

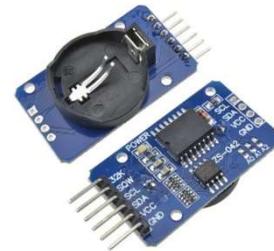
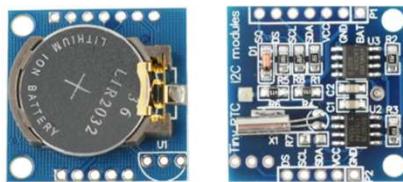
# Microcontroladores

Semana 11

Por Kalun José Lau Gan

## ¿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!

## Preguntas previas

- ¿Veremos manipulación de servos?
  - Si, hoy atenderemos ese tema
- **¿Cuáles son los temas pendientes en el curso?**
  - Comunicación serial I2C, SPI (Sem 12)
  - Comunicación serial UART (Sem 13)
  - Protocolos e interfaces propietarias (sensores DHT11, LEDs Neopixel WS2812, DS18B20 1-wire), Ultrasonido HC-SR04 (Sem 14)
  - Otras interfaces y protocolos (RS485, CAN, Bluetooth, Wi-Fi, ESP-Now, etc) (Sem 14)
  - PCB (CNC milling) y modelado 3D y manufactura aditiva (Sem 15)
  - Micros de 32bits (ESP32, Rpi-Pico, STM32, PIC32) (Sem 15)
- **¿Cuáles son las evaluaciones pendientes en el curso?**
  - Según el sílabo: LB3 (Sem12), PC2(Sem14), DD(Sem15) y TF(Sem16)

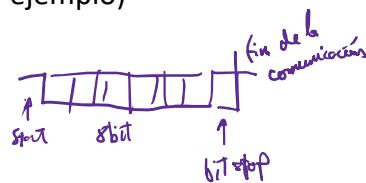
## Preguntas previas

- ¿Cómo se transmiten los datos en el UART?
  - Usando el registro UxTXB (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);

```



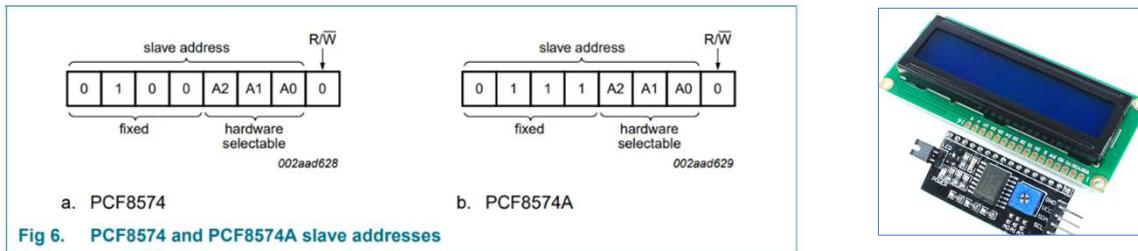
```

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);

```

## Preguntas 2024-2

- Puedo usar LiquidCrystal para el PIC18F57Q43?
  - Es una librería desarrollada para la plataforma Arduino, es incompatible con lo que se desarrolla en XC8, se tiene que reinterpretar la librería para el PIC18F57Q43.
  - Revisar repositorio del profesor Kalun para mayor información de librerías y casos
- ¿A qué se refiere con la dirección esclavo en el I2C?
  - Se refiere a la dirección única del dispositivo I2C a conectarse, si se usa varios dispositivos se debe de revisar que no haya conflicto de direcciones



## Preguntas 2025-2

- ¿Algún programa simple de aprender para desarrollar PCBs?
  - KiCAD (gratuito)
  - EasyEDA (gratuito y online)
  - Autodesk Eagle (gratis previo registro con cuenta de correo académico)
  - Autodesk Fusion (gratis previo registro con cuenta de correo académico)
  - FluxAI (de pago)
  - Proteus ARES (de pago)
  - Altium Designer (de pago)
- ¿Cuál es la temperatura recomendada para soldar con cautín?
  - 340°C y 360° para componentes thru-hole, 320°C para SMD, siempre que se trabaje con soldadura 60/40. Se recomienda utilizar FLUX

## 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

## Interrupciones vectorizadas a detalle

- Hay una tabla de interrupciones vectorizadas (IVT) contemplado en la tabla 11.1 de la hoja técnica.
- Cada fuente de interrupción que posee el microcontrolador se encuentra en una dirección en particular según

$$\text{IVTBASE} + 2 \times (\text{número de vector})$$

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

Vector Number	Interrupt source	Vector Number	Interrupt source (cont.)
0x0	Software Interrupt	0x3E	PWM0INT
0x1	HLOV (High/Low-Voltage-Detected)	0x3F	PWM0SINT
0x2	CSP (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-Cross Detection)	0x47	SCAN
0xA	AD (ADC Conversion Complete)	0x48	U3RX
0xB	ACT (Auto Clock Tuning)	0x49	U3TX
0xC	CAN (Controller Area Network)	0x4A	U3E
0xD	GMT1 (Signal Measurement Timer)	0x4B	U3
0xE	SMT1PRA	0x4C	-
0xF	SMT1PWA	0x4D	CLC4
0x10	ADT	0x4E - 0x4F	-
0x11 - 0x13	-	0x50	INT2
0x14	DMA1SCNT (Direct Memory Access)	0x51	CLC5
0x15	DMA1DCNT	0x52	CWG2 (Complementary Waveform Generator)
0x16	DMA1OR	0x53	NC02
0x17	DMA1A	0x54	DMA3SCNT
0x18	SP1RX (Serial Peripheral Interface)	0x55	DMA3DCNT
0x19	SP1TX	0x56	DMA3A
0x1A	SP1I	0x57	DMA3A
0x1B	TMR2	0x58	CCP3
0x1C	TMR1	0x59	CLC6
0x1D	TMR1G	0x5A	CWG3
0x1E	CCP1 (Complementary Waveform)	0x5B	TMR4
0x1F	TMR5	0x5C	DMA4DCNT
0x20	UIRX	0x5D	DMA4DCNT
0x21	UITX	0x5E	DMA4OR
0x22	U1E	0x5F	DMA4A
0x23 - 0x25	U1	0x60	U4RX
0x26	PWM1RINT	0x61	U4TX
0x27	PWM1GINT	0x63	U4
0x28	SP2RX	0x64	DMA5SCNT
0x29	SP2TX	0x65	DMA5DCNT
0x2A	SP2I	0x66	DMA5A
0x2C	TMR3	0x68	U5RX
0x2D	TMR2G	0x69	U5TX
0x2E	PWM2RINT	0x6A	U5E
0x2F	PWM2GINT	0x6B	U5
0x30	INT1	0x6C	DMA6SCNT
0x31	CLC2	0x6D	DMA6DCNT
0x32	CWG1 (Complementary Waveform Generator)	0x6E	DMA6OR
0x33	NC01 (Numerically Controlled Oscillator)	0x6F	DMA6A
0x34	DMA3SCNT	0x70	-
0x35	DMA3DCNT	0x71	CLC7
0x36	DMA4OR	0x72	CM2
0x37	DMA2A	0x73	NC03
0x38	I2C1RX	0x74 - 0x77	-
0x39	I2C1TX	0x78	NMI
0x3A	I2C1I	0x79	CLC8
0x3B	I2C1E	0x7A	CRC (Cyclic Redundancy Check)
0x3C	-	0x7B	TMR6
0x3D	CLC3	0x7C - 0x8F	-

## Interrupciones vectorizadas a detalle

- Plantilla de funciones de interrupciones en XC8 en alto nivel:

```

//----- un sola 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){
//-
} 1BH

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

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

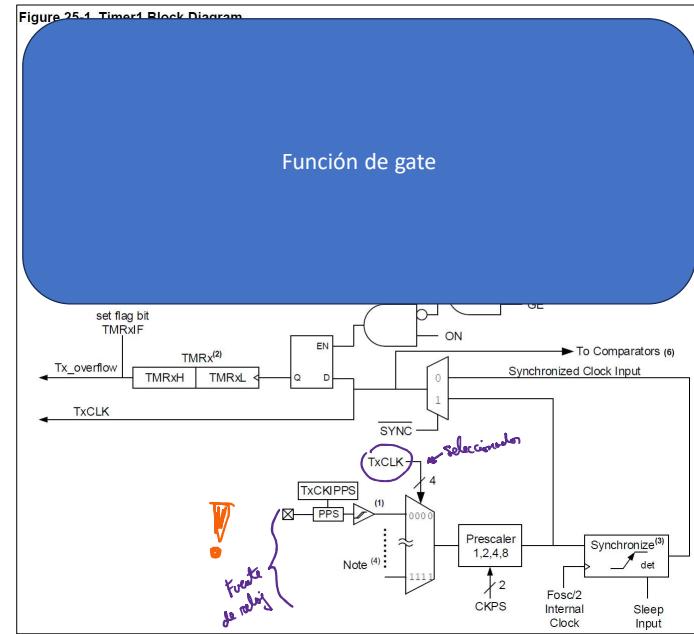
void __interrupt(irq(IRQ_INT2) INT2_ISR(void){
//-
}

