
4 #define _XTAL_FREQ 32000000UL
5
6 unsigned char horas=20,minutos=02,segundos=16,centesimas=58;
7 unsigned char centena,decena,unidad;
8
9 void configuro(void){
10     //configuracion del oscilador
11     OSCCON1 = 0x60;
12     OSCFRCQ = 0x06; //HFINTOSC a 32MHz
13     OSCEN = 0x40;
14     //configuracion del Timer1
15     T1CLK = 0x01;
16     T1CON = 0x33;
17     //configuracion del CCP1
18     CCP1CON = 0x81;
19     CCP1RH = 0x27;
20     CCP1RL = 0x10;
21     //configuracion de las interrupciones
22     PIE3bits.CCP1IE = 1;
23     INTCON0bits.GIE = 1;
24 }
25
26 void lcd_init(void){
27     TRISD = 0x00;
28     ANSEL0 = 0x00;
29     LCD_CONFIG();
30     _delay_ms(21);
31     BORRAR_LCD();
32     CURSOR_HOME();
33     CURSOR_ONOFF(OFF);
34 }
35
36 void convierte(unsigned char numero){
37     centena = numero / 100;
38     decena = (numero % 100) / 10;
39     unidad = numero % 10;
40 }
41
42 void main(void) {
43     configuro();
44     lcd_init();
45     POS_CURSOR(1,2);
46     ESCRIBE_MENSAJE("Relojon UPC",11);
47     while(1){
48         POS_CURSOR(2,2);
49         convierte(horas);
50         ENVIA_CHAR(decena+0x30);
51         ENVIA_CHAR(unidad+0x30);
52         ENVIA_CHAR(':');
53         convierte(minutos);
54         ENVIA_CHAR(decena+0x30);
55         ENVIA_CHAR(unidad+0x30);
56         ENVIA_CHAR(':');
57         convierte(segundos);
58         ENVIA_CHAR(decena+0x30);
59         ENVIA_CHAR(unidad+0x30);
60         ENVIA_CHAR(':');
61         convierte(centesimas);
62         ENVIA_CHAR(decena+0x30);
63         ENVIA_CHAR(unidad+0x30);
64     }
65 }
66
67 void __interrupt(irq(IRQ_CCPI)) CCP1_ISR(void){
68     PIE3bits.CCP1IF = 0;
69     if(centesimas == 99){
70         centesimas = 0;
71         if(segundos == 59){
72             segundos = 0;
73             if(minutos == 59){
74                 minutos = 0;
75                 if(horas == 23){
76                     horas = 0;
77                 } else{
78                     horas++;
79                 }
80             } else{
81                 minutos++;
82             }
83         } else{
84             segundos++;
85         }
86     } else{
87         centesimas++;
88     }
89 }
90
91 void __interrupt(irq(default)) DEFAULT_ISR(void){
92 }
93
94 // Unhandled interrupts go here
95
96

```

Abrir MPLABX, crear proyecto nuevo, llamar librería LCD, crear archivos \*.h y \*.c, transcribir el presente código y hacer pruebas (30 minutos)

29

## Ejercicio: Reloj empleando el Timer1



30

Equipo para medir la precision de un reloj electrónico basado en cristal de cuarzo:



31

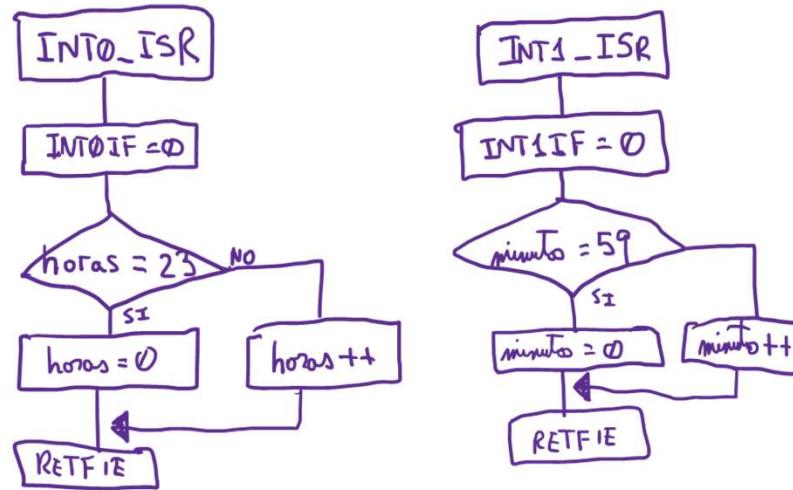
### Ejercicio: Reloj empleando el Timer1

- Etapa de ajuste de la hora empleando los pulsadores en RB0/INT0 y RB1/INT1
  - Según el circuito implementado, los botones que están en INT0 e INT1 respectivamente son activos en bajo
  - Se debe de activar las pullup en ambos puertos (WPUB.0 y WPUB.1)
  - Ambas INTs deben de configurarse en flaco descendente (INT0EDG y INT1EDG deben de ser cero).

32

## Ejercicio: Reloj empleando el Timer1

- Etapa de ajuste de la hora empleando los pulsadores en RB0/INT0 y RB1/INT1



33

## Ejercicio: Reloj empleando el Timer1

- Configuración de los puertos RB0 y RB1 donde están los pulsadores:

| Address | Name   | Bit Pos. | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|---------|--------|----------|---------|---------|---------|---------|---------|---------|---------|---------|
| 0x04C7  | TRISB  | 7:0      | TRISB7  | TRISB6  | TRISB5  | TRISB4  | TRISB3  | TRISB2  | TRISB1  | TRISB0  |
| 0x0408  | ANSELB | 7:0      | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 |
| 0x0409  | WPUB   | 7:0      | WPUB7   | WPUB6   | WPUB5   | WPUB4   | WPUB3   | WPUB2   | WPUB1   | WPUB0   |

Annotations in purple highlight specific bits and their functions:

- Bit 0 of TRISB is labeled "1 para entrada".
- Bit 0 of ANSELB is labeled "0 para digital".
- Bit 0 of WPUB is labeled "1 para habilitar, weak pull-up".

