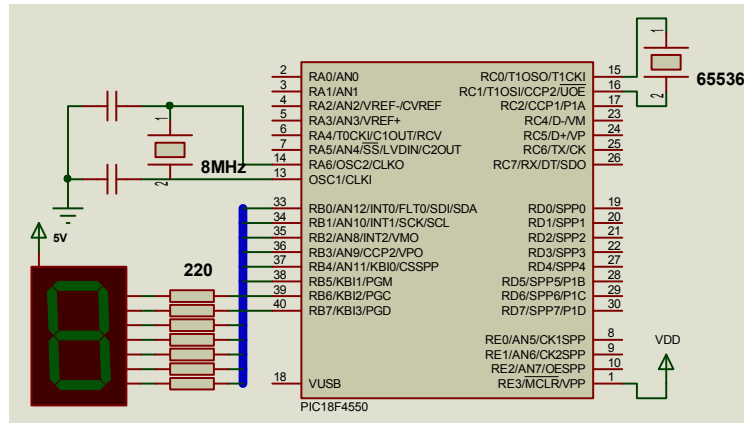


Nom i Cognoms: \_\_\_\_\_ Una possible Solució \_\_\_\_\_

- 1) El circuit de la figura, presenta la connexió d'un display de 7 segments al PORTB del PIC18F4550



- a) (1p.) Quin valor hem de carregar als registres TRISB i PORTB per a encendre tots leds del display? Per què?

El display és ànode comú, així que cal posar '0' per encendre els leds. Cal configurar el PORTB de sortida.

TRISB = 0  
PORTB = 0

- b) (1p.) Calculeu el corrent  $I_{RB0}$  quan el led corresponent està encès. Considereu  $V_\gamma = 1.1V$ ,  $V_{dd} = 5V$ ,  $V_{ol} = 0.6V$ ,  $V_{oh} = 4.3V$ . JUSTIFICA el resultat

$$I_{RB0} = (V_{dd} - V_\gamma - V_{ol}) / 220 = 15mA$$

$$I_{RB0} = \underline{15mA}$$

- c) (2p.) Volem mostrar pel display el comptatge de segons (cíclic del 0 al 9). Per a tal fi, usarem el timer1 connectat a un oscil.lador de quartz extern de  $2^{16}$  Hz.

```
main ( )
{
    ....//inicialitzacions    vàries    i
    correctes
    for ( ; ; )
    {
        PORTB = PatroLeds[segs];
        Sleep ( ) ;
    }
}

Void interrupt RSI_timer1 (void)
{
    segs ++;
    segs = segs % 10 ;
    TMR1IF = 0;
}
```

A partir del codi presentat, indiqueu el valor (en binari) del registre T1CON per a que el comptatge de segons sigui correcte. JUSTIFICA la resposta.

RD16 = 1 (16 bits)

T1CKPS1:2 = 00 (sense prescaler) T1OSCEN = 1 (oscil.lador ON)

T1SYNC = 1 (No sincronitzem. No podem sincronitzar si el micro està dormint)

TMR1CS = 1 (Clk extern) TMR1ON = 1

T1CON = 1000 1111

- d) (1p.) Durant quin percentatge de temps aproximat el micro estarà dormint? (Podeu considerar que el compilador optimitza el codi).

S'han d'executar 5 instruccions (2 del main i 3 de la RSI) + els increments i comparacions del bucle + el retfie + la latència (4 cicles) + un hipotètic salt a la rsi.

A bulto, un mínim de 20 cicles amb un compilador molt ben optimitzat.

Tcyc = 500ns → Temps de procés = 20\*500ns = 10us.

La resta del temps (999990us), el micro dorm. Un 99'999% del temps.

Nom i Cognoms: \_\_\_\_\_ Una possible Solució \_\_\_\_\_

	7	6	5	4	3	2	1	0
value after reset	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON
	0	0	0	0	0	0	0	0

RD16: 16-bit read/write mode enable bit  
 0 = Enables read/write of Timer3 in two 8-bit operations  
 1 = Enables read/write of Timer3 in 16-bit operation

T3CCP2:T3CCP1: Timer3 and Timer1 to CCPx enable bits  
 00 = Timer1 and Timer2 are the clock sources for CCP1 through CCP5  
 01 = Timer3 and Timer4 are the clock sources for CCP2 through CCP5;  
 Timer1 and Timer2 are the clock sources for CCP1  
 10 = Timer3 and Timer4 are the clock sources for CCP3 through CCP5;  
 Timer1 and Timer2 are the clock sources for CCP1 and CCP2  
 11 = Timer3 and Timer4 are the clock sources for CCP1 through CCP5

T3CKPS1:T3CKPS0: Timer3 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

T3SYNC: Timer3 external clock input synchronization select bit  
 When TMR3CS = 1  
 0 = Synchronizes external clock input  
 1 = Do not synchronize external clock input  
 When TMR3CS = 0  
 This bit is ignored.