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

```

## El Timer 1 en el PIC18F57Q43

- Módulo temporizador de **16 bits**
- Empleado para aplicaciones de RTC (Real-time clock)
- Función de gate
- Funciona con el CCP1 (modo captura ó modo comparación)
- Múltiples fuentes de reloj
- Si se va a emplear HFINTOSC, no emplear 12MHz ni 48MHz
- El registro de la cuenta del TMR1 esta desplegado en dos registros: TMR1H:TMR1L
- Similar para Timer3 y Timer5



## El Timer 1 en el PIC18F57Q43

- Selección de la fuente de reloj (registro T1CLK)
- Dos opciones para RTC:
  - Cristal de 32.768KHz (configurar como oscilador externo – en el PIC18F57Q43 es el oscilador secundario)
  - Oscilador principal del microcontrolador (uno externo ó el HFINTOSC)



Timer Clock Source Selection Register								
Bit	7	6	5	4	3	2	1	0
Access				R/W	R/W	R/W	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	Clock Source	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			F <sub>osc</sub>					
00001			F <sub>osc</sub> 4 ✓					
00000	Pin selected by T1CKIPPS		Pin selected by T3CKIPPS			Pin selected by T5CKIPPS		

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

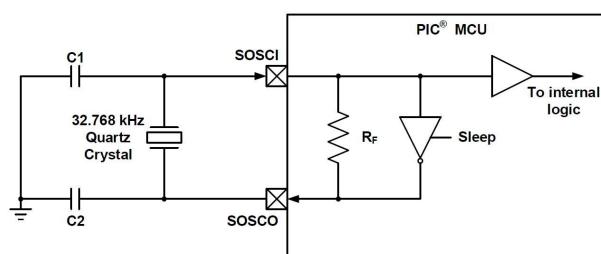
## El Timer 1 en el PIC18F57Q43

- Oscilador secundario

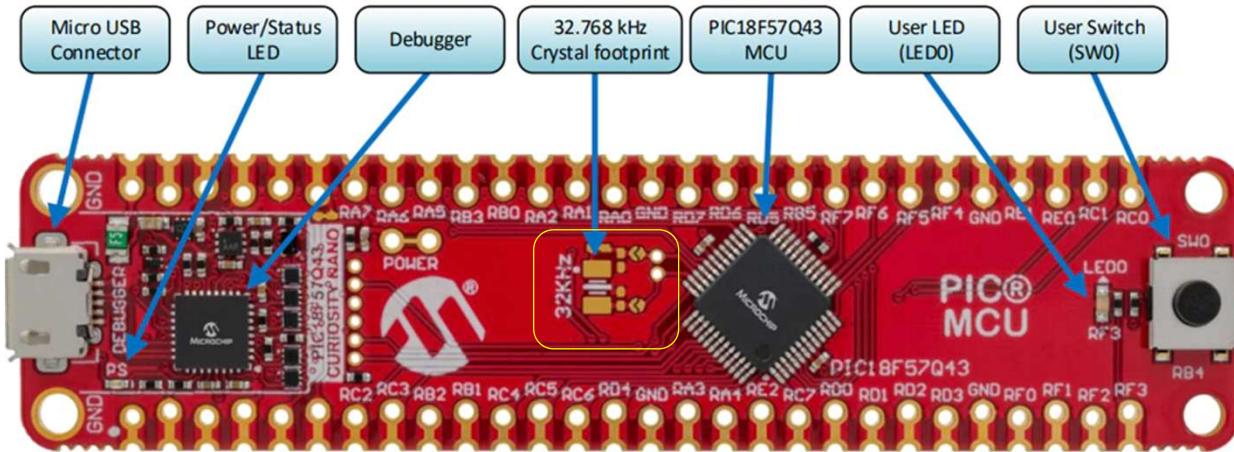


### 12.1.1.5 Secondary Oscillator

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.



## Montaje del cristal de 32.768KHz en el Curiosity Nano:



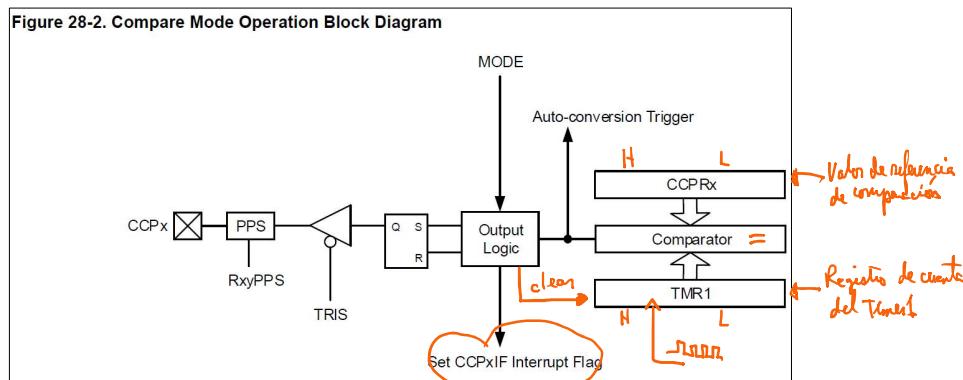
## El CCPx en el PIC18F57Q43

- Módulo captura/comparación/PWM (CCPx)

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

## El CCPx en el PIC18F57Q43

- El CCP1 posee tres modos de trabajo: Captura, Comparación y PWM
- CCP1 en modo comparación, evento especial de disparo se empleará para hacer la tarea de RTC junto con el Timer1
- Diagrama de bloques



## El CCPx 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.

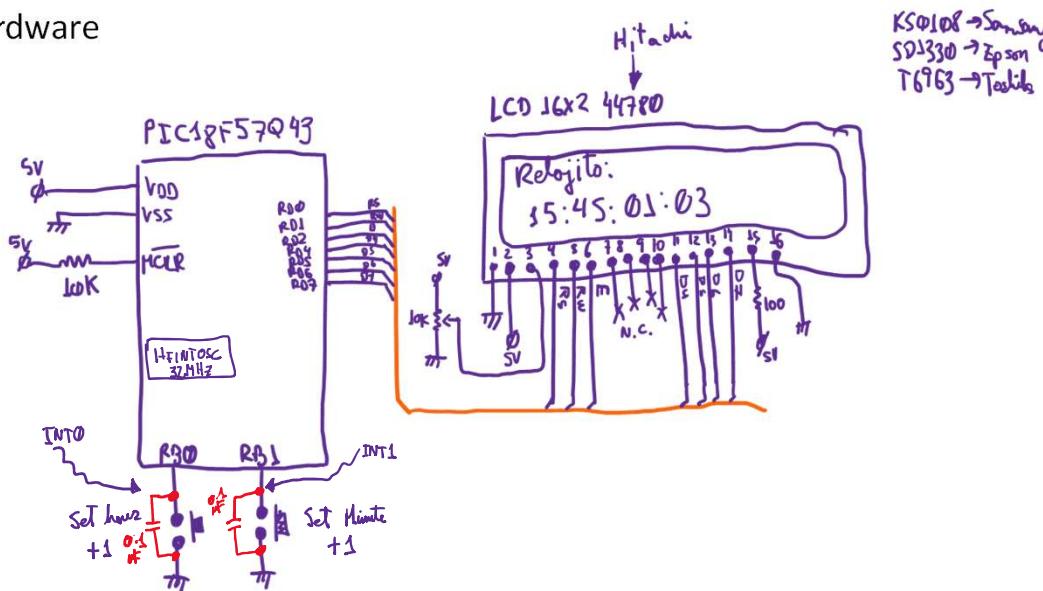
$$f_{osc} \rightarrow 32.768 \text{ KHz}$$

A revisar en la hoja técnica:

- Timer1:
  - Diagrama de bloques
  - Registros implicados (T1CON, etc)
- CCPx:
  - Diagrama de bloques
  - Registros implicados (CCP1CON, etc)
  - Configuración para que se active su interrupción (los registros donde están CCP1IE, CCP1IF, GIE)

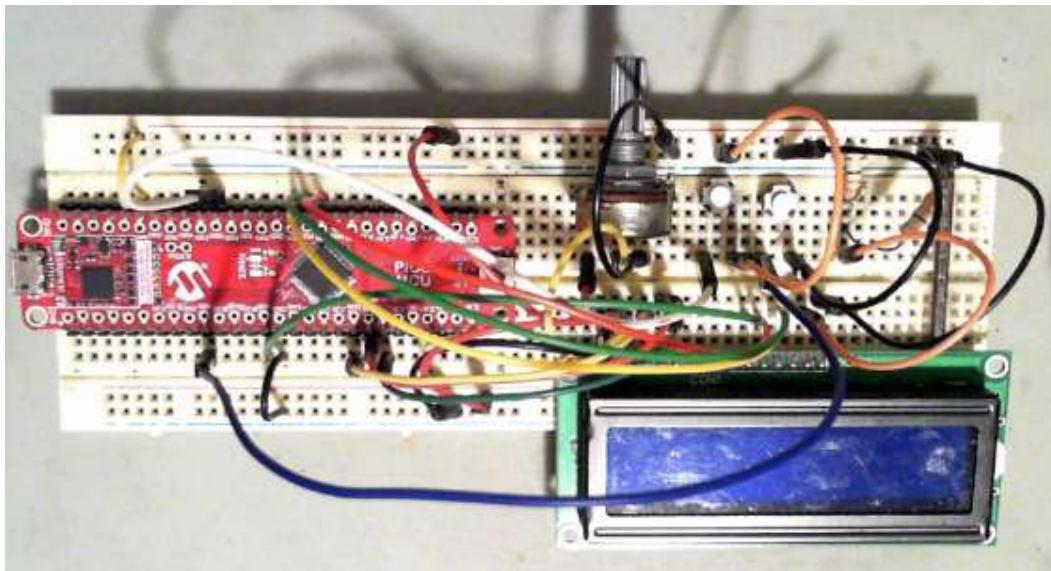
### Ejercicio: Reloj empleando el Timer1

- Hardware



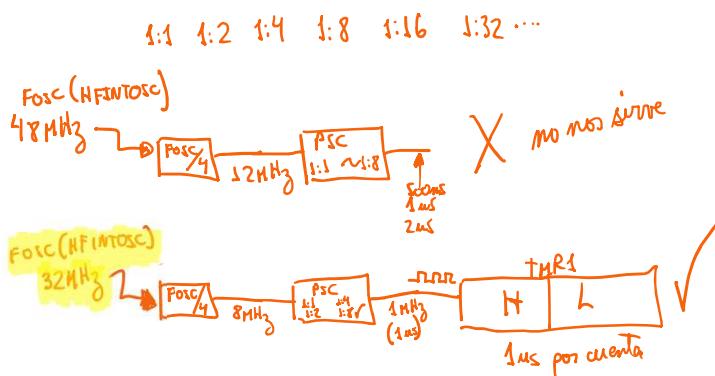
## Ejercicio: Reloj empleando el Timer1

- Hardware



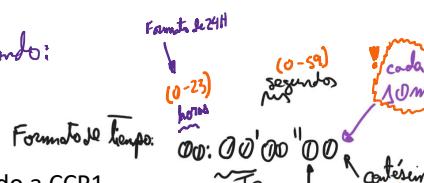
## Ejercicio: Reloj empleando el Timer1

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

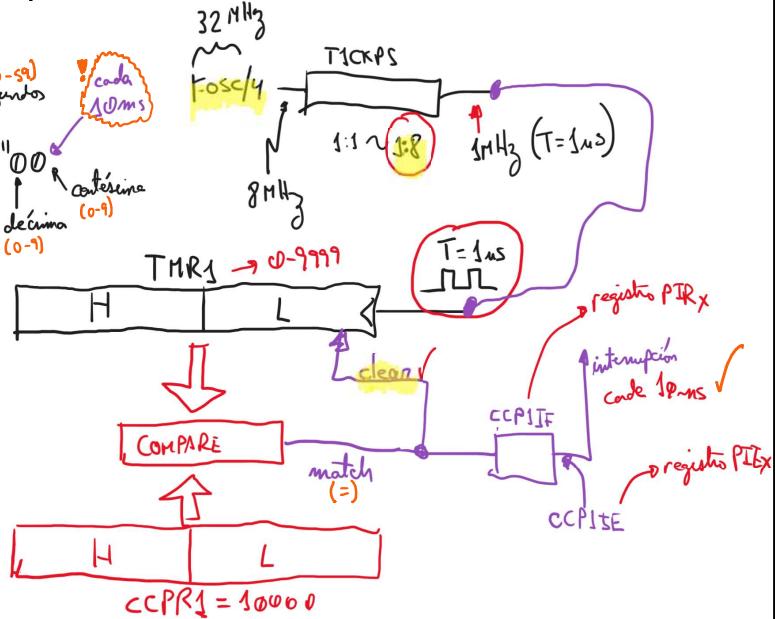


## Ejercicio: Reloj empleando el Timer1

Analizando:

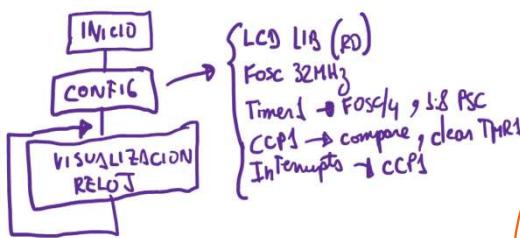


- Timer 1 conectado a CCP1
  - CCP1 en modo comparador evento especial de disparo
  - FOSC (HFINTOSC) debe de configurarse a 32MHz
  - Prescaler del Timer1: 1:8
  - Valor de referencia de comparación del CCP1: 10000
  - Cuando cuentas del Timer1 llegue a 10000 ocurrirá una igualdad y será reiniciada la cuenta, por lo tanto el renglón de cuentas será de 0000 a 9999
  - Cuando ocurra el match se levantará la bandera CCP1IF y generará una interrupción, esto sucederá de manera repetitiva y regular cada 10ms.

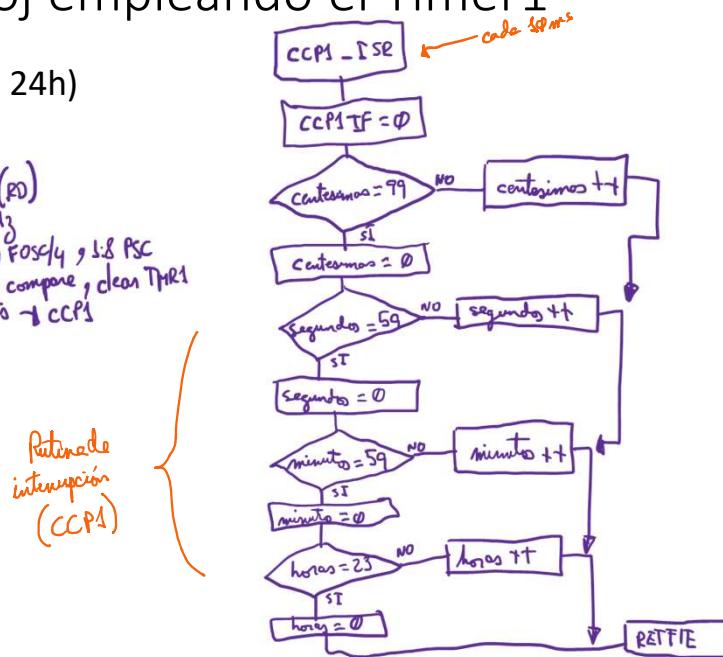


## Ejercicio: Reloj empleando el Timer1

- Algoritmo (formato de 24h)



00:00'00"00  
23:59'59"99  
HH:MM'SS"D



## Ejercicio: Reloj empleando el Timer1

- Cuatro tareas principales a cumplir con respecto a la base de tiempo:
  - Base de tiempo: HFINTOSC a 32MHz
  - Configurar el Timer1 (FOSC/4, 1:8 PSC)
  - Configurar el CCP1 (compare mode) y establecer valor de referencia de comparación a 10000 (10ms)
  - Configurar las interrupciones (del CCPx)

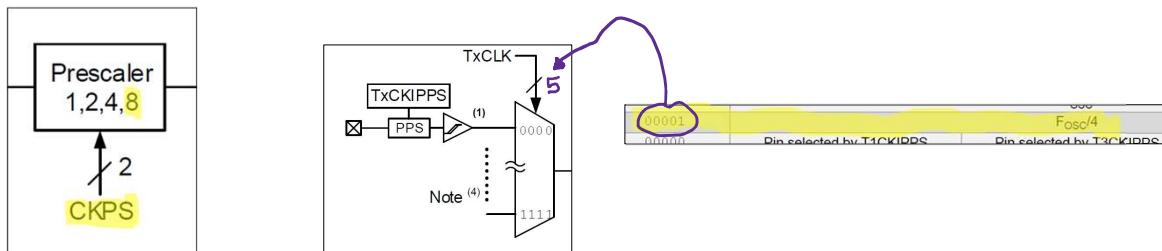
## Ejercicio: Reloj empleando el Timer1

- Configuración de la fuente de reloj: HFINTOSC a 32MHz:
  - OSCCON1 = 0x60; //NOSC HFINTOSC, NDIV 1:1
  - OSCFRQ = 0x06; //HFINTOSC a 32MHz
  - OSCEN = 0x40 //HFINTOSC enabled

## Ejercicio: Reloj empleando el Timer1

- Configuración del Timer1:

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



## Ejercicio: Reloj empleando el Timer1

- Registro T1CLK

$$\text{T1CLK} = 0x\Phi_1$$

↑  
Source Fosc/4

Name: TxCLK Address: 0x321,0x32D,0x339			
Timer Clock Selection Register			
Bit	7	6	5
Access	0	0	0
Reset	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			
10101		Reserved	
10100		CLC8_OUT	
10011		CLC7_OUT	
10010		CLC6_OUT	
10001		CLC5_OUT	
10000		CLC4_OUT	
01111		CLC3_OUT	
01110		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		HFIINTOSC	
00010		Fosc	
00001		Fosc/4	
00000	Pin selected by T1CKIPPS	Pin selected by T3CKIPPS	Pin selected by T5CKIPPS

## Ejercicio: Reloj empleando el Timer1

- Registro T1CON:

$$T1CON = 0x33$$

Prescaler 1:8

Timer1 ON

Timer Control Register									
Bit	7	6	5	CKPS[1:0]	4	3	2	SYNC	RD16
Access				R/W	R/W		X	R/W	R/W
Reset				0	0		0	0	0
<b>Bits 5:4 – CKPS[1:0] Timer Input Clock Prescaler Select</b>									
Reset States: POR/BOR = 00									
All Other Resets = uu									
<b>Value Description</b>									
11 1:8 Prescaler value									
10 1:4 Prescaler value									
01 1:2 Prescaler value									
00 1:1 Prescaler value									
<b>Bit 2 – SYNC Timer External Clock Input Synchronization Control</b>									
Reset States: POR/BOR = 0									
All Other Resets = u									
<b>Value Condition Description</b>									
x CS = F <sub>osc</sub> /4 or F <sub>osc</sub>									
1 All other clock sources									
0 All other clock sources									
<b>Bit 1 – RD16 16-Bit Read/Write Mode Enable</b>									
Reset States: POR/BOR = 0									
All Other Resets = u									
<b>Value Description</b>									
1 Enables register read/write of Timer in one 16-bit operation									
0 Enables register read/write of Timer in two 8-bit operations									
<b>Bit 0 – ON Timer On</b>									
Reset States: POR/BOR = 0									
All Other Resets = u									
<b>Value Description</b>									
1 Enables Timer									
0 Disables Timer									

## Ejercicio: Reloj empleando el Timer1

- Configuración del CCP1:

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

CCPR1H = 0x27  
 CCPR1L = 0x10  
 ~~~~~ 10000 en decimal