34

## Ejercicio: Reloj empleando el Timer1

- Configuración de las interrupciones de INT0 e INT1:

| Address | Name    | Bit Pos. | 7         | 6         | 5          | 4          | 3      | 2       | 1       | 0       |
|---------|---------|----------|-----------|-----------|------------|------------|--------|---------|---------|---------|
| 0x049F  | PIE1    | 7:0      | SMT1PWAIE | SMT1PRAIE | SMT1IE     | CM1IE      | ACTIE  | ADIE    | ZCDIE   | INT0IE  |
| 0x04A4  | PIE6    | 7:0      | DMA2AIE   | DMA2ORIE  | DMA2DCNTIE | DMA2SCNTIE | NCO1IE | CWG1IE  | CLC2IE  | INT1IE  |
| 0x04D6  | INTCON0 | 7:0      | GIE/GIEH  | GIEL      | IPEN       |            |        | INT2EDG | INT1EDG | INT0EDG |
| 0x04AF  | PIR1    | 7:0      | SMT1PWAI  | SMT1PRAI  | SMT1IF     | CM1IF      | ACTIF  | ADIF    | ZCDIF   | INT0IF  |
| 0x04B4  | PIR6    | 7:0      | DMA2AIF   | DMA2ORIF  | DMA2DCNTIF | DMA2SCNTIF | NCO1IF | CWG1IF  | CLC2IF  | INT1IF  |

35

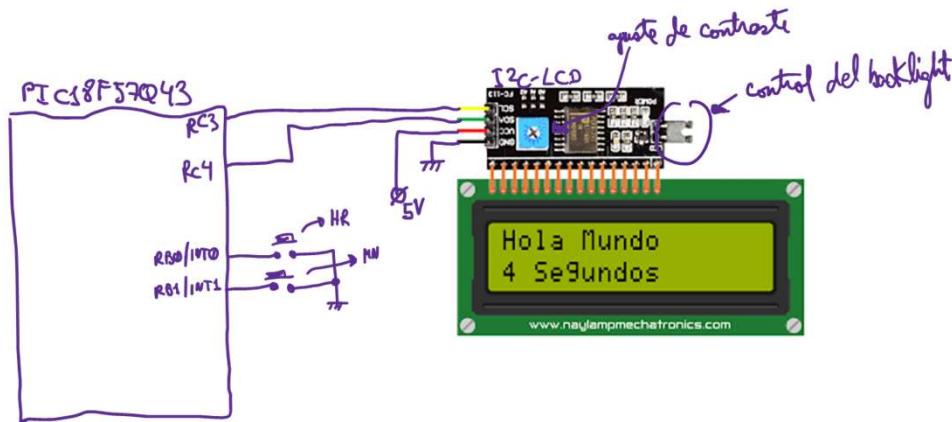
## Ejercicio: Reloj empleando el Timer1

• Código completo :

```
1 #include <xc.h>
2 #include "cabecera.h"
3 #include "LCD.h"
4 #define _XTAL_FREQ 32000000UL
5
6 unsigned char horas=10,minutos=10,segundos=10,centesimas=10;
7 unsigned char centena,decena,unidad;
8
9 void configuro(void){
10     //configuracion del oscilador
11     OSCCON1 = 0x60; //HFINTOSC, posts 1:1
12     OSCFRC0 = 0x06; //HFINTOSC a 32MHz
13     OSCEN = 0x40; //HFINTOSC enabled
14     //configuracion de E/S
15     TRISB = 0xFF; //RB1 RB0 como entradas
16     ANSELB = 0xFC; //RB1 RB0 como entradas
17     WUB = 0x03; //RB1 RB0 pullup enabled
18     //configuracion del Timer1
19     T1CKL = 0x01; //clk source fosc/4
20     T1CON = 0x33; //TMR1 ON, pres 1:8
21     //configuracion del CCP1
22     CCP1CON = 0x81; //compare mode, clear TMRI
23     CCP1R = 0x27;
24     CCP1R = 0x10; //valor de referencia 10000
25     //configuracion de las interrupciones
26     INTCON0bits.INT0EDG = 0; //falling edge en INT0
27     INTCON0bits.INT1EDG = 0; //falling edge en INT1
28     PIEbits.INT0IE = 1; //INT0 enabled
29     PIEbits.INT1IE = 1; //INT1 enabled
30     PIEbits.CCPIE = 1; //CCP1 enabled
31     PIRbits.INT0IF = 0; //flag INT0 bajada
32     PIRbits.INT1IF = 0; //flag INT1 bajada
33     PIRbits.CCP1IF = 0; //flag CCP1 bajada
34     INTCON0bits.GIE = 1; //global ints enabled
35 }
36
37 void lcd_init(void){
38     TRISD = 0x00;
39     ANSELD = 0x00;
40     LCD_CONFIG();
41     _delay_ms(21);
42     BORRAR_LCD();
43     CURSOR_HOME();
44     CURSOR_ONOFF(OFF);
45 }
46
47 void convierte(unsigned char numero){
48     centena = numero / 100;
49     decena = (numero % 100) / 10;
50     unidad = numero % 10;
51 }
52
53 void main(void) {
54     configuro();
55     lcd_init();
56     POS_CURSOR(1,2);
57     ESCRIBE_MENSAJE("Reloj en UPC",11);
58     while(1){
59         POS_CURSOR(2,2);
60         convierte(horas);
61         ENVIA_CHAR(decena+0x30);
62         ENVIA_CHAR(unidad+0x30);
63         ENVIA_CHAR(':');
64         convierte(minutos);
65         ENVIA_CHAR(decena+0x30);
66         ENVIA_CHAR(unidad+0x30);
67         ENVIA_CHAR(':');
68         convierte(segundos);
69         ENVIA_CHAR(decena+0x30);
70         ENVIA_CHAR(unidad+0x30);
71         ENVIA_CHAR(':');
72         convierte(centesimas);
73         ENVIA_CHAR(decena+0x30);
74         ENVIA_CHAR(unidad+0x30);
75     }
76 }
77
78 void __interrupt(irq(IRQ_CCPI)) CCP1_ISR(void){
79     PIR3bits.CCPIIF = 0;
80     if(centesimas == 99){
81         centesimas = 0;
82         if(segundos == 59){
83             segundos = 0;
84             if(minutos == 59){
85                 horas++;
86             }
87         }
88     }
89 }
90
91 void __interrupt(irq(IRQ_INTO)) INTO_ISR(void){
92     PIR6bits.INT0IF = 0;
93     if(horas == 23){
94         horas = 0;
95     }
96     else{
97         segundos++;
98     }
99 }
100
101 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
102     PIR6bits.INT1IF = 0;
103     if(minutos == 59){
104         minutos = 0;
105     }
106     else{
107         horas++;
108     }
109 }
110
111 void __interrupt(irq(default)) DEFAULT_ISR(void){
112     {
113         // Unhandled interrupts go here
114     }
115 }
```

36

## Mejora: empleo del modulo I2C-LCD

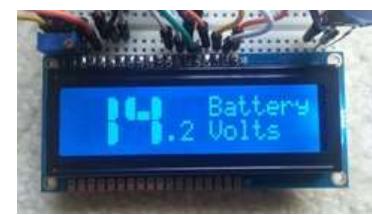
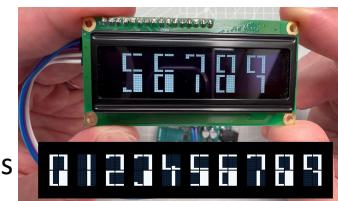


- Nota: Se debe de emplear la librería I2C\_LCD (revisar el repositorio)

37

## Ejercicio: Reloj empleando el Timer1

- Adicionales:
  - Cambio de formato 12H/24H
  - Sistema de alarma (activación/desactivación)
  - Función de cronómetro
  - Función de cuenta regresiva
  - Medición de temperatura / humedad con DHT11
  - Animación de secundero
  - Visualización de reloj analógico usando caracteres personalizados
  - Visualización de caracteres gigantes en el LCD
  - Sincronismo con NTP
  - Acelerómetro para encender la pantalla ante un movimiento



38