TMR3CS: Timer3 clock source select bit  
 0 = Instruction cycle clock (FOSC/4)  
 1 = External clock from pin RC0/T1OSO/T13CKI

TMR3ON: Timer3 on bit  
 0 = Stops Timer3  
 1 = Enables Timer3

Figure 8.8. T3CON contents (redraw with permission of Microchip)

	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 lsb's (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 (CCPx pin unaffected, CCPxIF bit is set).

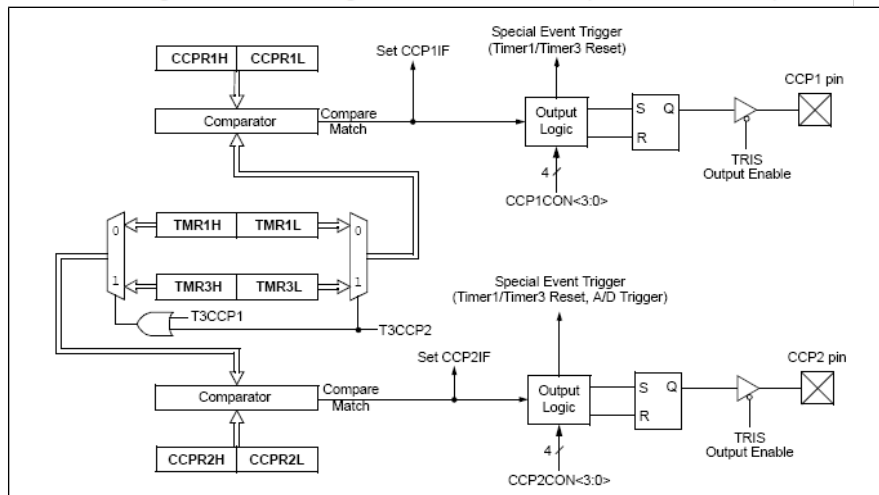
1011 = compare mode, trigger special event (CCPxIF bit is set)

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)



Nom i Cognoms: \_\_\_\_\_ **Una possible Solució** \_\_\_\_\_

- 2) Volem utilitzar la unitat CCP2 per fer una funció Delay\_us(int n) on rebem un paràmetre n que és el nombre de microsegons que volem esperar. Internament, farem servir el Timer3 ja que el Timer1 el tenim ocupat a l'exercici 1. L'oscil·lador connectat al nostre processador (Fosc) és de 8 MHz. Observeu el codi de la funció:

```
void Delay_us(int n)
{
    TIMER3 = 0 ;
    CCPR2 = n;
    while(!CCP2IF);
    CCP2IF=0;
}
```

- a) Per que la funció Delay\_us(int n) sigui efectiva i actuï correctament, amb quins valors haurem hagut d'inicialitzar, els registres CCP2CON i T3CON? Justifica la resposta!! (1p)

```
CCP2CON = 0x0A; //o xxxx1010b ; Cal que sigui en mode compare. No
//volem i no cal tocar el pin per res. Només que posi a 1 el
//Flag.
T3CON    = 0xD9; //11011x01b; Volem 16 bits, Enable, triem l'instruction
//clock, que com que està a 500ns (125ns*4), implica
//PRE=2 (01 en els bits) per tenir comptatge de
//microsegons. Triem el T3 com a Timer pel Compare
```

- b) Quin és el nombre màxim que podem acceptar com a paràmetre? Per què? (0,5p)

65535, que és  $2^{16} - 1$ . Perquè el CCPR2 és de 16 bits.

- c) Té sentit cridar la funció amb un n petit, com per exemple: Delay\_us(3) Justifica la resposta. (1p)

No. Amb un temps d'instrucció de 500ns, entre cridar la funció (és un funció no hi ha cap interrupció implicada), les assignacions, el while, borrar el flag i sortir, haurà passat ja molt més temps (amb bon compilador uns 6 us).