CCPR1 = 10000; X

Modo de comparación  
 } valor de referencia del comparador

## Ejercicio: Reloj empleando el Timer1

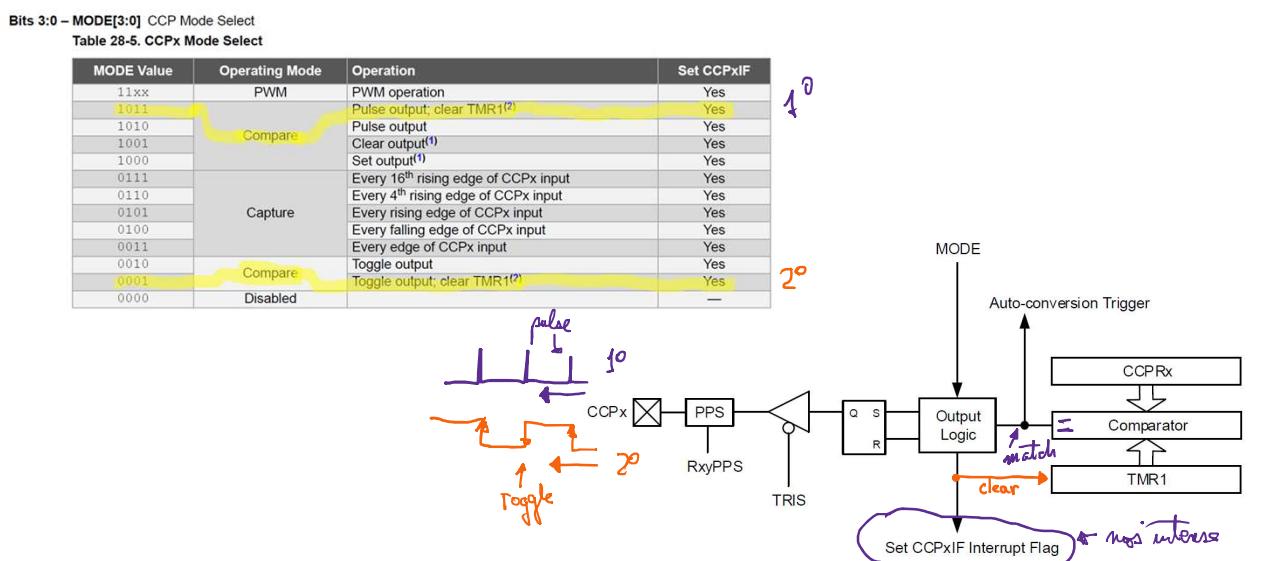
- Registro CCP1CON:

*CCP1CON = 0x81*

| CCP Control Register                                   |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
|--------------------------------------------------------|-----------------------|--------------------------------------------------|----------------------|-----|---|-------|---|---|-----|--|--|--|--|--|--|--|--|--|
| Bit                                                    | 7                     | 6                                                | 5                    | 4   | 3 | 2     | 1 | 0 |     |  |  |  |  |  |  |  |  |  |
| Access                                                 | R/W                   | EN                                               | OUT                  | FMT | 0 | MODES | 0 | 0 | R/W |  |  |  |  |  |  |  |  |  |
| Reset                                                  | 0                     | x                                                | 0                    | 0   | 0 | 0     | 0 | 0 | 0   |  |  |  |  |  |  |  |  |  |
| <b>Bit 7 – EN CCP Module Enable</b>                    |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| Value                                                  | Description           |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| 1                                                      | CCP is enabled        |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| 0                                                      | CCP is disabled       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| <b>Bit 5 – OUT CCP Output Data (read-only)</b>         |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| <b>Bit 4 – FMT CCPxRH:L Value Alignment (PWM mode)</b> |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| 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 |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| <b>Bits 3:0 – MODE[3:0] CCP Mode Select</b>            |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| Table 28-5. CCPx Mode Select                           |                       |                                                  |                      |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| <b>MODE Value</b>                                      | <b>Operating Mode</b> | <b>Operation</b>                                 | <b>Set CCPxIF</b>    |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |
| 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                                                   | Capture               | 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              |                                                  | —                    |     |   |       |   |   |     |  |  |  |  |  |  |  |  |  |

## Ejercicio: Reloj empleando el Timer1

- Registro CCP1CON:



# Ejercicio: Reloj empleando el Timer1

- Configuración de las interrupciones del CCP1:

|        |      |     |        |               |         |        |        |        |          |          |
|--------|------|-----|--------|---------------|---------|--------|--------|--------|----------|----------|
| 0x04A1 | PIE3 | 7:0 | TMROIE | <b>CCP1IE</b> | TMR1GIE | TMR1IE | TMR2IE | SPI1IE | SPI1TXIE | SPI1RXIE |
|--------|------|-----|--------|---------------|---------|--------|--------|--------|----------|----------|

1 → habilitar la int del CCP1

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

1 → habilitar global de bits

|        |      |     |        |               |         |        |        |        |          |          |
|--------|------|-----|--------|---------------|---------|--------|--------|--------|----------|----------|
| 0x04B1 | PIR3 | 7:0 | TMROIF | <b>CCP1IF</b> | TMR1GIF | TMR1IF | TMR2IF | SPI1IF | SPI1TXIF | SPI1RXIF |
|--------|------|-----|--------|---------------|---------|--------|--------|--------|----------|----------|

bandera

```

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=55;
7 unsigned char centena,decena,unidad;
8
9 void configuro(void){
10    //configuracion del oscilador
11    OSCCON1 = 0x60;
12    OSCFRC = 0x06; //HFINTOSC a 32MHz
13    OSCEN = 0x40;
14    //configuracion del Timer1
15    T1CLK = 0x01;
16    T1CON = 0x33;
17    //configuracion del CCP1
18    CCPICON = 0x81;
19    CCPRIH = 0x27;
20    CCPRLH = 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 }
```

**Ejercicio: Reloj empleando el Timer1**

```

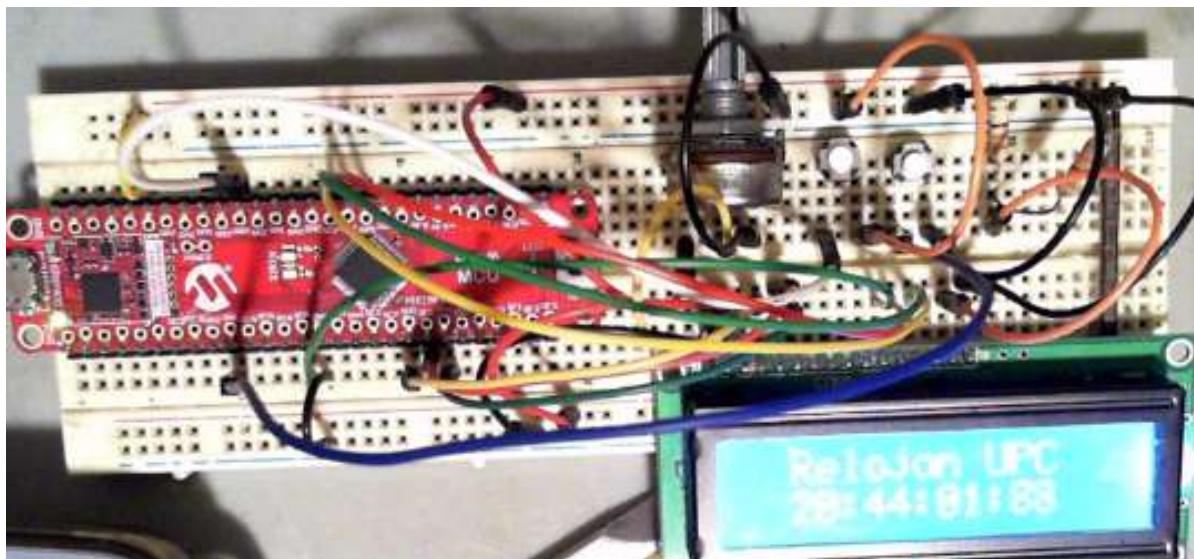
36 void convierte(unsigned char numero){
37    centena = numero / 100;
38    decena = (numero % 100) / 10;
39    unidad = numero % 10;
40 }
41
42 void main(void) {
43    configuro();
44    lcd_init();
45    POS_CURSOR(1,2);
46    ESCRIBE_MENSAJE("Reloj en UPC",11);
47    while(1){
48        POS_CURSOR(2,2);
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 }
```

void \_\_interrupt(irq(IRQ\_CCPI)) CCP1\_ISR(void){  
 PIR3bits.CCP1IF = 0;  
 if(centesimas == 99){  
 centesimas = 0;  
 if(segundos == 59){  
 segundos = 0;  
 if(minutos == 59){  
 minutos = 0;  
 if(horas == 23){  
 horas = 0;  
 } else{  
 horas++;  
 }  
 } else{  
 minutos++;  
 }  
 } else{  
 segundos++;  
 }  
 } else{  
 centesimas++;  
 }  
}

void \_\_interrupt(irq(default)) DEFAULT\_ISR(void){  
 {  
 // Unhandled interrupts go here  
 }  
}

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

## Ejercicio: Reloj empleando el Timer1



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

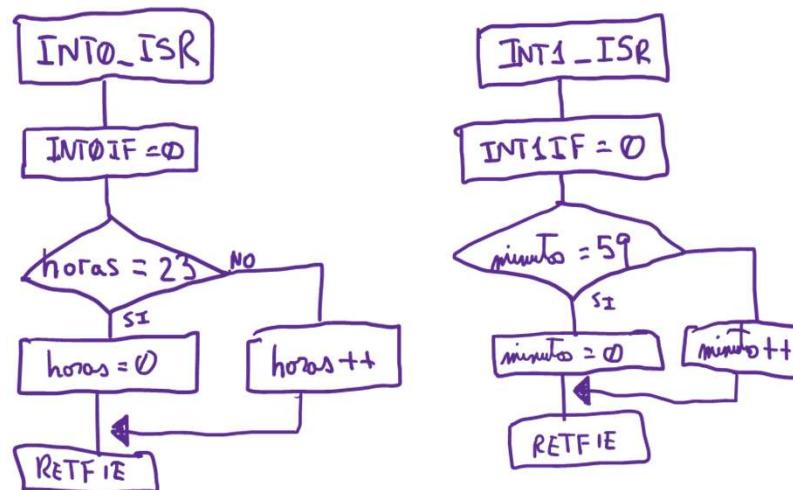


## Ejercicio: Reloj empleando el Timer1

- 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
  - Hay que establecer RB0 e RB1 como entradas digitales (TRISB.0 y TRISB.1 en uno, ANSELB.0 y ANSEBL.1 en cero)
  - 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).
  - Recordar que por defecto las interrupciones están en modo vectorizado
  - Por defecto la INT0 esta para RB0 e INT1 para RB1, de cambiarse los pines se tendrá que configurar INT0PPS e INT1PPS respectivamente.

## Ejercicio: Reloj empleando el Timer1

- Etapa de ajuste de las horas y minutos empleando los pulsadores en RB0/INT0 y RB1/INT1



## 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:

- Handwritten notes: "1 para entrada" with arrows pointing to TRISB1 and ANSELB1.
- Handwritten notes: "0 para digital" with arrows pointing to TRISB0 and ANSELB0.
- Handwritten notes: "1 para hoklite, weak pull-up" with arrows pointing to WPUB0.

## 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      | SMT1PWAIF | SMT1PRAIF | SMT1IF     | CM1IF      | ACTIF   | ADIF    | ZCDIF   | INT0IF |
| 0x04B4  | PIR6    | 7:0      | DMA2AIF   | DMA2ORIF  | DMA2DCNTIF | DMA2SCNTIF | NCO1IF  | CWG1IF  | CLC2IF  | INT1IF |

Annotations:

- Handwritten notes: "habilitar INT0" with an arrow pointing to INT0IE.
- Handwritten notes: "habilitar INT1" with an arrow pointing to INT1IE.
- Handwritten notes: "detección flanco en INT1" with an arrow pointing to INT1EDG.
- Handwritten notes: "detección flanco en INT0" with an arrow pointing to INT0EDG.
- Handwritten notes: "bandera de INT0" with an arrow pointing to INT0IF.
- Handwritten notes: "bandera de INT1" with an arrow pointing to INT1IF.