## 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, ailerones (flaps), timón, robótica.
- Son clasificados por: tamaño, torque, velocidad, precisión, tamaño



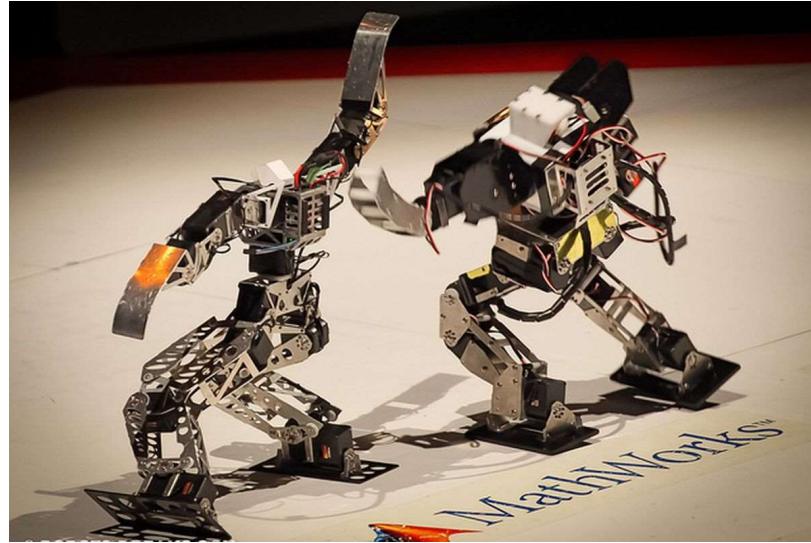
39

## Servos en el hobby profesional



40

## Servos en el hobby profesional



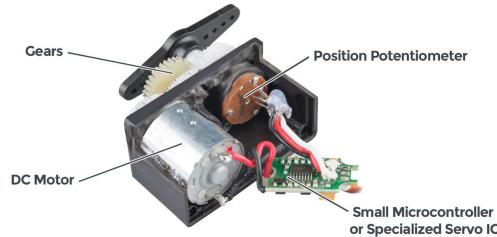
41



42

## El servo

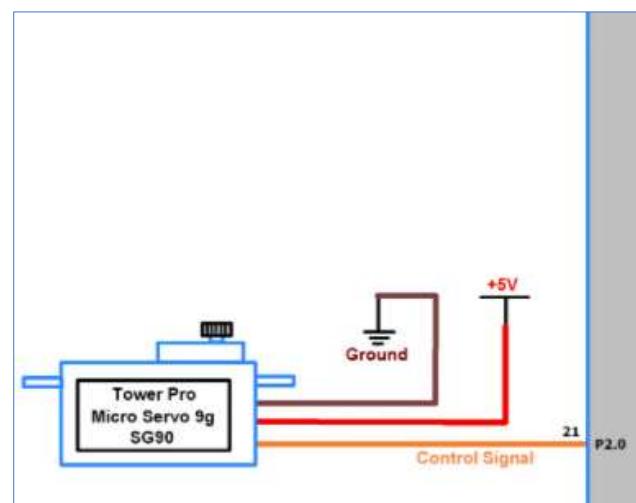
- Internamente posee un motor realimentado con un potenciómetro y mecanismo de reducción.



43

## El servo

- Conexión con el microcontrolador:



44

## Specs del miniservo sg90:

- De preferencia usar fuente externa para los servomecanismos

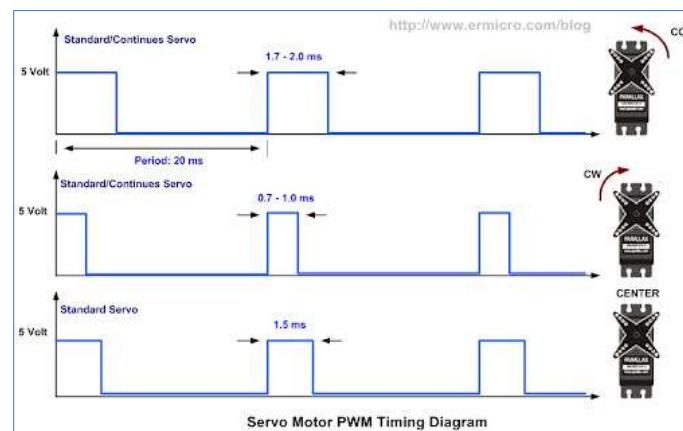
### SPECIFICATIONS:

|                                            | Operating Voltage          |                           |
|--------------------------------------------|----------------------------|---------------------------|
|                                            | 4.8V                       | 6.0V                      |
| <b>Stall Torque</b>                        | 1.3 kg/cm<br>(18.09 oz/in) | 15 kg/cm<br>(20.86 oz/in) |
| <b>Max Speed</b>                           | 0.12sec/60°                | 0.12sec/60°               |
| <b>Idle Current</b>                        | 5 mA                       | 6 mA                      |
| <b>No Load Running Current</b>             | 100 mA                     | 120 mA                    |
| <b>Stall Current</b>                       | 700 mA                     | 800 mA                    |
| <b>Operating Voltage Range</b>             | 4.8V to 6V                 |                           |
| <b>Pulse Width Range</b>                   | 900 to 2100 µs             |                           |
| <b>Limit Angle</b>                         | 120° (900 to 2100 µs)      |                           |
| <b>Dead Band Width</b>                     | 10 µs                      |                           |
| <b>Stop position</b>                       | 1500 ( $\pm 5$ ) µs        |                           |
| <b>CW Rotation Signal Range</b>            | 900 to 1500 µs             |                           |
| <b>CCW Rotation Signal Range</b>           | 1500 to 2100 µs            |                           |
| <b>Internal Gears</b>                      | Plastic                    |                           |
| <b>Cable Length</b>                        | 20 cm (7.9")               |                           |