Nom i Cognoms: \_\_\_\_\_ Una possible Solució \_\_\_\_\_

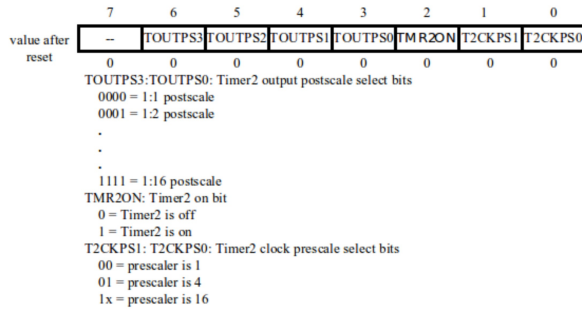


Figure 8.6. T2CON control register (redraw with permission of Microchip)

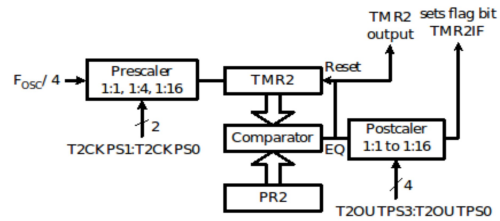


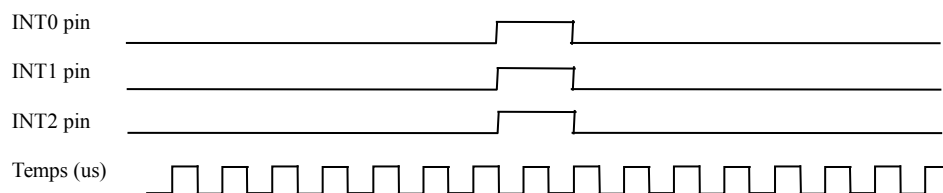
Figure 8.5 Timer2 block diagram (redraw with permission of Microchip)

- 3) Observeu el següent codi. El rellotge del PIC és de **8MHz**. (suposeu que, inicialment, INT0IE=0; INT1IE=1; INT2IE=0; GIEH=1; INT0IP=1; INT1IP=1; INT2IP=1, INT0IF=0; INT1IF=0; INT2IF=0);

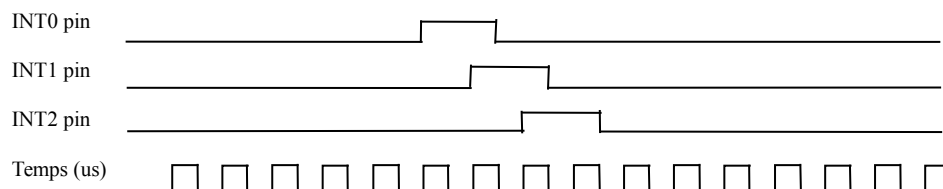
```
void interrupt InterruptHigh()
{
    if ((INT0IF)&&(INT0IE))
    {
        INT0IF=0; INT1IF=0;INT2IF=0;
        INT2IE=1;
    }
    if ((INT1IF)&&(INT1IE))
    {
        INT0IF=0; INT1IF=0;INT2IF=0;
        INT0IE=1;
    }
    if ((INT2IF)&&(INT2IE))
    {
        INT0IF=0; INT1IF=0;INT2IF=0;
        EncénLED();
    }
}
```

Ara observeu les següent seqüències d'events, amb referència temporal, als pins d'interrupció, que estan ben configurats:

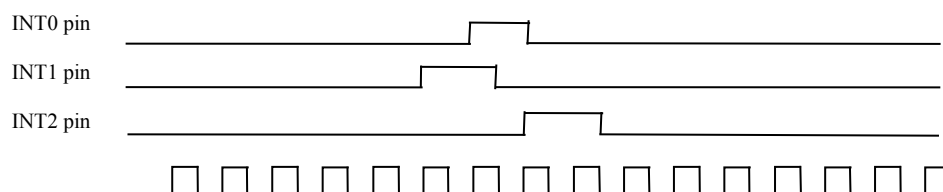
#### Seqüència 1:



#### Seqüència 2:



#### Seqüència 3:



**Nom i Cognoms:** \_\_\_\_\_ **Una possible Solució** \_\_\_\_\_

Alguna o algunes d'aquestes seqüències provocaran l'encesa del LED? Justifica la resposta (1p) ?

Cap de les seqüències encendrà el LED. La clau és la latència d'interrupció (3-4 cicles, és a dir 1,5 o 2 us) més l'execució dels if (1-2us). Independentment de la seqüència, sempre entrarem a un dels IF amb els 3 flags a «1». Els posarem a 0 i ja no entrarem més a la rutina d'interrupció.

- 4) Programeu tot el necessari per generar al PIN CCP1 un senyal PWM de freqüència 1KHz i DC=18%. Supposeu Fosc=8MHz. (1,5p)

Per la freq. del Pwm:  $1\text{ms} = (\text{PRy} + 1) \times 4 \times 125\text{ns} \times \text{PRE}$ , si PRE=1 o 4 PRy no cap en 8 bits, per tant PRE = 16 i **PRy=124**

Pel DC:  $180\text{us} = \text{CCPRxL}:\text{CCPxCON}\langle 5:4 \rangle \times 125\text{ns} \times 16$ . Per tant tenim que,  $\text{CCPRxL}:\text{CCPxCON}\langle 5:4 \rangle = 90$ . Ho posem en binari 10 bits: 0001011010, per tant: **CCPRxL = 22** i els bits  $\text{CCPxCON}\langle 5:4 \rangle = 10$ .