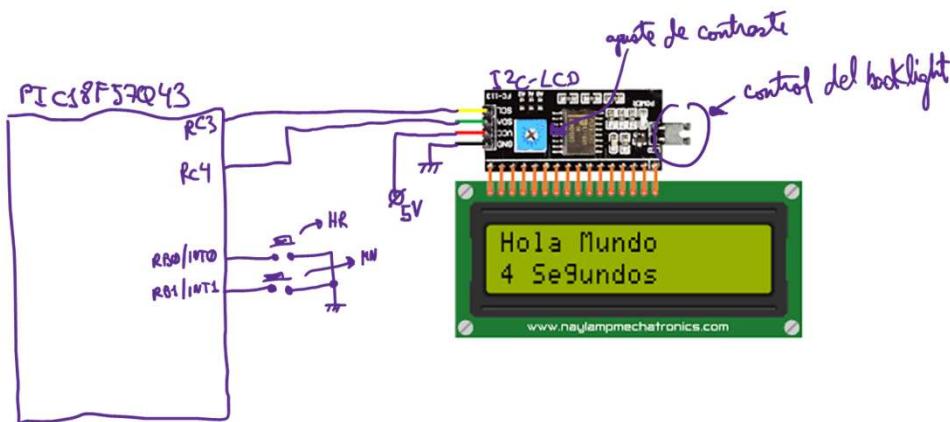
## Ejercicio: Reloj empleando el Timer1

```

1 #include <xc.h>
2 #include "cabecera.h"
3 #include "LCD.h"
4 #define _XTAL_FREQ 32000000UL
5
6 unsigned char horas=10,minutos=10,segundos=10,centesimas=10;
7 unsigned char centena,decena,unidad;
8
9 void configuro(void){
10     //configuración del oscilador
11     OSCCON1 = 0x60; //HFINTOSC, posts 1:1
12     OSCFRC = 0x06; //HFINTOSC a 32MHz
13     OSCEN = 0x40; //HFINTOSC enabled
14     //configuración de E/S
15     TRISB = 0xFF; //RB1 RB0 como entradas
16     ANSELB = 0xFC; //RB1 RB0 como digitales
17     WPUB = 0x03; //RB1 RB0 pullup enabled
18     //configuración del Timer1
19     T1CKL = 0x01; //clk source fosc/4
20     T1CON = 0x33; //TMR1 ON, pres 1:8
21     //configuración del CCP1
22     CCPICON = 0x81; //compare mode, clear TMRI
23     CCPRH = 0x27;
24     CCPRL = 0x10; //valor de referencia 10000
25     //configuración de las interrupciones
26     INTCONbits.INT0EGD = 0; //falling edge en INT0
27     INTCONbits.INT1EGD = 0; //falling edge en INT1
28     PIE1bits.INT0IE = 1; //INT0 enabled
29     PIE6bits.INT1IE = 1; //INT1 enabled
30     PIE3bits.CCPIE = 1; //CCPI enabled
31     PIRbits.INT0IF = 0; //flag INT0 bajada
32     FIRbits.INT1IF = 0; //flag INT1 bajada
33     FIR3bits.CCP1IF = 0; //flag CCP1 bajada
34     INTCONbits.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 }
44
45     CURSOR_HOME();
46     CURSOR_ONOFF(OFF);
47 }
48
49 void convierte(unsigned char numero){
50     centena = numero / 100;
51     decena = (numero % 100) / 10;
52     unidad = numero % 10;
53 }
54
55 void main(void) {
56     configuro();
57     lcd_init();
58     POS_CURSOR(1,2);
59     ESCRIBE_MENSAJE("Reloj en UPC",11);
60     while(1){
61         POS_CURSOR(2,2);
62         convierte(horas);
63         ENVIA_CHAR(decena+0x30);
64         ENVIA_CHAR(unidad+0x30);
65         ENVIA_CHAR(':' );
66         convierte(minutos);
67         ENVIA_CHAR(decena+0x30);
68         ENVIA_CHAR(unidad+0x30);
69         ENVIA_CHAR(':' );
70         convierte(centesimas);
71         ENVIA_CHAR(decena+0x30);
72         ENVIA_CHAR(unidad+0x30);
73         ENVIA_CHAR(':' );
74         ENVIA_CHAR(unidad+0x30);
75     }
76 }
77
78 void __interrupt(irq(IRQ_CCPI)) CCP1_ISR(void){
79     PIR3bits.CCP1IF = 0;
80     if(centesimas == 99){
81         centesimas = 0;
82         if(segundos == 59){
83             segundos = 0;
84             if(minutos == 59){
85                 minutos = 0;
86                 if(horas == 23){
87                     horas = 0;
88                 } else{
89                     horas++;
90                 }
91             } else{
92                 minutos++;
93             }
94         } else{
95             segundos++;
96         }
97     }
98 }
99
100 void __interrupt(irq(IRQ_INTO)) INTO_ISR(void){
101     PIR1bits.INT0IF = 0;
102     if(horas == 23){
103         horas = 0;
104     } else{
105         horas++;
106     }
107 }
108
109 void __interrupt(irq(IRQ_INT1)) INT1_ISR(void){
110     PIR6bits.INT1IF = 0;
111     if(minutos == 59){
112         minutos = 0;
113     } else{
114         minutos++;
115     }
116 }
117
118 void __interrupt(irq(default)) DEFAULT_ISR(void){
119 }
120
121
122
123
124 }
125
126 void __interrupt(irq(default)) UNHANDLED_ISR(void){
127 }
128 // Unhandled interrupts go here
129 }
```

• Código completo final:

## Mejora: empleo del modulo I2C-LCD

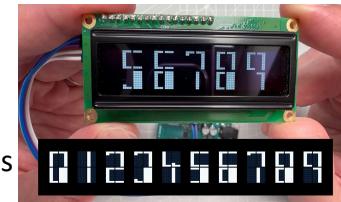


- Nota: Se debe de emplear la librería I2C\_LCD (revisar el repositorio)

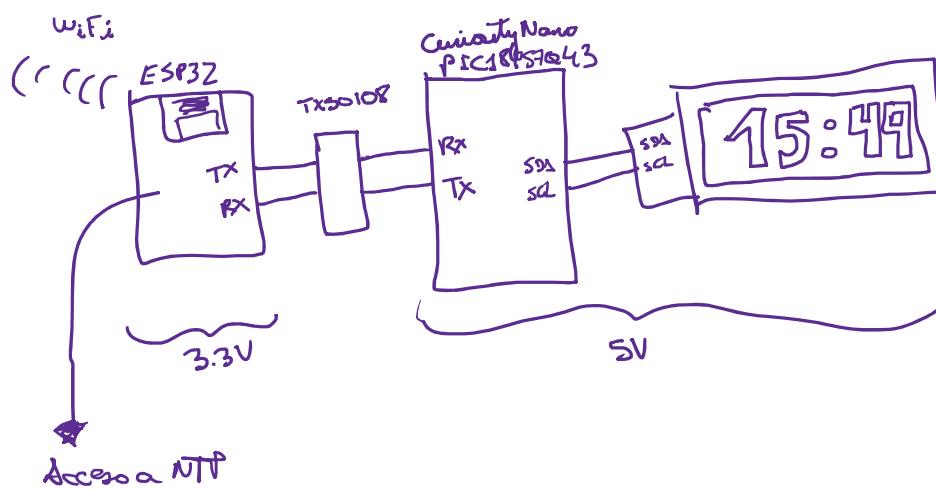
## Ejercicio: Reloj empleando el Timer1

- 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



### Adicional: Conectarlo el reloj a NTP



## El servo

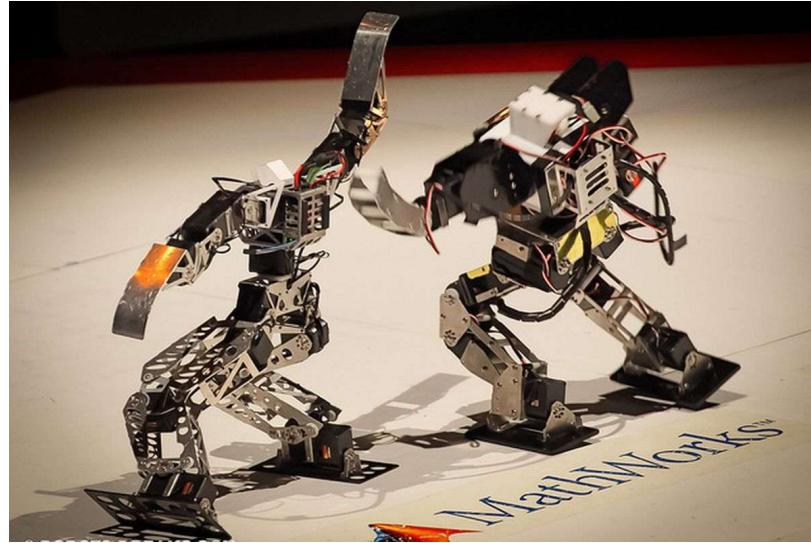
- Elemento electromecánico realimentado (posee un lazo de control cerrado)
- Rango de movimiento: 180 grados (no da la vuelta completa!)
- Empleado comúnmente en hobby para radiocontrol de vehículos terrestres, aéreos y acuáticos, para el control de posición (ángulo)
  - Acelerador, freno, dirección vehicular, alerones (flaps), timón, robótica.
- Son clasificados por: tamaño, torque, velocidad, precisión, tamaño



## Servos en el hobby profesional

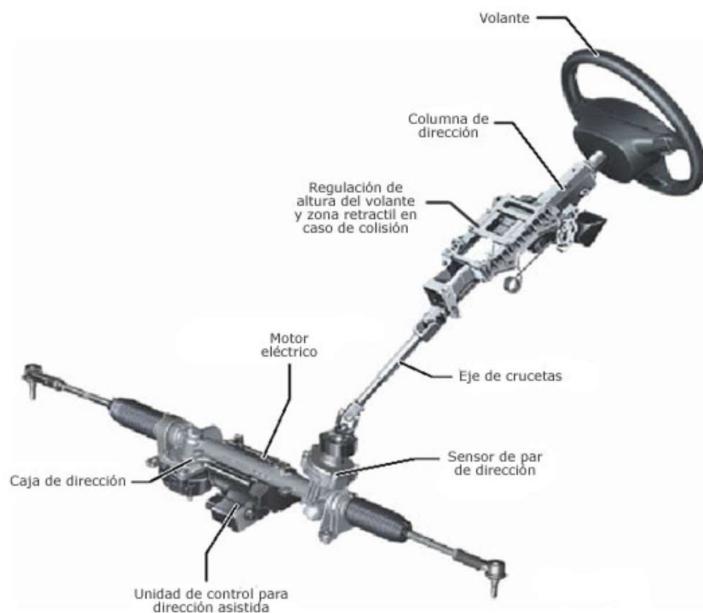


## Servos en el hobby profesional



## Dirección servoasistida

Componentes de la dirección electromecánica



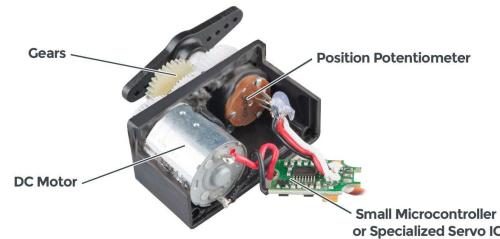
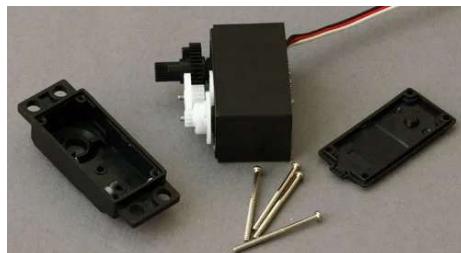
[www.mecanicavirtual.org](http://www.mecanicavirtual.org)

## Tren de aterrizaje de un dron Shahed 136



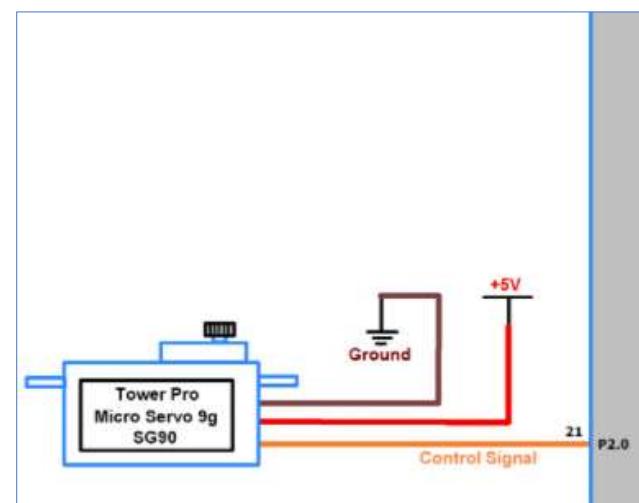
## El servo

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



## El servo

- Conexión con el microcontrolador:



## Specs del miniservo sg90:

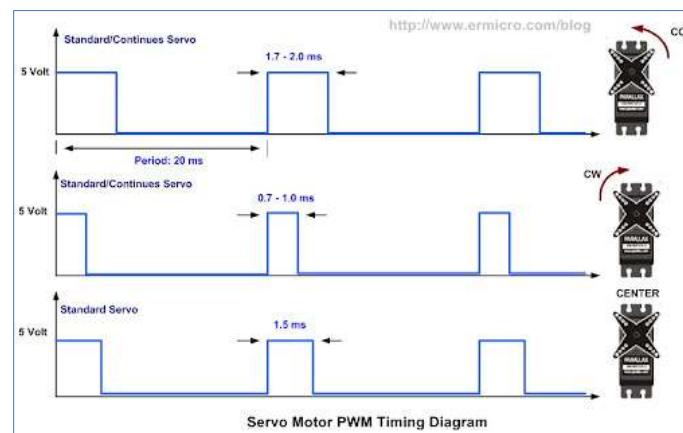
- De preferencia usar fuente externa para los servomecanismos

### SPECIFICATIONS:

|                                            | Operating Voltage          |                            |
|--------------------------------------------|----------------------------|----------------------------|
|                                            | 4.8V                       | 6.0V                       |
| <b>Stall Torque</b>                        | 1.3 kg/cm<br>(18.09 oz/in) | 1.5 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)           |                            |