| <b>Dimensions (detailed dimensions)</b>    |                            |                           |
| Length                                     | 32.6 mm (1.28")            |                           |
| Width                                      | 12.5 mm (0.49")            |                           |
| Height                                     | 27.3 mm (1.08")            |                           |
| <b>Weight (of servo itself)</b>            | 9 g (0.32 oz)              |                           |
| <b>Weight (including horns and screws)</b> | 15.1 g (0.53 oz)           |                           |

45

## El servo

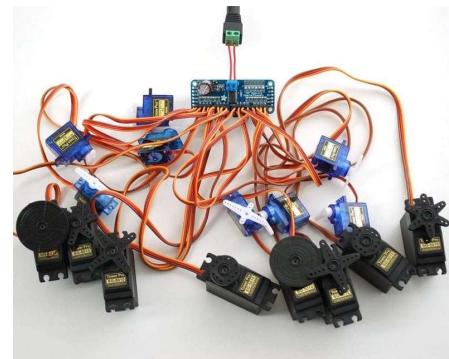
- Modo de funcionamiento:
  - Tren de pulsos de periodo 20ms (f=50Hz)
  - El ancho del pulso (1.0ms – 2.0ms) positivo determinará la posición del eje del servo
  - Al quitarle el tren de pulsos el servo se inactivará



46

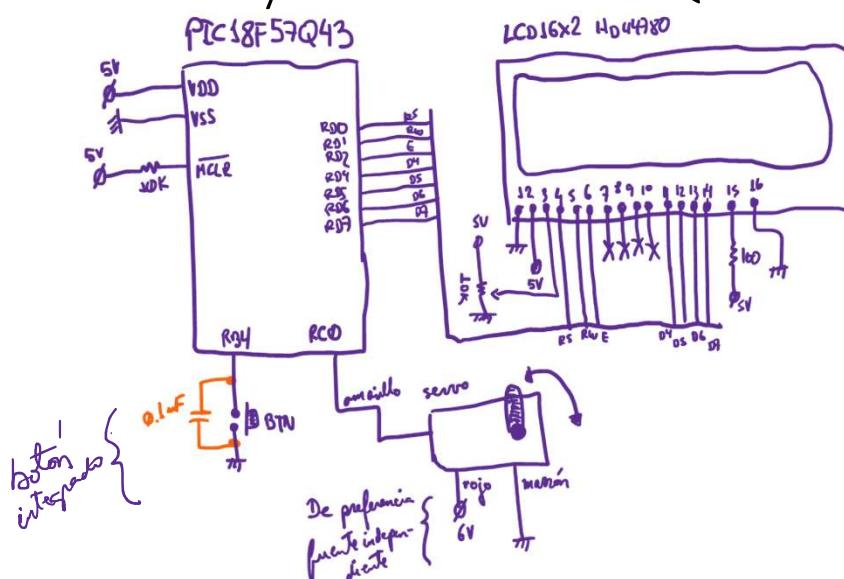
# Opciones para obtener la señal requerida para manipular el servo

- Utilizando retardos ✓
  - PWM a través del módulo CCP ✗ ✓
  - Uso de temporizadores (Timer0)
  - PCA9685A (controlador I2C para sistemas con LEDs)



47

# Circuito de prueba para manipular un servo con el Curiosity Nano PIC18F57Q43



48

## ¿Cómo generar la onda cuadrada de 50Hz con el módulo CCP del PIC18F57Q43?

- El CCP en modo PWM nos permitiría obtener dicha señal cuadrada
- Tenemos que validar si el CCP-PWM permite generar una señal a 50Hz

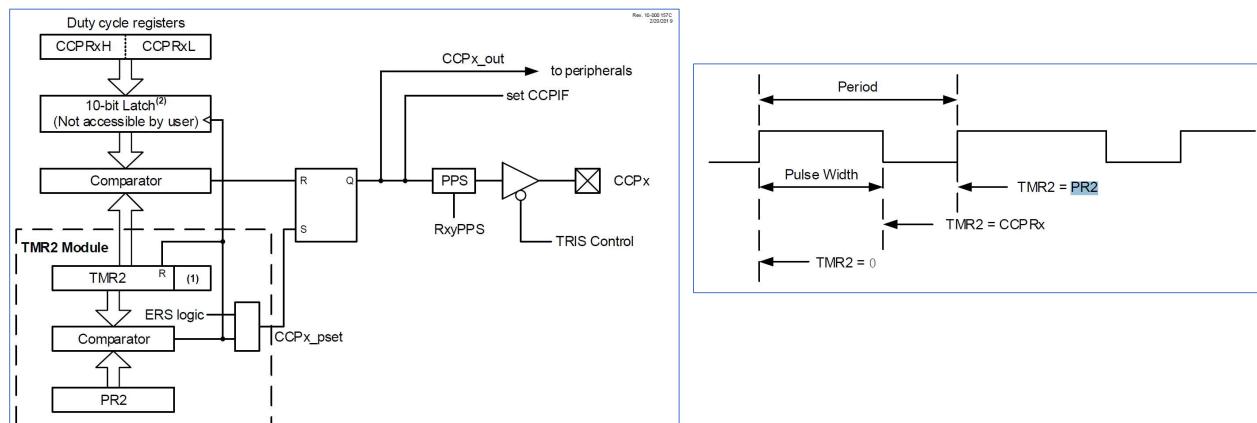
Table 28-1. CCP Mode - Timer Resources

| CCP Mode | Timer Resource           |
|----------|--------------------------|
| Capture  | Timer1, Timer3 or Timer5 |
| Compare  |                          |
| PWM      | Timer2, Timer4 or Timer6 |

49

## ¿Cómo generar la onda cuadrada de 50Hz con el módulo CCP del PIC18F57Q43?

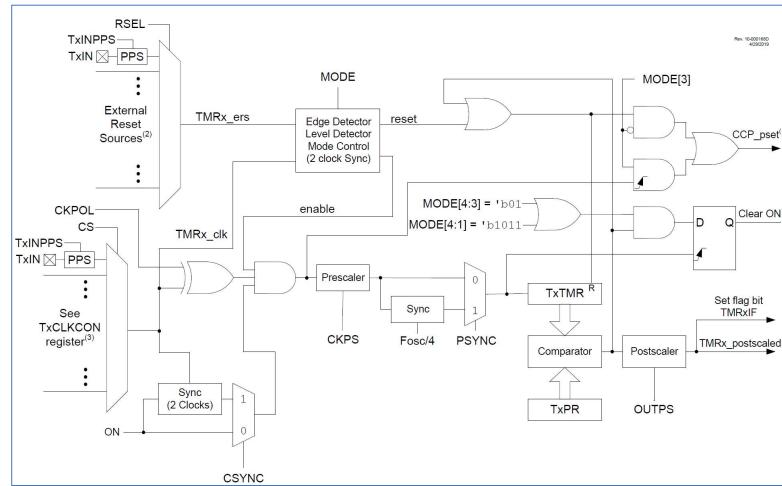
- Diagrama de bloques del modo PWM



50

## ¿Cómo generar la onda cuadrada de 50Hz con el módulo CCP del PIC18F57Q43?

- En el modo PWM del CCP, el periodo lo determina Timer2:



51

## ¿Cómo generar la onda cuadrada de 50Hz con el módulo CCP del PIC18F57Q43?

- En el modo PWM del CCP, el periodo lo determina Timer2:

### Equation 28-1. PWM Period

$$\text{PWM Period} = [(T2PR + 1)] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

20ms → T2PR para cada opción de prescale  
teniendo en cuenta que Fosc = 16MHz!  
