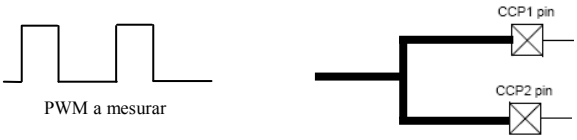
Nom i Cognoms:_____**Una possible solució**_____

1) Fem servir la unitat de *capture* del Pic18F4550 per **mesurar** la freqüència i *duty cicle* d'un senyal PWM generat externament. Com a base de temps, farem servir l'oscil.lador del PIC que és de 4MHz.



Si volem garantir una resolució de l'1% al mesurar el *duty cicle*, quina freqüència màxima pot tenir el senyal PWM que llegim?

La mesura es farà en la unitat de capture, entre un flanc ascendent i un flanc descendent. El clock de més freqüència que pot arribar a la unitat és d'1MHz (fosc/4, sense prescaler), per tant un tick cada 1us.
Per tenir una resolució de l'1% (1%) cal poder comptar com a minim 100 tics, per tant el periode ha de ser com a mínim de 100us, corresponent a una freqüència de 10KHz.

Nota: si vau llegir l'1'1% (u coma u per cent) també es compta com a correcte. La l (lletra) i el 1 (nombre) es veuen igual en Times New Roman.

	7	6	5	4	3	2	1	0
value after reset	RD16	--	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
	0	0	0	0	0	0	0	

RD16: 16-bit read/write mode enable bit
0 = Enables read/write of Timer1 in two 8-bit operations
1 = Enable read/write of Timer1 in 16-bit operation
T1CKPS1:T1CKPS0: Timer1 input clock prescale select bits
00 = 1:1 prescale value
01 = 1:2 prescale value
10 = 1:4 prescale value
11 = 1:8 prescale value
T1OSCEN: Timer1 oscillator enable bit
0 = Timer1 oscillator is shut off
1 = Timer1 oscillator is enabled
T1SYNC: Timer1 external clock input synchronization select bit
When TMR1CS = 1
0 = Synchronize external clock input
1 = Do not synchronize external clock input
When TMR1CS = 0
This bit is ignored.
TMR1CS: Timer1 clock source select bit
0 = Instruction cycle clock (FOSC/4)
1 = External clock from pin RC0/T1OSO/T13CKI
TMR1ON: Timer1 on bit
0 = Stop Timer1
1 = Enables Timer1

Figure 8.4. T1CON contents (redraw with permission of Micro

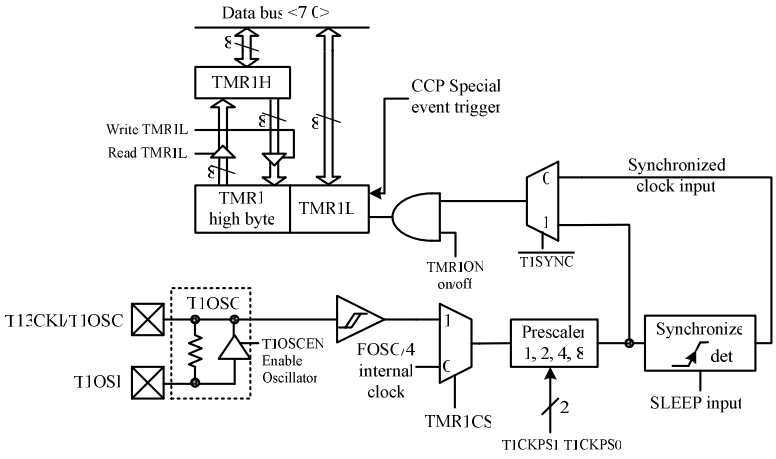
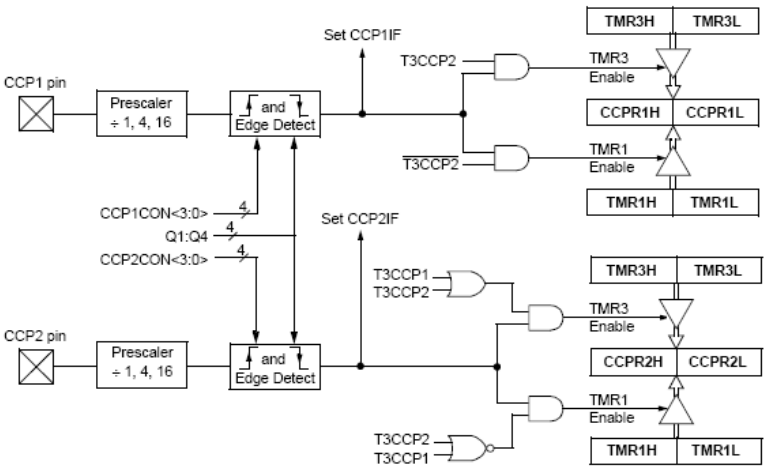