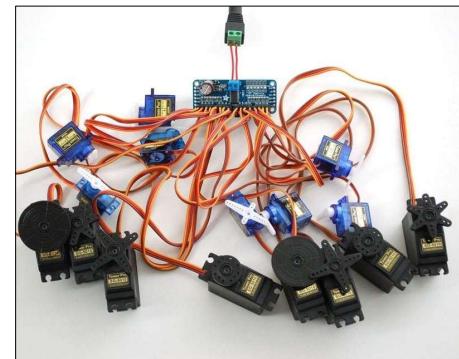
## El servo

- 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

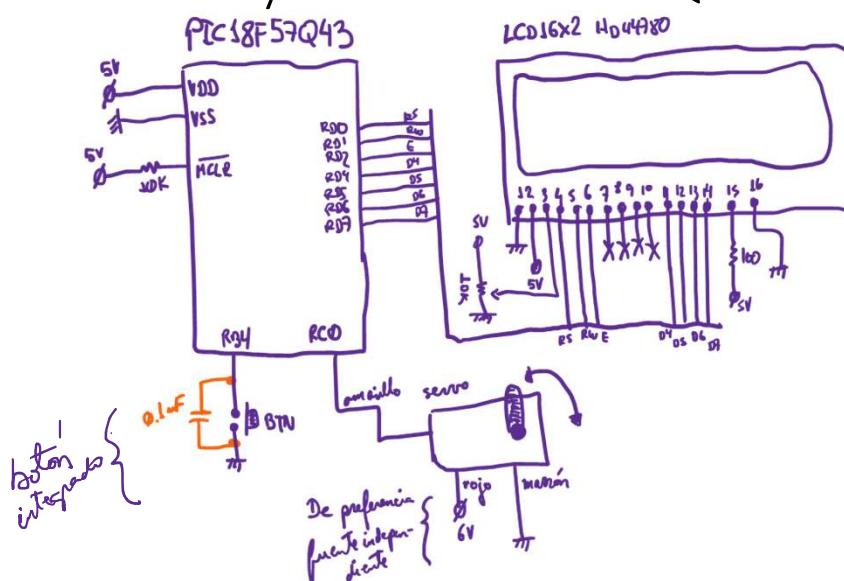


# Opciones para obtener la señal requerida para manipular el servo

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



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



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

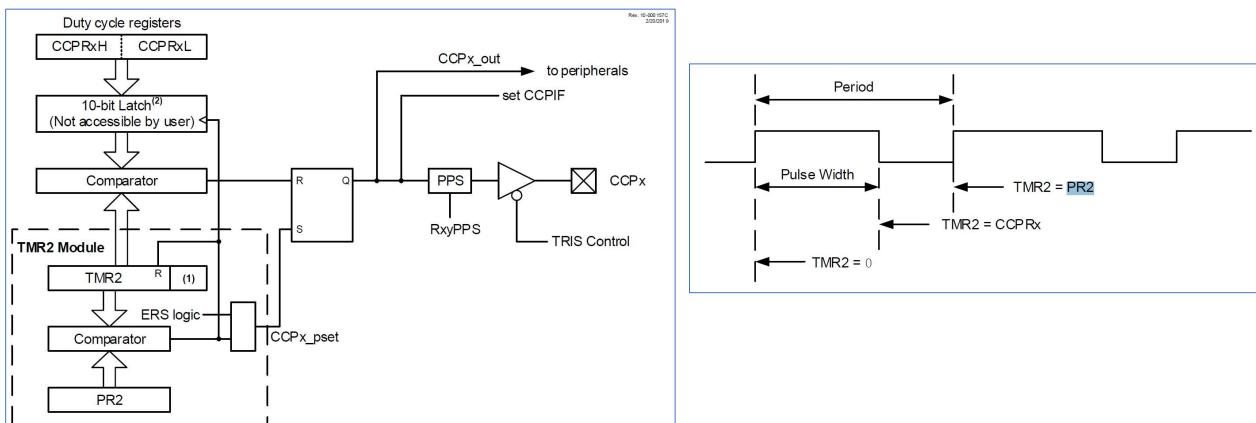
- El CCP en modo PWM nos permitiría obtener dicha señal cuadrada
- Tenemos que validar si el CCP-PWM permite generar una señal a 50Hz

Table 28-1. CCP Mode - Timer Resources

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

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

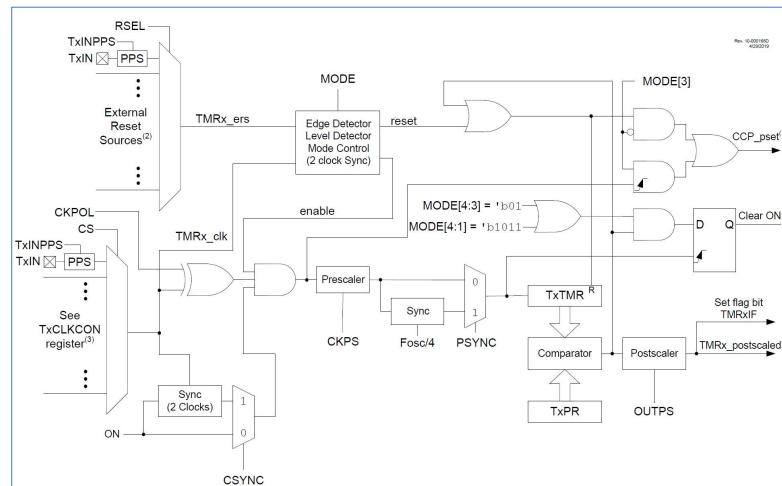
- Diagrama de bloques del modo PWM



Nota: PR2 en el diagrama es el T2PR en el PIC18F57Q43

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

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



## ¿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

| CKPS[2:0] | Timer Clock Prescale Select |
|-----------|-----------------------------|
| 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               |

## ¿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

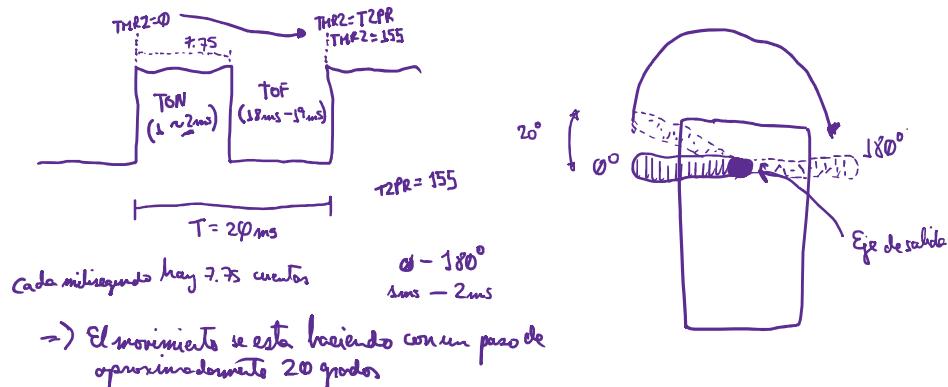
$\downarrow$   
 T2PR para cada opción de prescale  
 teniendo en cuenta que Fosc = 4 MHz?  
 T2PR → (registro de 8 bits)  
 $0 - 255$

|       |            |
|-------|------------|
| 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   |

## Resolución del duty cycle de tu PWM

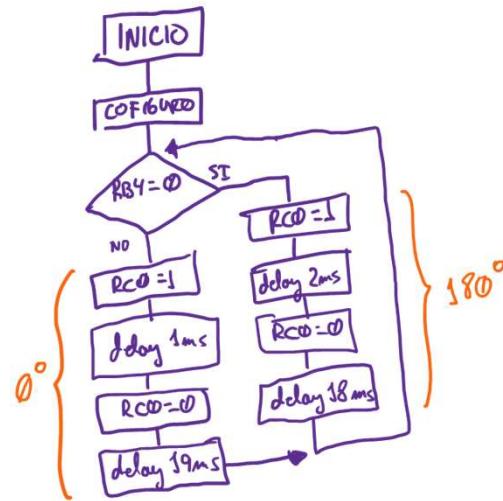
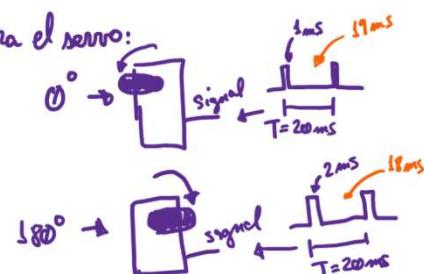


- En consecuencia: No se puede usar el CCP en modo PWM para controlar un servo porque se va a mover escalonadamente. Cada cambio de valor en el duty cycle determinado por CCPRx lo va a realizar en 20 grados angulares del eje del servomecanismo.

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

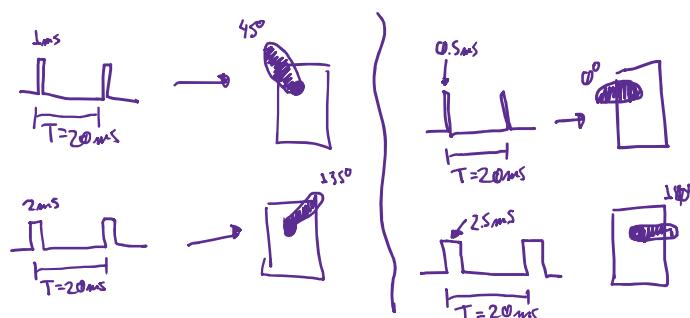
- Usando el botón, intercambiar entre 0° y 180° el servomecanismo

Para el servo:



## El servo no se mueve mas de los 180°!

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



## 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

```

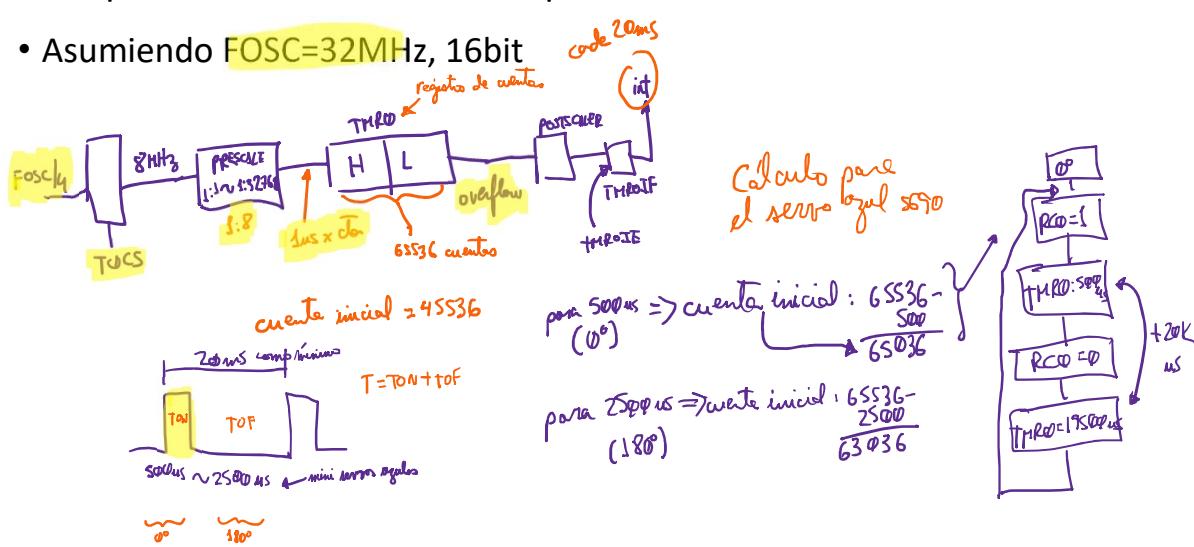
1 #include <xc.h>
2 #include "cabecera.h"
3 #include "LCD.h"
4 #define _XTAL_FREQ 32000000UL
5
6 void configuro(void){
7     //configuración del oscilador
8     OSCCON1 = 0x60;
9     OSCFRC = 0x06;           //HFINTOSC a 32MHz
10    OSCEN = 0x40;
11    //configuración de las E/S
12    TRISBbits.TRISB4 = 1;    //RB4 entrada
13    ANSELBbits.ANSELB4 = 0;  //RB4 digital
14    WPUBbits.WPUB4 = 1;     //RB4 con pullup activado
15    TRISChits.TRISCO = 0;   //RC0 salida
16    ANSELChits.ANSELCO = 1; //RC0 digital
17 }
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 }
```

```

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 }
```

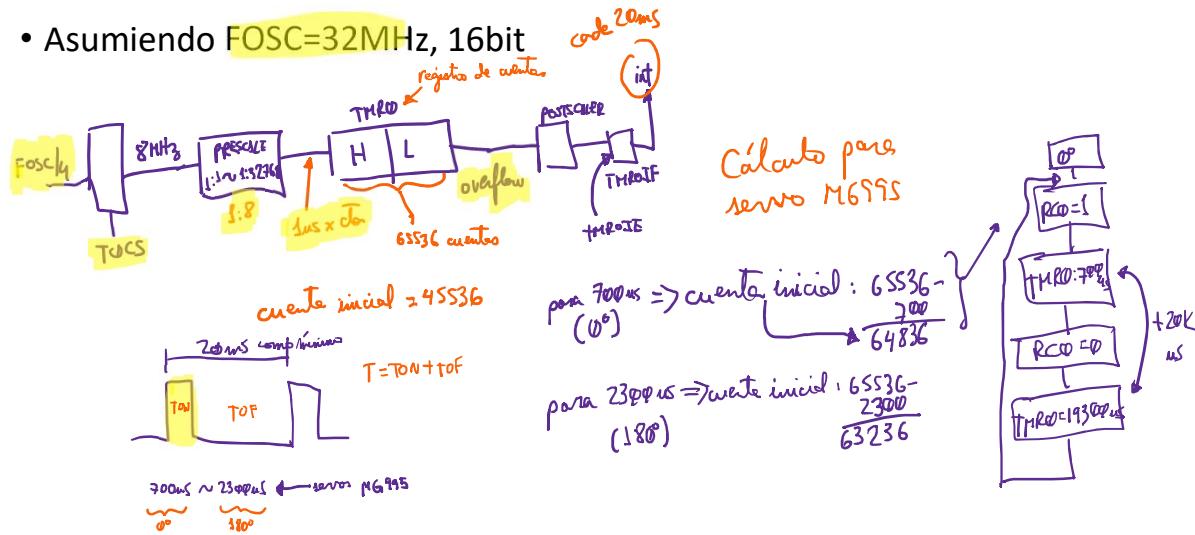
## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

- Asumiendo FOSC=32MHz, 16bit



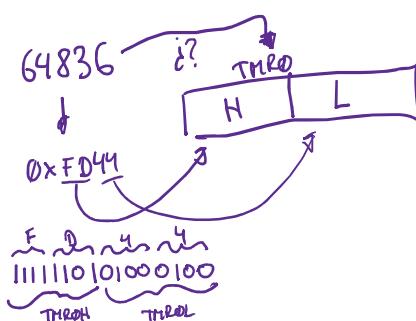
## Empleando el Timer0 para temporizar 20ms del periodo necesario para el servo

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



¿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



$$\text{TMROH} = (64836 \gg 8) \& 0x00FF$$

Binary representation:

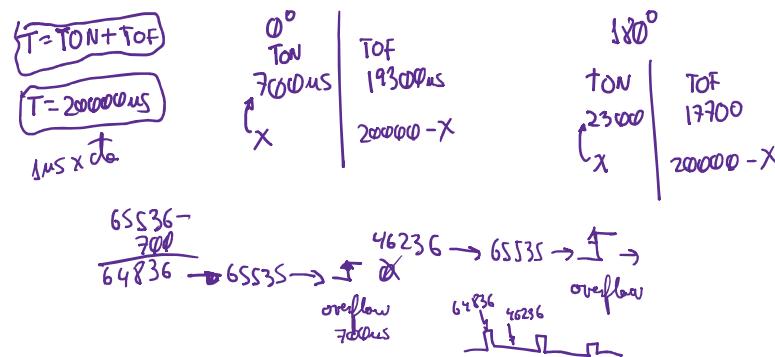
|                  |     |                  |
|------------------|-----|------------------|
| 0000000011111101 | AND | 0000000000000000 |
| 0000000001111111 | AND | 0000000000000000 |
| 0000000011111101 | AND | 0000000000000000 |