T2PR → (registro de 8 bits)  
0 - 255

1:1 →  
1:2  
1:4  
1:8  
1:16 → 4999  
1:32 → 2499  
1:64 → 624  
1:128 → 312  
El mas bajo valor obtenido y no se puede usar

| Value | Description     |
|-------|-----------------|
| 111   | 1:128 Prescaler |
| 110   | 1:64 Prescaler  |
| 101   | 1:32 Prescaler  |
| 100   | 1:16 Prescaler  |
| 011   | 1:8 Prescaler   |
| 010   | 1:4 Prescaler   |
| 001   | 1:2 Prescaler   |
| 000   | 1:1 Prescaler   |

52

## ¿Cómo generar la onda cuadrada de 50Hz con el módulo CCP del PIC18F57Q43?

- En el modo PWM del CCP, el periodo lo determina Timer2:

### Equation 28-1. PWM Period

$$\text{PWM Period} = [(T2PR + 1)] \cdot 4 \cdot T_{OSC} \cdot (\underbrace{\text{TMR2 Prescale Value}}_{\text{CKPS[2:0]}})$$

20ms

$\downarrow$   
 T2PR para cada opción de prescale  
 teniendo en cuenta que Fosc = 4 MHz?  
 T2PR → (registro de 8 bits)  
 $0 - 255$

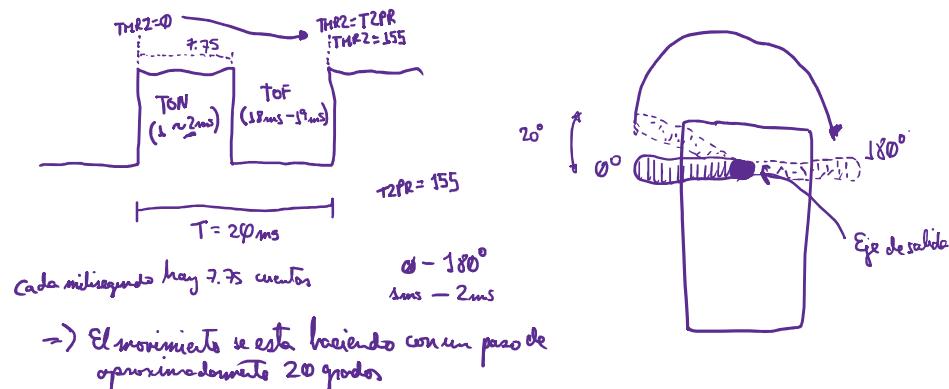
|       |            |
|-------|------------|
| 1:1   | → X        |
| 1:2   | → X        |
| 1:4   | → 499 X    |
| 1:8   | → 2499 X   |
| 1:16  | → 1249 X   |
| 1:32  | → 624 X    |
| 1:64  | → 311.5 X  |
| 1:128 | → 155.25 ✓ |

$$\boxed{T2PR = 155}$$

| CKPS[2:0] Timer Clock Prescale Select |                 |
|---------------------------------------|-----------------|
| Value                                 | Description     |
| 111                                   | 1:128 Prescaler |
| 110                                   | 1:64 Prescaler  |
| 101                                   | 1:32 Prescaler  |
| 100                                   | 1:16 Prescaler  |
| 011                                   | 1:8 Prescaler   |
| 010                                   | 1:4 Prescaler   |
| 001                                   | 1:2 Prescaler   |
| 000                                   | 1:1 Prescaler   |

53

## Resolución del duty cycle de tu PWM



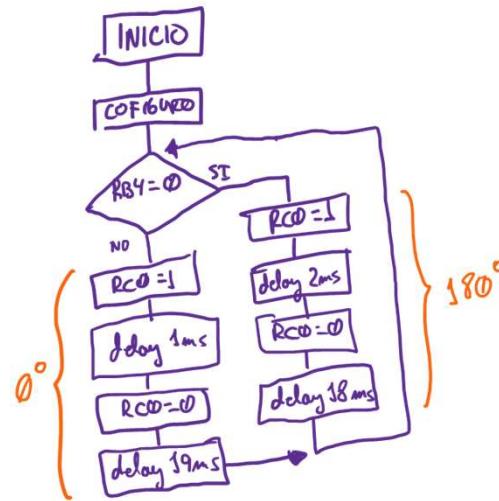
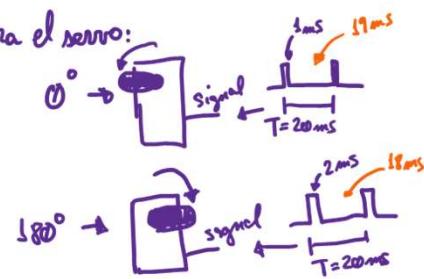
- En consecuencia: No se puede usar el CCP en modo PWM para controlar un servo porque se va a mover escalonadamente. Cada cambio de valor en el duty cycle determinado por CCPRx lo va a realizar en 20 grados angulares del eje del servomecanismo.

54

## Estrategia de usar retardos - `__delay_us()`;

- Usando el botón, intercambiar entre 0° y 180° el servomecanismo

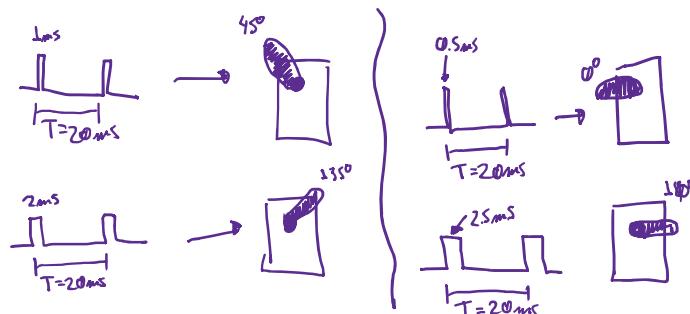
Para el servo:



55

## El servo no se mueve los 180°!

- Los servos sg90 requieren de un rango mas amplio del ancho del pulso positivo para poder movilizarse mas grados



56

## Código ejemplo empleando retardos

- Recomendación:

- Empezar con intervalos de 1ms a 2ms en el ancho positivo de la onda cuadrada y luego disminuir progresivamente el límite inferior al igual que el límite superior.