Figure 8.3 Timer1 block diagram 16-bit mode (redraw with permission of Microchip)



15.2 Capture Mode

In Capture mode, the CCPRxH:CCPRxL register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding CCPx pin. An event is defined as one of the following:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

The event is selected by the mode select bits, CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set; it must be cleared in software. If another capture occurs before the value in register CCPRx is read, the old captured value is overwritten by the new captured value.

2) Volem fer servir la unitat de *compare* del Pic18F4550, per generar un senyal periòdic pel pin CCP1, de freqüència 800Hz. L'oscilador del PIC és de 10MHz.
Calcula els valors a posar als registres T1CON, CCP1CON, CCPR1H i CCPR1L.

800 Hz vol dir un període de 1,25ms. Això vol dir que en 1,25ms hem de tenir el senyal a 1 i a 0. Farem servir el mode toggle de la unitat compare amb la meitat de temps: 0'625 ms = 625000 ns.
Amb us oscil.lador de 10MHz tenim fosc/4 = 2,5 MHz, que vol dir un període de 400ns al timer.
En 625000ns hi ha 1562,5 ticks de 400ns, per tant 1562 o 1563 és el valor a carregar als registres CCPRH:CCPRL.
Al CCP1CON posarem: xxxx0010 mode compare i toggle.
Al T1CON hi posarem: 10000001 que vol dir 16 bits, font a fosc/4, no prescaler i start.

3) Calcula el valor de ADRESH i ADRESL després de fer una conversió, si tenim una tensió AN0 de 2,15V, a AN1 de 1V a AN2 de 2V i a AN3 de 3V. El canal triat és el 0 i els bits 4 i 5 del registre ADCON1 estàn a 1.

Tindrem referències de 2V i 3V i Vin de 2'15V. Per tant:

Dout = (2'15V – 2V) * (2^10 / (3V – 2V)) = 153,6

Si suposem justificació a la dreta:

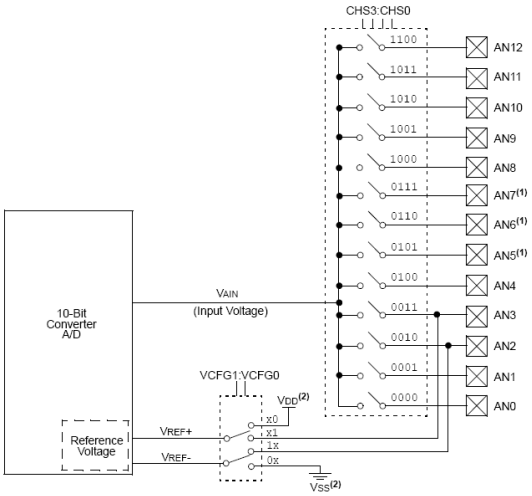
ADRESH = 0x00,
ADRESL = 0x99 (truncant) o 0x9A (arrodonint)

REGISTER 21-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 5 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD



Nom i Cognoms:_____Una possible solució_____

4) Per a declarar una rutina de servei a la interrupció de baixa prioritat usant el compilador C18, ho fem d'aquesta manera:

```
#pragma interruptlow rsi
void rsi (void)
{
        ...
}
```

Explica quines seran les diferències en el codi generat pel compilador posant-hi aquest pragma o no posant-lo.

Aquest pragma provoca que la rutina acabi amb *retfie* en comptes de *ret*.