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

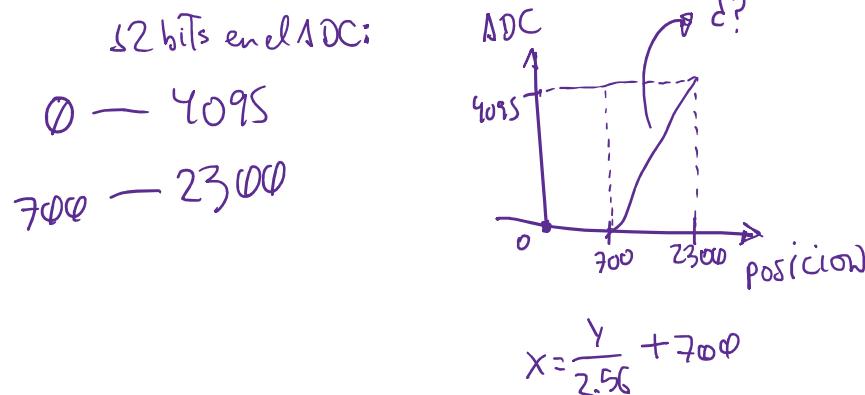
Binary representation:

|                  |     |                  |
|------------------|-----|------------------|
| 1111110101000100 | AND | 1111110101000100 |
| 0000000001111111 | AND | 0000000001111111 |
| 0000000001000100 | AND | 0000000001000100 |

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



Relación ADC con temporizado de servo



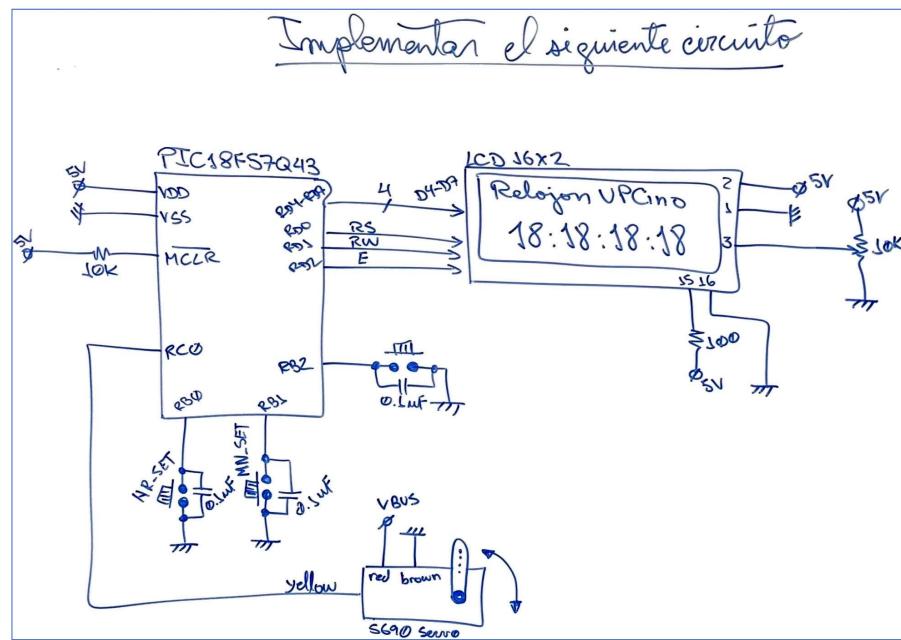
## Cuestionario:

- 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.

## Ejercicio 2024-1

Implementar el siguiente circuito

- Hardware



## 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);
}

```

## 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

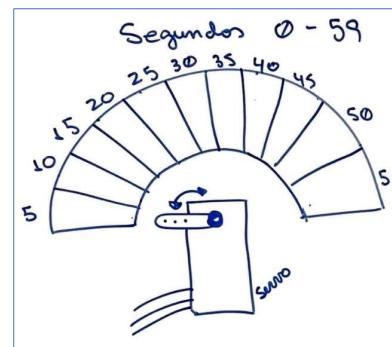


Diagrama que muestra el ciclo de trabajo para el servo:

0.5ms - 2.5ms → (2ms de ancho)

0° - 180° → 180/11 = 16 cada sección

162us → 182us

```

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(175.00+(182*1));
        break;
    case 10 ... 14 :
        ...
}

```

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

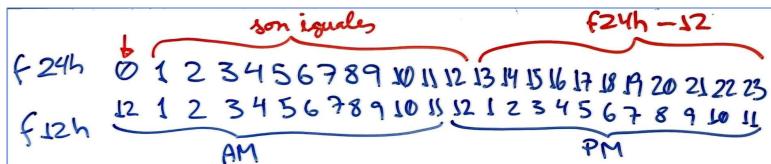
```

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;
}

```

## 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);
    }
}

```

## 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.

**Clock 24H UC2024**  
14:01:53:76

**Clock 12H UC2024**  
02:01:53 PM

- 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).

## Asignación (minilab) Semana 11 2024-2 EL51

- Basado en el ejemplo desarrollado en el laboratorio, añadir lo siguiente:
  - Agregar un pulsador a RB2 en configuración activo en bajo para que funcione la INT2.
  - La INT2 permitirá el cambio de formato de visualización de entre 24H y 12H tal como se muestra a continuación:

**Reloj en 12H**  
03:15'45"11 PM

**Reloj en 24H**  
15:15'45"11

- Revisar las condiciones de entrega en la actividad del AV.

## Minilab Sem11 NRC8436 Lunes 13:00

- Al circuito implementado agregarle un elemento de visualización ó un elemento de emisión de sonido (buzzer) y un pulsador mas para la activación/desactivación de función de alarma.
- Implementar una funcionalidad de alarma el cual pueda ajustarse la hora de alarma.
- El estado de la alarma (activada/desactivada) debe de visualizarse en el LCD
- Carga de video en la actividad del AV Unidad 3 Semana 11:
  - Presentándose
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 17:00

## Minilab Sem11 NRC8436 Lunes 17:00

- Al circuito implementado agregarle un elemento de visualización ó un elemento de emisión de sonido (buzzer), y un pulsador mas para la activación/desactivación de función de alarma.
- Implementar una funcionalidad de alarma el cual pueda ajustarse la hora de alarma.
- El estado de la alarma (activada/desactivada) debe de visualizarse en el LCD
- Carga de video en la actividad del AV Unidad 3 Semana 11:
  - Presentándose
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 20:55

## Minilab Sem11 NRC8444 Jueves 14:00

- Al circuito implementado agregarle un elemento de visualización ó un elemento de emisión de sonido (buzzer), y un pulsador mas para la activación/desactivación de función de alarma.
- Implementar una funcionalidad de alarma el cual pueda ajustarse la hora de alarma.
- El estado de la alarma (activada/desactivada) debe de visualizarse en el LCD
- Carga de video en la actividad del AV Unidad 3 Semana 11:
  - Presentándose
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 18:00

## Minilab Sem11 NRC8444 Jueves 18:00

- Al circuito implementado agregarle un elemento de visualización ó un elemento de emisión de sonido (buzzer), y un pulsador mas para la activación/desactivación de función de alarma.
- Implementar una funcionalidad de alarma el cual pueda ajustarse la hora de alarma.
- El estado de la alarma (activada/desactivada) debe de visualizarse en el LCD
- Carga de video en la actividad del AV Unidad 3 Semana 11:
  - Presentándose
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 21:30

# Minilab Sem11 NRC8447 Viernes 09:00

- Al circuito implementado agregarle un elemento de visualización ó un elemento de emisión de sonido (buzzer), y un pulsador mas para la activación/desactivación de función de alarma.
- Implementar una funcionalidad de alarma el cual pueda ajustarse la hora de alarma.
- El estado de la alarma (activada/desactivada) debe de visualizarse en el LCD
- Carga de video en la actividad del AV Unidad 3 Semana 11:
  - Presentándote
  - Mostrando el código en XC8 C en la pantalla del MPLABX de tu PC
  - Evidencia de funcionamiento del circuito hecho en el protoboard
- Recordar que este minilab representa 2p del LB3
- Límite 12:30