- No bajar de 0.5ms del límite inferior ni superar los 2.5ms de límite superior

```

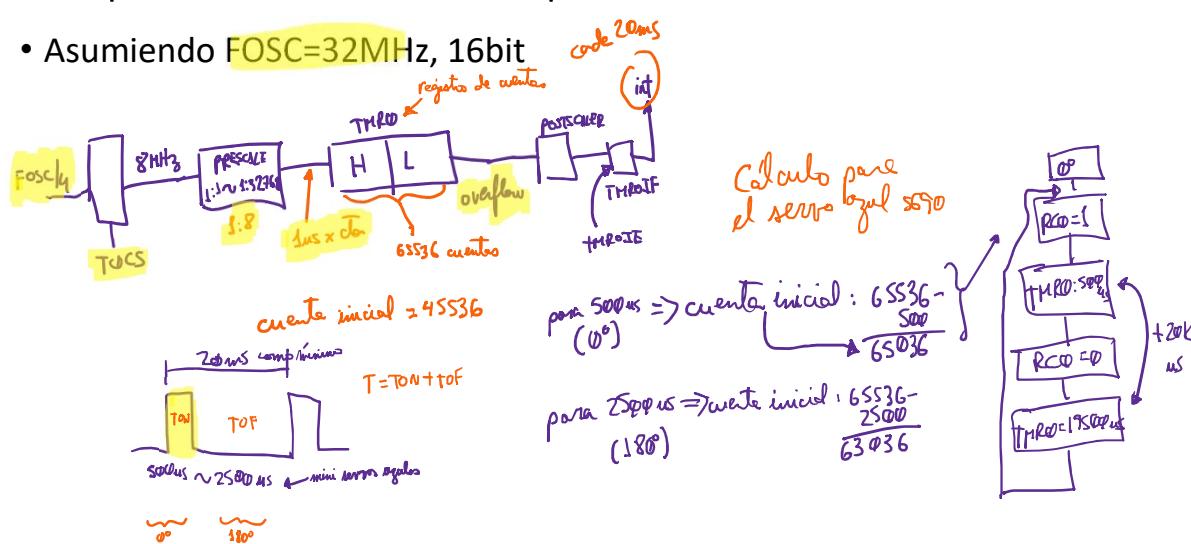
1  #include <xc.h>
2  #include "cabecera.h"
3  #include "LCD.h"
4  #define _XTAL_FREQ 32000000UL
5
6  void configuro(void){
7      //configuración del oscilador
8      OSCCON1 = 0x60;
9      OSCFRC = 0x06;           //HFINTOSC a 32MHz
10     OSCEN = 0x40;
11
12     //configuración de las E/S
13     TRISBbits.TRISB4 = 1;    //RB4 entrada
14     ANSELBbits.ANSELB4 = 0;  //RB4 digital
15     WPUBbits.WPUB4 = 1;     //RB4 con pullup activado
16     TRISChits.TRISCO = 0;   //RC0 salida
17     ANSELChits.ANSELCO = 1; //RC0 digital
18
19  void lcd_init(void){
20      TRISD = 0x00;
21      ANSELD = 0x00;
22      LCD_CONFIG();
23      __delay_ms(22);
24      BORRAR_LCD();
25      CURSOR_HOME();
26      CURSOR_ONOFF(OFF);
27  }
28
29  void main(void) {
30      configuro();
31      lcd_init();
32      POS_CURSOR(1,0);
33      ESCRIBE_MENSAJE("Servo UPCino", 12);
34      while(1){
35          if(PORTBbits.RB4 == 0){
36              LATCbits.LATCO = 1;
37              __delay_us(500);
38              LATCbits.LATCO = 0;
39              __delay_us(19500);
40          }
41          else{
42              LATCbits.LATCO = 1;
43              __delay_us(2500);
44              LATCbits.LATCO = 0;
45              __delay_us(17500);
46          }
47      }
48  }

```

57

## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

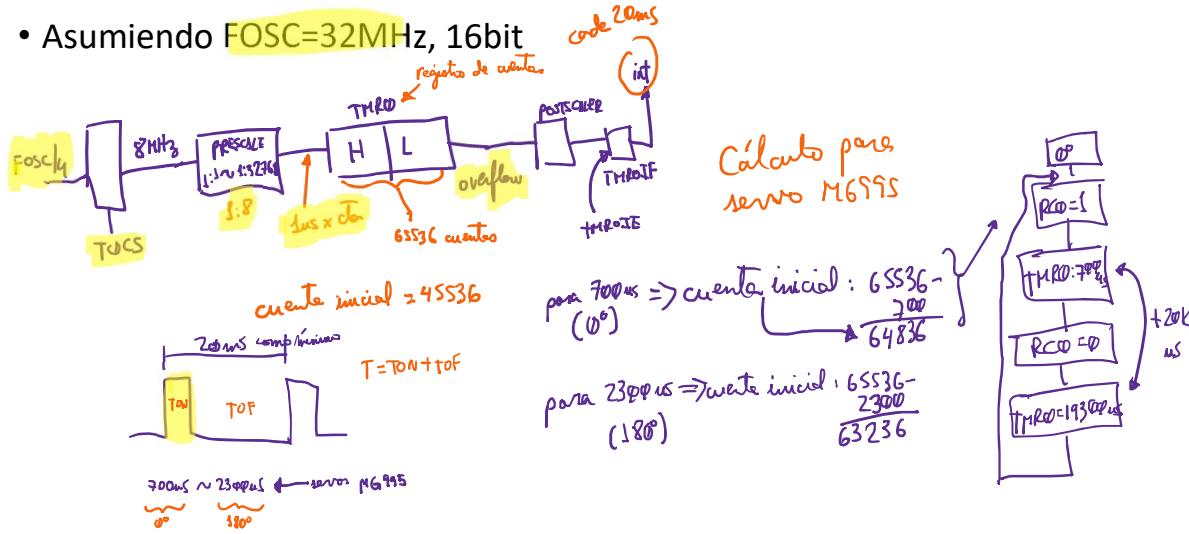
- Asumiendo FOSC=32MHz, 16bit



58

## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

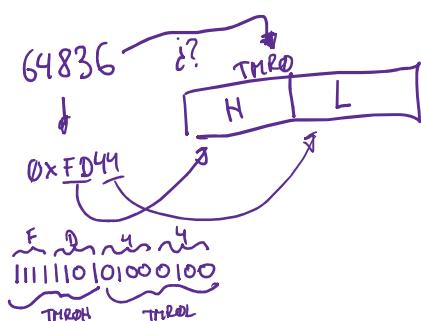
- Asumiendo  $\text{FOSC}=32\text{MHz}$ , 16bit



59

¿Cómo ingreso un número de cuenta inicial que esta en formato decimal a los registros de cuenta del Timer0?

Se tiene un número 64836 que deseamos colocar como cuenta inicial



$$\text{TMROD} = (64836 > 8) \& 0x00FF$$

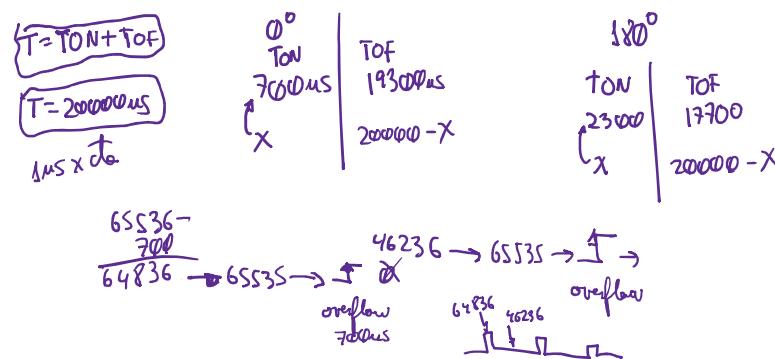
Explicación:  $64836_{10} = 111110101000100_2$ . La parte alta (TMROH) es 11111010 y la parte baja (TMROL) es 01000100.

$$\text{TMROL} = 64836 \& 0x00FF$$

Explicación:  $64836_{10} = 111110101000100_2$ . La parte alta (TMROH) es 11111010 y la parte baja (TMROL) es 01000100.