5) Volem detectar si dues temperatures llegides de dos sensors ubicats en un reactor són diferents, i disparar una alarma en cas de que això passi. Un enginyer de telecomunicacions presenta el següent fragment de codi, escrit en C stàndar per a un micro genèric:

```
static int iTemperatures[2];

void interrupt vReadTemperatures (void)
{
   iTemperatures[0] = !! read in value from hardware
   iTemperatures[1] = !! read in value from hardware
}

void main (void)
{
   int iTemp0, iTemp1;

   while (TRUE)
   {
      iTemp0 = iTemperatures[0];
      iTemp1 = iTemperatures[1];
      if (iTemp0 != iTemp1)
         !! Set off howling alarm;
   }
}
```

La rsi salta periòdicament (per timer). Raoneu si el codi és correcte, i en cas contrari detalleu la seqüència que s'hauria de produir per a que saltés una falsa alarma.
NOTA: Només es mostra un fragment de programa on heu de trobar un error. La resta de codi que no es mostra (inicialitzacions, ubicació del vector d'interrupcions, crida a la rsi, ...) es pot assumir que és totalment correcta.

Una possible seqüència de falsa alarma:
Suposem que les temperatures són iguals. P.ex: T = 10ºC
S'executa la primera assignació *itemp0 = iTempreratures[0]*; (10ºC)
Salta la interrupció just després de l'assignació.
A la rsi llegim les noves temperatures.
Suposem que les temperatures són iguals, però ara ja T = 11ºC. Retornem de la rsi.
S'executa la segona assignació *itemp1 = iTempreratures[1];* (11ºC)
El resultat de la comparació *if (itemp0!=itemp1)* es compleix.
Salta l'alarma tot i que les temperatures són iguals

6) Volem realitzar la pràctica del freqüencímetre, i s'ens plantegen dues estratègies per a mesurar la freqüència d'un senyal extern connectat a un pin d'entrada:

> Opció a. Mesurar la freqüència: Comptar amb el timer1 el nombre de tics que ens arriben al pin d'entrada, i comprovar a cada segon el contingut del timer1.
> Opció b. Mesurar el període: Comptar amb el timer1 el temps que transcorre entre 2 tics consecutius del senyal d'entrada. El timer1 s'incrementa cada milisegon.

Suposant que la freqüència del senyal d'entrada ronda els 10Hz., Quin error cometem amb cada estratègia? Quina opció us sembla millor?
NOTA: Quantifiqueu l'error de comptatge com a una unitat de la xifra menys significativa.

Opció a:
El timer1 compta 10 cicles en 1 seg.
Tenim la incertesa d'una unitat. Llavors: 1/10 = 10% d'error.

Opció b:
El timer 1 compta 100 tics en un període del senyal d'entrada (1/10Hz / 1 mseg)
Tenim la incertesa d'una unitat. Llavors 1/100 = 1% d'error

Opció a. Error = _____%

Opció b. Error = _____%

	7	6	5	4	3	2	1	0
value after reset	--	--	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
	0	0	0	0	0	0	0	0

DCxB1:DCxB0: PWM duty cycle bit 1 and bit 0 for CCP module x
capture mode:
unused
compare mode:
unused
PWM mode:
These two bits are the lsbs (bit 1 and bit 0) of the 10-bit PWM duty cycle.
CCPxM3:CCPxM0: CCP module x mode select bits
0000 = capture/compare/PWM disabled (resets CCPx module)
0001 = reserved
0010 = compare mode, toggle output on match (CCPxIF bit is set)
0100 = capture mode, every falling edge
0101 = capture mode, every rising edge
0110 = capture mode, every 4th rising edge
0111 = capture mode, every 16th rising edge
1000 = compare mode, initialize CCP pin low, on compare match force CCP pin high (CCPxIF bit is set)
1001 = compare mode, initialize CCP pin high, on compare match force CCP pin low (CCPxIF bit is set)
1010 = compare mode, generate software interrupt on compare match (CCP pin unaffected, CCPxIF bit is set).
1011 = compare mode, trigger special event (CCPxIF bit is set)
For CCP1 and CCP2: Timer1 or Timer3 is reset on event
For all other modules: CCPx pin is unaffected and is configured as an I/O port.
11xx = PWM mode

Figure 8.10 CCPxCON register (x = 1,..,5) (redraw with permission of Microchip)