## Código reloj con alarma semana 11 2025-2

```

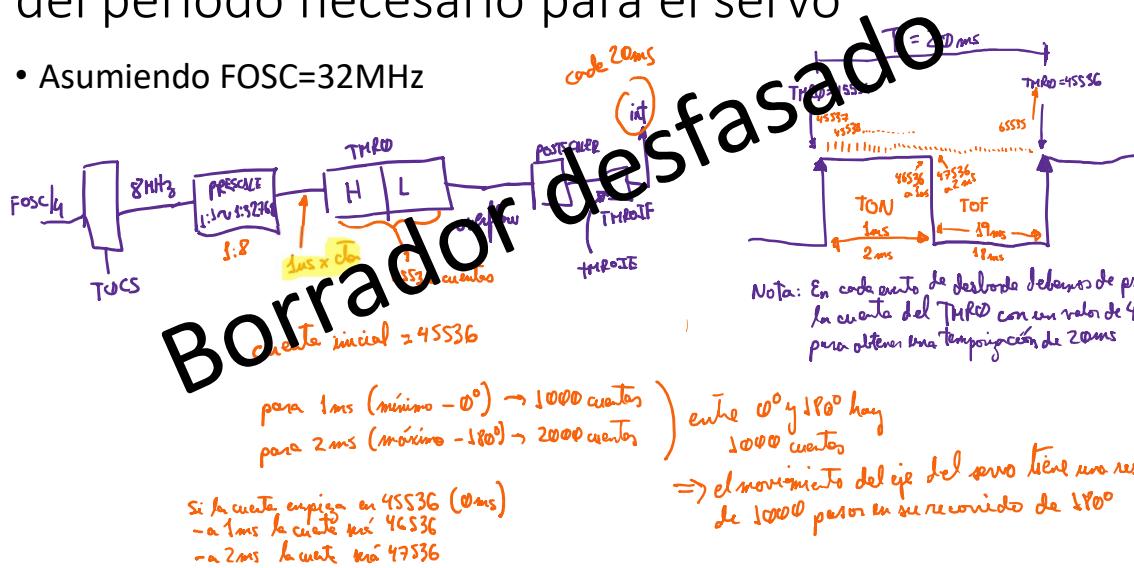
1 //Reloj con alarma
2 //Hecho por Kalun Lau
3 //UPC 97 de Noviembre del 2025
4
5 #include xc8.h
6 #include "cabecera.h"
7 #include "LCD.h"
8 #include "uart.h" //REQ 32000000UL
9 #define RELOJ 0
10 #define SET_CLK 1
11 #define ALARM 2
12 #define APAGADA 3
13 #define ENCENDIDA 4
14
15 unsigned char horas=10,minutos=10,segundos=10,centesimas=10; //hora inicial 10:10:10
16 unsigned char al_hora=17,al_mins=00; //alarma ajustada a 17:00 inicialmente
17 unsigned char char_alarma,decena,unidad;
18 unsigned char PRECIO;
19 unsigned char ALARM = APAGADA;
20
21 void config(void){
22     //Configuración del oscilador
23     OSCCON1 = _hex8; //INTOSC, ports 1:1
24     OSCCON2 = _hex8; //OSC2P/OSC2N a 1MHz
25     OSCEN = _hex8; //INTOSC enabled
26     //Configuración de E/S
27     TRISB = _hex8; //PB2 R3 R0 como entradas
28     ANSEL = _hex8; //PB2 R3 R0 como digitales
29     WPUB = _hex8; //PB2 R3 R0 pullup enabled
30     TRISBbits.TRISB_0 = 0; //R3 estara el buzzer
31     ANSELbits.ANSEL_0 = 0; //R3 estara el buzzer
32     //Configuración del Timer
33     T1CLK = _hex8; //1k source fosc/4
34
35     void __interrupt(IRQ_CCP1) CCP1_ISR(void){
36         PIR3bits.CCP1IF = 0;
37         if(centesimas == 99){
38             centesimas = 0;
39             if(segundos == 59){
40                 segundos = 0;
41                 if(minutos == 59){
42                     minutos = 0;
43                     if(horas == 23){
44                         horas = 0;
45                     }
46                     else{
47                         horas++;
48                     }
49                 }
50                 else{
51                     minutos++;
52                 }
53             }
54             else{
55                 segundos++;
56             }
57         }
58         else{
59             if(al_mins == 59){
60                 al_mins = 0;
61                 if(al_hora == 23){
62                     al_hora = 0;
63                 }
64                 else{
65                     al_hora++;
66                 }
67             }
68         }
69     }
70
71     void DISP_ALARMA(void){
72         POS_CURSOR(1,14);
73         ESCRIBE_MENSAJE("ALARMA");
74         POS_CURSOR(2,14);
75         ESCRIBE_MENSAJE("APAGADA");
76     }
77
78     void _interrupt(irq(IRQ_IN1)) IN1_ISR(void){
79         PIR6bits.INT1IF = 0;
80         switch(WK00){
81             case RELOJ:
82                 ALARM = ENCENDIDA;
83                 break;
84             case SET_CLK:
85                 if(horas == 23){
86                     horas = 0;
87                 }
88                 else{
89                     horas++;
90                 }
91             case SET_AL:
92                 if(al_hora == 23){
93                     al_hora = 0;
94                 }
95                 else{
96                     al_hora++;
97                 }
98             }
99         }
100
101     void main(void){
102         config();
103         lcd_init();
104         while(1){
105             switch(WK00){
106                 case RELOJ:
107                     POS_CURSOR(1,2);
108                     ESCRIBE_MENSAJE("Reloj en UPC",11);
109                     POS_CURSOR(2,2);
110                     convierte(horas);
111                     ENVIA_CHAR(unidad+hex30);
112                     ENVIA_CHAR(unidad+hex30);
113                     ENVIA_CHAR(' ');
114                     convierte(minutos);
115                     ENVIA_CHAR(unidad+hex30);
116                     ENVIA_CHAR(unidad+hex30);
117                     convierte(segundos);
118                     ENVIA_CHAR(unidad+hex30);
119                     ENVIA_CHAR(unidad+hex30);
120                     ENVIA_CHAR(' ');
121                     ENVIA_CHAR(unidad+hex30);
122                     ENVIA_CHAR(unidad+hex30);
123                     ENVIA_CHAR(' ');
124                     ENVIA_CHAR(unidad+hex30);
125                     ENVIA_CHAR(unidad+hex30);
126                     ENVIA_CHAR(unidad+hex30);
127                     ENVIA_CHAR(unidad+hex30);
128                     ENVIA_CHAR(unidad+hex30);
129                     ENVIA_CHAR(unidad+hex30);
130                     ENVIA_CHAR(unidad+hex30);
131                     ENVIA_CHAR(unidad+hex30);
132                     ENVIA_CHAR(unidad+hex30);
133                     ENVIA_CHAR(' ');
134                     convierte(centesimas);
135                     ENVIA_CHAR(unidad+hex30);
136                     ENVIA_CHAR(unidad+hex30);
137                     ENVIA_CHAR(unidad+hex30);
138                     ENVIA_CHAR(unidad+hex30);
139                     ENVIA_CHAR(unidad+hex30);
140                     ENVIA_CHAR(unidad+hex30);
141                     ENVIA_CHAR(unidad+hex30);
142                     ENVIA_CHAR(unidad+hex30);
143                     ENVIA_CHAR(unidad+hex30);
144                     ENVIA_CHAR(unidad+hex30);
145                     ENVIA_CHAR(unidad+hex30);
146                     ENVIA_CHAR(unidad+hex30);
147                     ENVIA_CHAR(unidad+hex30);
148                     ENVIA_CHAR(unidad+hex30);
149                     ENVIA_CHAR(unidad+hex30);
150                     ENVIA_CHAR(unidad+hex30);
151                     ENVIA_CHAR(unidad+hex30);
152                     ENVIA_CHAR(unidad+hex30);
153                     ENVIA_CHAR(unidad+hex30);
154                     ENVIA_CHAR(unidad+hex30);
155                     ENVIA_CHAR(unidad+hex30);
156                     ENVIA_CHAR(unidad+hex30);
157                     ENVIA_CHAR(unidad+hex30);
158                     ENVIA_CHAR(unidad+hex30);
159                     ENVIA_CHAR(unidad+hex30);
160                     ENVIA_CHAR(unidad+hex30);
161                     ENVIA_CHAR(unidad+hex30);
162                     ENVIA_CHAR(unidad+hex30);
163                     ENVIA_CHAR(unidad+hex30);
164                     ENVIA_CHAR(unidad+hex30);
165                     ENVIA_CHAR(unidad+hex30);
166                     ENVIA_CHAR(unidad+hex30);
167                     ENVIA_CHAR(unidad+hex30);
168                     ENVIA_CHAR(unidad+hex30);
169                     ENVIA_CHAR(unidad+hex30);
170                     ENVIA_CHAR(unidad+hex30);
171                     ENVIA_CHAR(unidad+hex30);
172                     ENVIA_CHAR(unidad+hex30);
173                     ENVIA_CHAR(unidad+hex30);
174                 }
175             }
176         }
177     }
178
179     void __interrupt(irq(default)) DEFAULT_ISR(void){
180         if(al_min == 0)
181             break;
182     }
183
184     void _interrupt(irq(IRQ_IN0)) INT0_ISR(void){
185         if(al_min == 0)
186             break;
187     }
188
189     void _interrupt(irq(IRQ_IN2)) INT2_ISR(void){
190         if(al_min == 0)
191             break;
192     }
193
194     void _interrupt(irq(IRQ_IN3)) INT3_ISR(void){
195         if(al_min == 0)
196             break;
197     }
198
199     void _interrupt(irq(IRQ_IN4)) INT4_ISR(void){
200         if(al_min == 0)
201             break;
202     }
203
204     void _interrupt(irq(IRQ_IN5)) INT5_ISR(void){
205         if(al_min == 0)
206             break;
207     }
208
209     void _interrupt(irq(IRQ_IN6)) INT6_ISR(void){
210         if(al_min == 0)
211             break;
212     }
213
214     void _interrupt(irq(IRQ_IN7)) INT7_ISR(void){
215         if(al_min == 0)
216             break;
217     }
218
219     void _interrupt(irq(IRQ_IN8)) INT8_ISR(void){
220         if(al_min == 0)
221             break;
222     }
223
224     void _interrupt(irq(IRQ_IN9)) INT9_ISR(void){
225         if(al_min == 0)
226             break;
227     }
228
229     void _interrupt(irq(IRQ_IN10)) INT10_ISR(void){
230         if(al_min == 0)
231             break;
232     }
233
234     void _interrupt(irq(IRQ_IN11)) INT11_ISR(void){
235         if(al_min == 0)
236             break;
237     }
238
239     void _interrupt(irq(IRQ_IN12)) INT12_ISR(void){
240         if(al_min == 0)
241             break;
242     }
243
244     void _interrupt(irq(IRQ_IN13)) INT13_ISR(void){
245         if(al_min == 0)
246             break;
247     }
248
249     void _interrupt(irq(IRQ_IN14)) INT14_ISR(void){
250         if(al_min == 0)
251             break;
252     }
253
254     void _interrupt(irq(IRQ_IN15)) INT15_ISR(void){
255         if(al_min == 0)
256             break;
257     }
258
259     void _interrupt(irq(IRQ_IN16)) INT16_ISR(void){
260         if(al_min == 0)
261             break;
262     }
263
264     void _interrupt(irq(IRQ_IN17)) INT17_ISR(void){
265         if(al_min == 0)
266             break;
267     }
268
269     void _interrupt(irq(IRQ_IN18)) INT18_ISR(void){
270         if(al_min == 0)
271             break;
272     }
273
274     void _interrupt(irq(IRQ_IN19)) INT19_ISR(void){
275         if(al_min == 0)
276             break;
277     }
278
279     void _interrupt(irq(IRQ_IN20)) INT20_ISR(void){
280         if(al_min == 0)
281             break;
282     }
283
284     void _interrupt(irq(IRQ_IN21)) INT21_ISR(void){
285         if(al_min == 0)
286             break;
287     }
288
289     void _interrupt(irq(IRQ_IN22)) INT22_ISR(void){
290         if(al_min == 0)
291             break;
292     }
293
294     void _interrupt(irq(IRQ_IN23)) INT23_ISR(void){
295         if(al_min == 0)
296             break;
297     }
298
299     void _interrupt(irq(IRQ_IN24)) INT24_ISR(void){
300         if(al_min == 0)
301             break;
302     }
303
304     void _interrupt(irq(IRQ_IN25)) INT25_ISR(void){
305         if(al_min == 0)
306             break;
307     }
308
309     void _interrupt(irq(IRQ_IN26)) INT26_ISR(void){
310         if(al_min == 0)
311             break;
312     }
313
314     void _interrupt(irq(IRQ_IN27)) INT27_ISR(void){
315         if(al_min == 0)
316             break;
317     }
318
319     void _interrupt(irq(IRQ_IN28)) INT28_ISR(void){
320         if(al_min == 0)
321             break;
322     }
323
324     void _interrupt(irq(IRQ_IN29)) INT29_ISR(void){
325         if(al_min == 0)
326             break;
327     }
328
329     void _interrupt(irq(IRQ_IN30)) INT30_ISR(void){
330         if(al_min == 0)
331             break;
332     }
333
334     void _interrupt(irq(IRQ_IN31)) INT31_ISR(void){
335         if(al_min == 0)
336             break;
337     }
338
339     void _interrupt(irq(IRQ_IN32)) INT32_ISR(void){
340         if(al_min == 0)
341             break;
342     }
343
344     void _interrupt(irq(IRQ_IN33)) INT33_ISR(void){
345         if(al_min == 0)
346             break;
347     }
348
349     void _interrupt(irq(IRQ_IN34)) INT34_ISR(void){
350         if(al_min == 0)
351             break;
352     }
353
354     void _interrupt(irq(IRQ_IN35)) INT35_ISR(void){
355         if(al_min == 0)
356             break;
357     }
358
359     void _interrupt(irq(IRQ_IN36)) INT36_ISR(void){
360         if(al_min == 0)
361             break;
362     }
363
364     void _interrupt(irq(IRQ_IN37)) INT37_ISR(void){
365         if(al_min == 0)
366             break;
367     }
368
369     void _interrupt(irq(IRQ_IN38)) INT38_ISR(void){
370         if(al_min == 0)
371             break;
372     }
373
374     void _interrupt(irq(IRQ_IN39)) INT39_ISR(void){
375         if(al_min == 0)
376             break;
377     }
378
379     void _interrupt(irq(IRQ_IN40)) INT40_ISR(void){
380         if(al_min == 0)
381             break;
382     }
383
384     void _interrupt(irq(IRQ_IN41)) INT41_ISR(void){
385         if(al_min == 0)
386             break;
387     }
388
389     void _interrupt(irq(IRQ_IN42)) INT42_ISR(void){
390         if(al_min == 0)
391             break;
392     }
393
394     void _interrupt(irq(IRQ_IN43)) INT43_ISR(void){
395         if(al_min == 0)
396             break;
397     }
398
399     void _interrupt(irq(IRQ_IN44)) INT44_ISR(void){
400         if(al_min == 0)
401             break;
402     }
403
404     void _interrupt(irq(IRQ_IN45)) INT45_ISR(void){
405         if(al_min == 0)
406             break;
407     }
408
409     void _interrupt(irq(IRQ_IN46)) INT46_ISR(void){
410         if(al_min == 0)
411             break;
412     }
413
414     void _interrupt(irq(IRQ_IN47)) INT47_ISR(void){
415         if(al_min == 0)
416             break;
417     }
418
419     void _interrupt(irq(IRQ_IN48)) INT48_ISR(void){
420         if(al_min == 0)
421             break;
422     }
423
424     void _interrupt(irq(IRQ_IN49)) INT49_ISR(void){
425         if(al_min == 0)
426             break;
427     }
428
429     void _interrupt(irq(IRQ_IN50)) INT50_ISR(void){
430         if(al_min == 0)
431             break;
432     }
433
434     void _interrupt(irq(IRQ_IN51)) INT51_ISR(void){
435         if(al_min == 0)
436             break;
437     }
438
439     void _interrupt(irq(IRQ_IN52)) INT52_ISR(void){
440         if(al_min == 0)
441             break;
442     }
443
444     void _interrupt(irq(IRQ_IN53)) INT53_ISR(void){
445         if(al_min == 0)
446             break;
447     }
448
449     void _interrupt(irq(IRQ_IN54)) INT54_ISR(void){
450         if(al_min == 0)
451             break;
452     }
453
454     void _interrupt(irq(IRQ_IN55)) INT55_ISR(void){
455         if(al_min == 0)
456             break;
457     }
458
459     void _interrupt(irq(IRQ_IN56)) INT56_ISR(void){
460         if(al_min == 0)
461             break;
462     }
463
464     void _interrupt(irq(IRQ_IN57)) INT57_ISR(void){
465         if(al_min == 0)
466             break;
467     }
468
469     void _interrupt(irq(IRQ_IN58)) INT58_ISR(void){
470         if(al_min == 0)
471             break;
472     }
473
474     void _interrupt(irq(IRQ_IN59)) INT59_ISR(void){
475         if(al_min == 0)
476             break;
477     }
478
479     void _interrupt(irq(IRQ_IN60)) INT60_ISR(void){
480         if(al_min == 0)
481             break;
482     }
483
484     void _interrupt(irq(IRQ_IN61)) INT61_ISR(void){
485         if(al_min == 0)
486             break;
487     }
488
489     void _interrupt(irq(IRQ_IN62)) INT62_ISR(void){
490         if(al_min == 0)
491             break;
492     }
493
494     void _interrupt(irq(IRQ_IN63)) INT63_ISR(void){
495         if(al_min == 0)
496             break;
497     }
498
499     void _interrupt(irq(IRQ_IN64)) INT64_ISR(void){
500         if(al_min == 0)
501             break;
502     }
503
504     void _interrupt(irq(IRQ_IN65)) INT65_ISR(void){
505         if(al_min == 0)
506             break;
507     }
508
509     void _interrupt(irq(IRQ_IN66)) INT66_ISR(void){
510         if(al_min == 0)
511             break;
512     }
513
514     void _interrupt(irq(IRQ_IN67)) INT67_ISR(void){
515         if(al_min == 0)
516             break;
517     }
518
519     void _interrupt(irq(IRQ_IN68)) INT68_ISR(void){
520         if(al_min == 0)