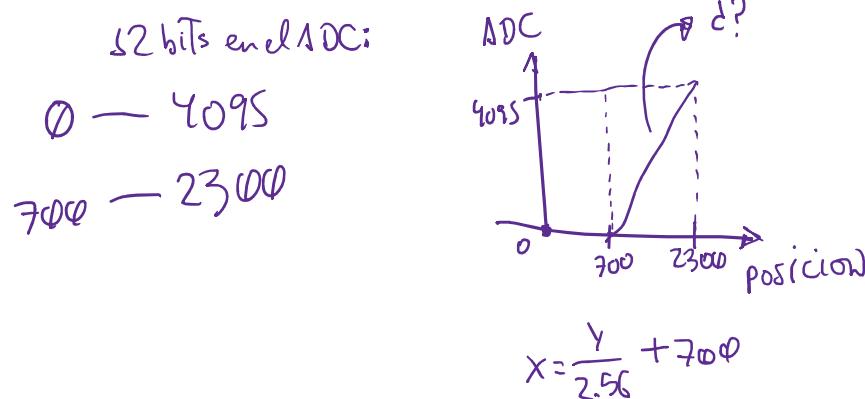
60

La relación T, TON, TOF con cuentas iniciales y posición de servo.



61

Relación ADC con temporizado de servo



62

## Cuestionario:

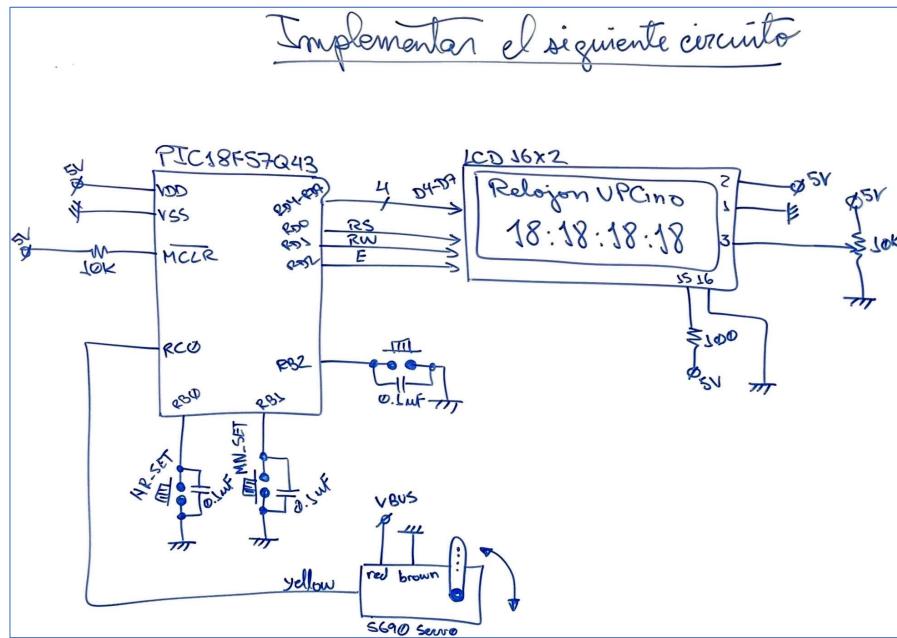
- Ya que se ha analizado el Timer0 y el Timer3 como fuente de tiempo para obtener los periodos de un servo. Es posible manipular dos servos, uno con el Timer0 y otro con el Timer3 junto con el manejo adecuado de las interrupciones.

63

# Ejercicio 2024-1

Implementar el siguiente circuito

- Hardware



64

## Movimiento del servo para el reloj para que funcione como un péndulo electrónico

- El presente código va a mover el servo entre 0º y 180º dependiendo si el valor de los segundos sea par o impar.
- Para ello se usa la operación de residuo de valor 2, si el resultado es cero significará que los segundos están en valor par.

```

if((segundos%2) == 0){
    LATCbits.LATC0 = 1;
    __delay_us(2000);
    LATCbits.LATC0 = 0;
    __delay_us(18000);
}
else{
    LATCbits.LATC0 = 1;
    __delay_us(1000);
    LATCbits.LATC0 = 0;
    __delay_us(19000);
}

```

65

## Movimiento del servo para que ubique el valor de los segundos del reloj en su eje de salida

- Dado que el movimiento del servo es de 180 grados, se divide en once posiciones donde cada posición comprenderá un rango de 5 segundos. Con esta propuesta el servo solo se moverá a uno de las once posiciones establecidas cada cinco segundos para prevenir un consumo excesivo de energía.
- Se contempla el uso de la función switch/case para seleccionar la posición que debe de moverse el servo

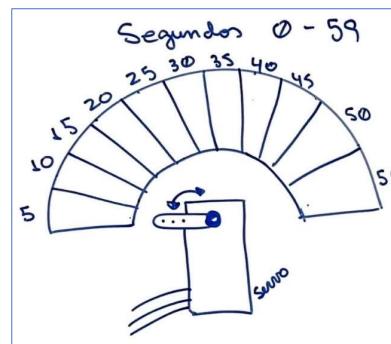


Diagrama que muestra el ciclo de trabajo del servo:

- 0.5ms - 2.5ms → (2ms de ancho)
- 0° - 180° → 180°/11 = 16.36° por sección
- 16.36° × 11 = 182.36° → 182.36°/1000ms = 182.36µs

```

switch(segundo){
    case 0 ... 4 :
        LATCbits.LC0=1;
        __delay_us(2500);
        LATCbits.LC0=0;
        __delay_us(17500);
        break;
    case 5 ... 9 :
        LATCbits.LC0=1;
        __delay_us(2500-(182*1));
        LATCbits.LC0=0;
        __delay_us(17500+(182*1));
        break;
    case 10 ... 14 :
        ...
}

```

66

## Movimiento del servo para que ubique el valor de los segundos del reloj en su eje de salida

```

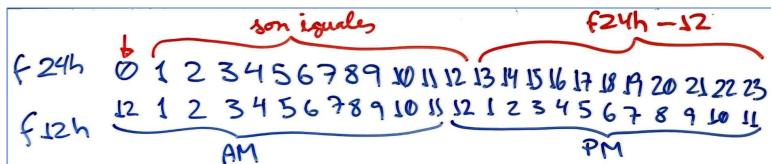
switch(segundos){
    case 0 ... 4:
        LATCbits.LATC0 = 1;
        __delay_us(2500);
        LATCbits.LATC0 = 0;
        __delay_us(17500);
        break;
    case 5 ... 9:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - 182);
        LATCbits.LATC0 = 0;
        __delay_us(17500 + 182);
        break;
    case 10 ... 14:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*2));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*2));
        break;
    case 15 ... 19:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*3));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*3));
        break;
    case 20 ... 24:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*4));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*4));
        break;
    case 25 ... 29:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*5));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*5));
        break;
    case 30 ... 34:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*6));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*6));
        break;
    case 35 ... 39:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*7));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*7));
        break;
    case 40 ... 44:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*8));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*8));
        break;
    case 45 ... 49:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*9));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*9));
        break;
    case 50 ... 54:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*10));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*10));
        break;
    case 55 ... 59:
        LATCbits.LATC0 = 1;
        __delay_us(2500 - (182*11));
        LATCbits.LATC0 = 0;
        __delay_us(17500 + (182*11));
        break;
}

```

67

## Cambio de formato a 12h a partir de formato de 24h

- Analizando los formatos de 12h y 24h tenemos los siguientes:



- Para la parte numérica verificamos lo siguiente:
  - Cuando en el formato de 24h la hora es cero, en el formato de 12h la hora es doce.