521             break;
522     }
523
524     void _interrupt(irq(IRQ_IN69)) INT69_ISR(void){
525         if(al_min == 0)
526             break;
527     }
528
529     void _interrupt(irq(IRQ_IN70)) INT70_ISR(void){
530         if(al_min == 0)
531             break;
532     }
533
534     void _interrupt(irq(IRQ_IN71)) INT71_ISR(void){
535         if(al_min == 0)
536             break;
537     }
538
539     void _interrupt(irq(IRQ_IN72)) INT72_ISR(void){
540         if(al_min == 0)
541             break;
542     }
543
544     void _interrupt(irq(IRQ_IN73)) INT73_ISR(void){
545         if(al_min == 0)
546             break;
547     }
548
549     void _interrupt(irq(IRQ_IN74)) INT74_ISR(void){
550         if(al_min == 0)
551             break;
552     }
553
554     void _interrupt(irq(IRQ_IN75)) INT75_ISR(void){
555         if(al_min == 0)
556             break;
557     }
558
559     void _interrupt(irq(IRQ_IN76)) INT76_ISR(void){
560         if(al_min == 0)
561             break;
562     }
563
564     void _interrupt(irq(IRQ_IN77)) INT77_ISR(void){
565         if(al_min == 0)
566             break;
567     }
568
569     void _interrupt(irq(IRQ_IN78)) INT78_ISR(void){
570         if(al_min == 0)
571             break;
572     }
573
574     void _interrupt(irq(IRQ_IN79)) INT79_ISR(void){
575         if(al_min == 0)
576             break;
577     }
578
579     void _interrupt(irq(IRQ_IN80)) INT80_ISR(void){
580         if(al_min == 0)
581             break;
582     }
583
584     void _interrupt(irq(IRQ_IN81)) INT81_ISR(void){
585         if(al_min == 0)
586             break;
587     }
588
589     void _interrupt(irq(IRQ_IN82)) INT82_ISR(void){
590         if(al_min == 0)
591             break;
592     }
593
594     void _interrupt(irq(IRQ_IN83)) INT83_ISR(void){
595         if(al_min == 0)
596             break;
597     }
598
599     void _interrupt(irq(IRQ_IN84)) INT84_ISR(void){
600         if(al_min == 0)
601             break;
602     }
603
604     void _interrupt(irq(IRQ_IN85)) INT85_ISR(void){
605         if(al_min == 0)
606             break;
607     }
608
609     void _interrupt(irq(IRQ_IN86)) INT86_ISR(void){
610         if(al_min == 0)
611             break;
612     }
613
614     void _interrupt(irq(IRQ_IN87)) INT87_ISR(void){
615         if(al_min == 0)
616             break;
617     }
618
619     void _interrupt(irq(IRQ_IN88)) INT88_ISR(void){
620         if(al_min == 0)
621             break;
622     }
623
624     void _interrupt(irq(IRQ_IN89)) INT89_ISR(void){
625         if(al_min == 0)
626             break;
627     }
628
629     void _interrupt(irq(IRQ_IN90)) INT90_ISR(void){
630         if(al_min == 0)
631             break;
632     }
633
634     void _interrupt(irq(IRQ_IN91)) INT91_ISR(void){
635         if(al_min == 0)
636             break;
637     }
638
639     void _interrupt(irq(IRQ_IN92)) INT92_ISR(void){
640         if(al_min == 0)
641             break;
642     }
643
644     void _interrupt(irq(IRQ_IN93)) INT93_ISR(void){
645         if(al_min == 0)
646             break;
647     }
648
649     void _interrupt(irq(IRQ_IN94)) INT94_ISR(void){
650         if(al_min == 0)
651             break;
652     }
653
654     void _interrupt(irq(IRQ_IN95)) INT95_ISR(void){
655         if(al_min == 0)
656             break;
657     }
658
659     void _interrupt(irq(IRQ_IN96)) INT96_ISR(void){
660         if(al_min == 0)
661             break;
662     }
663
664     void _interrupt(irq(IRQ_IN97)) INT97_ISR(void){
665         if(al_min == 0)
666             break;
667     }
668
669     void _interrupt(irq(IRQ_IN98)) INT98_ISR(void){
670         if(al_min == 0)
671             break;
672     }
673
674     void _interrupt(irq(IRQ_IN99)) INT99_ISR(void){
675         if(al_min == 0)
676             break;
677     }
678
679     void _interrupt(irq(IRQ_IN100)) INT100_ISR(void){
680         if(al_min == 0)
681             break;
682     }
683
684     void _interrupt(irq(IRQ_IN101)) INT101_ISR(void){
685         if(al_min == 0)
686             break;
687     }
688
689     void _interrupt(irq(IRQ_IN102)) INT102_ISR(void){
690         if(al_min == 0)
691             break;
692     }
693
694     void _interrupt(irq(IRQ_IN103)) INT103_ISR(void){
695         if(al_min == 0)
696             break;
697     }
698
699     void _interrupt(irq(IRQ_IN104)) INT104_ISR(void){
700         if(al_min == 0)
701             break;
702     }
703
704     void _interrupt(irq(IRQ_IN105)) INT105_ISR(void){
705         if(al_min == 0)
706             break;
707     }
708
709     void _interrupt(irq(IRQ_IN106)) INT106_ISR(void){
710         if(al_min == 0)
711             break;
712     }
713
714     void _interrupt(irq(IRQ_IN107)) INT107_ISR(void){
715         if(al_min == 0)
716             break;
717     }
718
719     void _interrupt(irq(IRQ_IN108)) INT108_ISR(void){
720         if(al_min == 0)
721             break;
722     }
723
724     void _interrupt(irq(IRQ_IN109)) INT109_ISR(void){
725         if(al_min == 0)
726             break;
727     }
728
729     void _interrupt(irq(IRQ_IN110)) INT110_ISR(void){
730         if(al_min == 0)
731             break;
732     }
733
734     void _interrupt(irq(IRQ_IN111)) INT111_ISR(void){
735         if(al_min == 0)
736             break;
737     }
738
739     void _interrupt(irq(IRQ_IN112)) INT112_ISR(void){
740         if(al_min == 0)
741             break;
742     }
743
744     void _interrupt(irq(IRQ_IN113)) INT113_ISR(void){
745         if(al_min == 0)
746             break;
747     }
748
749     void _interrupt(irq(IRQ_IN114)) INT114_ISR(void){
750         if(al_min == 0)
751             break;
752     }
753
754     void _interrupt(irq(IRQ_IN115)) INT115_ISR(void){
755         if(al_min == 0)
756             break;
757     }
758
759     void _interrupt(irq(IRQ_IN116)) INT116_ISR(void){
760         if(al_min == 0)
761             break;
762     }
763
764     void _interrupt(irq(IRQ_IN117)) INT117_ISR(void){
765         if(al_min == 0)
766             break;
767     }
768
769     void _interrupt(irq(IRQ_IN118)) INT118_ISR(void){
770         if(al_min == 0)
771             break;
772     }
773
774     void _interrupt(irq(IRQ_IN119)) INT119_ISR(void){
775         if(al_min == 0)
776             break;
777     }
778
779     void _interrupt(irq(IRQ_IN120)) INT120_ISR(void){
780         if(al_min == 0)
781             break;
782     }
783
784     void _interrupt(irq(IRQ_IN121)) INT121_ISR(void){
785         if(al_min == 0)
786             break;
787     }
788
789     void _interrupt(irq(IRQ_IN122)) INT122_ISR(void){
790         if(al_min == 0)
791             break;
792     }
793
794     void _interrupt(irq(IRQ_IN123)) INT123_ISR(void){
795         if(al_min == 0)
796             break;
797     }
798
799     void _interrupt(irq(IRQ_IN124)) INT124_ISR(void){
800         if(al_min == 0)
801             break;
802     }
803
804     void _interrupt(irq(IRQ_IN125)) INT125_ISR(void){
805         if(al_min == 0)
806             break;
807     }
808
809     void _interrupt(irq(IRQ_IN126)) INT126_ISR(void){
810         if(al_min == 0)
811             break;
812     }
813
814     void _interrupt(irq(IRQ_IN127)) INT127_ISR(void){
815         if(al_min == 0)
816             break;
817     }
818
819     void _interrupt(irq(IRQ_IN128)) INT128_ISR(void){
820         if(al_min == 0)
821             break;
822     }
823
824     void _interrupt(irq(IRQ_IN129)) INT129_ISR(void){
825         if(al_min == 0)
826             break;
827     }
828
829     void _interrupt(irq(IRQ_IN130)) INT130_ISR(void){
830         if(al_min == 0)
831             break;
832     }
833
834     void _interrupt(irq(IRQ_IN131)) INT131_ISR(void){
835         if(al_min == 0)
836             break;
837     }
838
839     void _interrupt(irq(IRQ_IN132)) INT132_ISR(void){
840         if(al_min == 0)
841             break;
842     }
843
844     void _interrupt(irq(IRQ_IN133)) INT133_ISR(void){
845         if(al_min == 0)
846             break;
847     }
848
849     void _interrupt(irq(IRQ_IN134)) INT134_ISR(void){
850         if(al_min == 0)
851             break;
852     }
853
854     void _interrupt(irq(IRQ_IN135)) INT135_ISR(void){
855         if(al_min == 0)
856             break;
857     }
858
859     void _interrupt(irq(IRQ_IN136)) INT136_ISR(void){
860         if(al_min == 0)
861             break;
862     }
863
864     void _interrupt(irq(IRQ_IN137)) INT137_ISR(void){
865         if(al_min == 0)
866             break;
867     }
868
869     void _interrupt(irq(IRQ_IN138)) INT138_ISR(void){
870         if(al_min == 0)
871             break;
872     }
873
874     void _interrupt(irq(IRQ_IN139)) INT139_ISR(void){
875         if(al_min == 0)
876             break;
877     }
878
879     void _interrupt(irq(IRQ_IN140)) INT140_ISR(void){
880         if(al_min == 0)
881             break;
882     }
883
884     void _interrupt(irq(IRQ_IN141)) INT141_ISR(void){
885         if(al_min == 0)
886             break;
887     }
888
889     void _interrupt(irq(IRQ_IN142)) INT142_ISR(void){
890         if(al_min == 0)
891             break;
892     }
893
894     void _interrupt(irq(IRQ_IN143)) INT143_ISR(void){
895         if(al_min == 0)
896             break;
897     }
898
899     void _interrupt(irq(IRQ_IN144)) INT144_ISR(void){
900         if(al_min == 0)
901             break;
902     }
903
904     void _interrupt(irq(IRQ_IN145)) INT145_ISR(void){
905         if(al_min == 0)
906             break;
907     }
908
909     void _interrupt(irq(IRQ_IN146)) INT146_ISR(void){
910         if(al_min == 0)
911             break;
912     }
913
914     void _interrupt(irq(IRQ_IN147)) INT147_ISR(void){
915         if(al_min == 0)
916             break;
917     }
918
919     void _interrupt(irq(IRQ_IN148)) INT148_ISR(void){
920         if(al_min == 0)
921             break;
922     }
923
924     void _interrupt(irq(IRQ_IN149)) INT149_ISR(void){
925         if(al_min == 0)
926             break;
927     }
928
929     void _interrupt(irq(IRQ_IN150)) INT150_ISR(void){
930         if(al_min == 0)
931             break;
932     }
933
934     void _interrupt(irq(IRQ_IN151)) INT151_ISR(void){
935         if(al_min == 0)
936             break;
937     }
938
939     void _interrupt(irq(IRQ_IN152)) INT152_ISR(void){
940         if(al_min == 0)
941             break;
942     }
943
944     void _interrupt(irq(IRQ_IN153)) INT153_ISR(void){
945         if(al_min == 0)
946             break;
947     }
948
949     void _interrupt(irq(IRQ_IN154)) INT154_ISR(void){
950         if(al_min == 0)
951             break;
952     }
953
954     void _interrupt(irq(IRQ_IN155)) INT155_ISR(void){
955         if(al_min == 0)
956             break;
957     }
958
959     void _interrupt(irq(IRQ_IN156)) INT156_ISR(void){
960         if(al_min == 0)
961             break;
962     }
963
964     void _interrupt(irq(IRQ_IN157)) INT157_ISR(void){
965         if(al_min == 0)
966             break;
967     }
968
969     void _interrupt(irq(IRQ_IN158)) INT158_ISR(void){
970         if(al_min == 0)
971             break;
972     }
973
974     void _interrupt(irq(IRQ_IN159)) INT159_ISR(void){
975         if(al_min == 0)
976             break;
977     }
978
979     void _interrupt(irq(IRQ_IN160)) INT160_ISR(void){
980         if(al_min == 0)
981             break;
982     }
983
984     void _interrupt(irq(IRQ_IN161)) INT161_ISR(void){
985         if(al_min == 0)
986             break;
987     }
988
989     void _interrupt(irq(IRQ_IN162)) INT162_ISR(void){
990         if(al_min == 0)
991             break;
992     }
993
994     void _interrupt(irq(IRQ_IN163)) INT163_ISR(void){
995         if(al_min == 0)
996             break;
997     }
998
999     void _interrupt(irq(IRQ_IN164)) INT164_ISR(void){
1000         if(al_min == 0)
1001             break;
1002     }
1003
1004     void _interrupt(irq(IRQ_IN165)) INT165_ISR(void){
1005         if(al_min == 0)
1006             break;
1007     }
1008
1009     void _interrupt(irq(IRQ_IN166)) INT166_ISR(void){
1010         if(al_min == 0)
1011             break;
1012     }
1013
1014     void _interrupt(irq(IRQ_IN167)) INT167_ISR(void){
1015         if(al_min == 0)
1016             break;
1017     }
1018
1019     void _interrupt(irq(IRQ_IN168)) INT168_ISR(void){
1020         if(al_min == 0)
1021             break;
1022     }
1023
1024     void _interrupt(irq(IRQ_IN169)) INT169_ISR(void){
1025         if(al_min == 0)
1026             break;
1027     }
1028
1029     void _interrupt(irq(IRQ_IN170)) INT170_ISR(void){
1030         if(al_min == 0)
1031             break;
1032     }
1033
1034     void _interrupt(irq(IRQ_IN171)) INT171_ISR(void){
1035         if(al_min == 0)
1036             break;
1037     }
1038
1039     void _interrupt(irq(IRQ_IN172)) INT172_ISR(void){
1040         if(al_min == 0)
1041             break;
1042     }
1043
1044     void _interrupt(irq(IRQ_IN173)) INT173_ISR(void){
1045         if(al_min == 0)
1046             break;
1047     }
1048
1049     void _interrupt(irq(IRQ_IN174)) INT174_ISR(void){
1050         if(al_min == 0)
1051             break;
1052     }
1053
1054     void _interrupt(irq(IRQ_IN175)) INT175_ISR(void){
1055         if(al_min == 0)
1056             break;
1057     }
1058
1059     void _interrupt(irq(IRQ_IN176)) INT176_ISR(void){
1060         if(al_min == 0)
1061             break;
1062     }
1063
1064     void _interrupt(irq(IRQ_IN177)) INT177_ISR(void){
1065         if(al_min == 0)
1066             break;
1067     }
1068
1069     void _interrupt(irq(IRQ_IN178)) INT178_ISR(void){
1070         if(al_min == 0)
1071             break;
1072     }
1073
1074     void _interrupt(irq(IRQ_IN179)) INT179_ISR(void){
1075         if(al_min == 0)
1076             break;
1077     }
1078
1079     void _interrupt(irq(IRQ_IN180)) INT180_ISR(void){
1080         if(al_min == 0)
1081             break;
1082     }
1083
1084     void _interrupt(irq(IRQ_IN181)) INT181_ISR(void){
1085         if(al_min == 0)
1086             break;
1087     }
1088
1089     void _interrupt(irq(IRQ_IN182)) INT182_ISR(void){
1090         if(al_min == 0)
1091             break;
1092     }
1093
1094     void _interrupt(irq(IRQ_IN183)) INT183_ISR(void){
1095         if(al_min == 0)
1096             break;
1097     }
1098
1099     void _interrupt(irq(IRQ_IN184)) INT184_ISR(void){
1100         if(al_min == 0)
1101             break;
1102     }
1103
1104     void _interrupt(irq(IRQ_IN185)) INT185_ISR(void){
1105         if(al_min == 0)
1106             break;
1107     }
1108
1109     void _interrupt(irq(IRQ_IN186)) INT18
```

## Fin de la sesión

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

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

- Asumiendo FOSC=32MHz



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

- Escalamiento ángulo de servo vs cuentas del Timer