  - Cuando en el formato de 24h la hora es entre uno y doce, se mantiene el mismo valor en el formato de 12h.
  - Cuando en el formato de 24h la hora esta desde trece en adelante, en el formato de 12h se debe de restar el valor de doce.
- Para la parte de visualizar AM ó PM se tiene lo siguiente:
  - Cuando en el formato de 24h la hora es entre cero y once, en el formato de 12h se debe de mostrar AM.
  - Cuando en el formato de 24h la hora es entre doce y veintitrés, en el formato de 12h se debe de mostrar PM.

```

else if(formato == f_12h){
    POS_CURSOR(1,0);
    ESCRIBE_MENSAJE("Clock 12H uC2024",16);
    POS_CURSOR(2,2);
    if(horas == 0){
        ENVIA_CHAR(((horas + 12) / 10) + 0x30);
        ENVIA_CHAR(((horas + 12) % 10) + 0x30);
    }
    else if(horas>0 && horas<13){
        ENVIA_CHAR((horas / 10) + 0x30);
        ENVIA_CHAR((horas % 10) + 0x30);
    }
    else if(horas>12){
        ENVIA_CHAR(((horas - 12) / 10) + 0x30);
        ENVIA_CHAR(((horas - 12) % 10) + 0x30);
    }
    ENVIA_CHAR(':');
    convierte(minutos);
    ENVIA_CHAR(decena+0x30);
    ENVIA_CHAR(unidad+0x30);
    ENVIA_CHAR(':');
    convierte(segundos);
    ENVIA_CHAR(decena+0x30);
    ENVIA_CHAR(unidad+0x30);
    ENVIA_CHAR(' ');
    if(horas>=0 && horas<12){
        ESCRIBE_MENSAJE("AM",2);
    }
    else if(horas>=12){
        ESCRIBE_MENSAJE("PM",2);
    }
}

```

68

## Asignación Semana 11 2024-1

- Modificar el ejemplo desarrollado anteriormente para que mediante un pulsador adicional en RB2 y en interrupción externa INT2 permita intercambiar el formato de visualización de la hora entre 24h y 12h tal como se muestra en las siguientes imágenes. Tener en cuenta que en el formato 24h se debe de visualizar hasta las centésimas.



- Carpeta compartida: <https://bit.ly/3yHDrd>
- Formato de nombre de archivo de video de evidencia: EL256\_[tu sección]\_[tu primer apellido]\_[tu primer nombre]\_Sem11.mp4
  - Ejemplo de nombre de archivo de video de evidencia: EL256\_El59\_Perez\_Juan\_Sem11.mp4
- Video grabarlo en calidad 720p como máximo y 30 segundos de duración como máximo (presentación personal, mostrar su código en el MPLABX, mostrar su implementación y finalmente manipular los pulsadores para verificar los cambios).

69

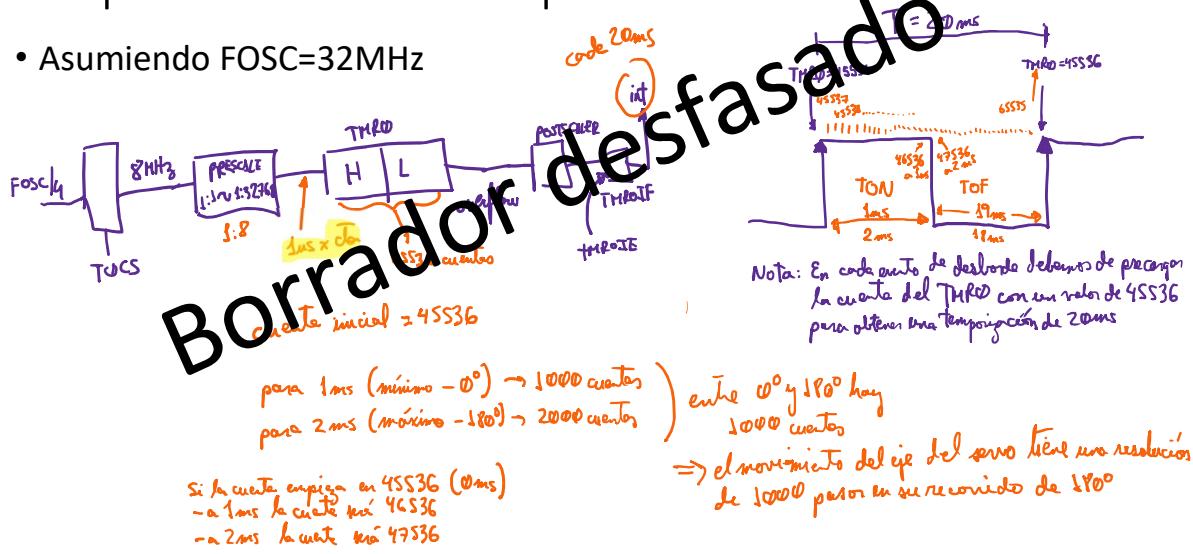
## Fin de la sesión

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

70

## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

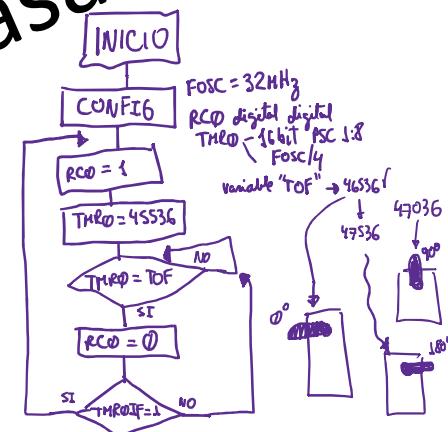
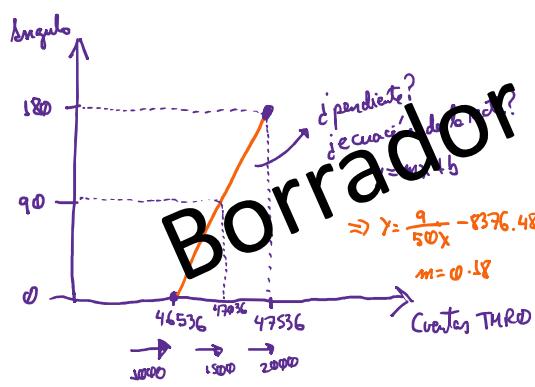
- Asumiendo FOSC=32MHz



71

## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

- Escalamiento ángulo de servo vs cuentas del Timer0



72