Posem el CCP1CON en mode PWM i amb els bits 5 i 4 a 10: **CCP1CON=0x2C**  
Configurem el Timer2, amb el PRE=16 (només hi ha 1,4 o 16), ON i sense Post ja que no afecta al PWM: **T2CON=0x06**

Finalment, cal que el TRIS estigui de sortida!! **TRISC=0;**

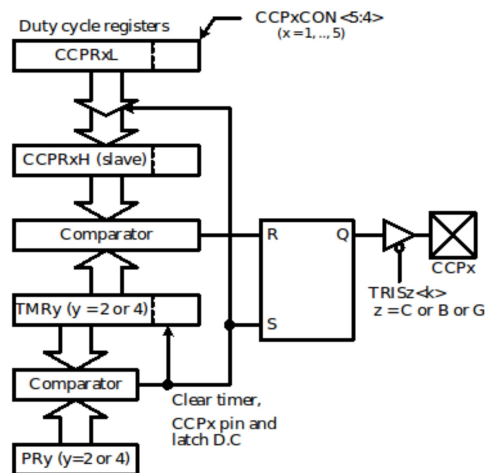


Figure 8.24 Simplified PWM block diagram (redraw with permission of Microchip)

$$\text{PWM period} = [(\text{PRy}) + 1] \times 4 \times \text{TOSC} \times (\text{TMRy prescale factor})$$

$$\text{PWM duty cycle} = (\text{CCPRxL}:\text{CCPxCON}\langle 5:4 \rangle) \times \text{TOSC} \times (\text{TMRy prescale factor})$$

**Nom i Cognoms:** \_\_\_\_\_ **Una possible Solució**

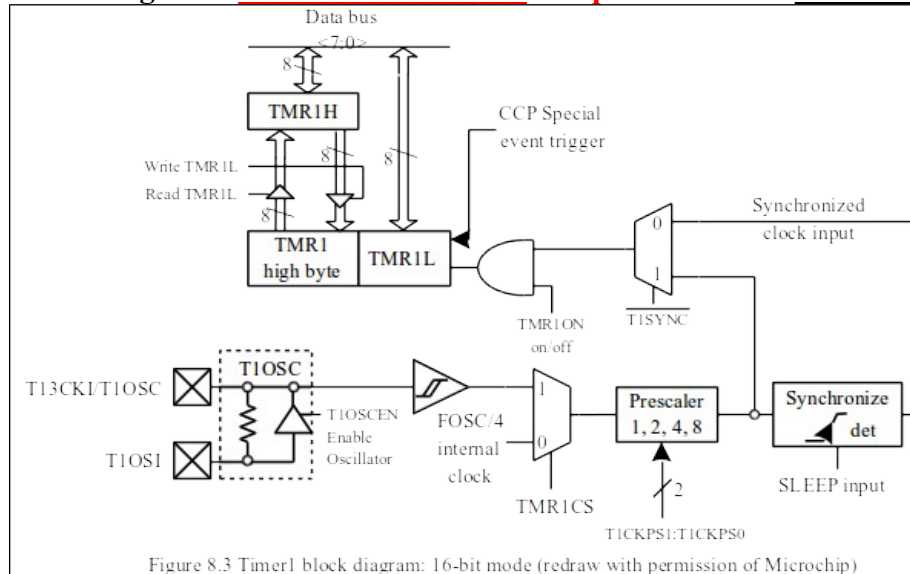


Figure 8.3 Timer1 block diagram: 16-bit mode (redraw with permission of Microchip)

**REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER**

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
bit 7		bit 0					

**Legend:**

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared      x = Bit is unknown

- |         |  |
|---------|--|
| bit 7   | <b>RD16:</b> 16-Bit Read/Write Mode Enable bit<br>1 = Enables register read/write of Timer1 in one 16-bit operation<br>0 = Enables register read/write of Timer1 in two 8-bit operations   |
| bit 6   | <b>T1RUN:</b> Timer1 System Clock Status bit<br>1 = Device clock is derived from Timer1 oscillator<br>0 = Device clock is derived from another source  |
| bit 5-4 | <b>T1CKPS1:T1CKPS0:</b> Timer1 Input Clock Prescale Select bits<br>11 = 1:8 Prescale value<br>10 = 1:4 Prescale value<br>01 = 1:2 Prescale value<br>00 = 1:1 Prescale value  |
| bit 3   | <b>T1OSCEN:</b> Timer1 Oscillator Enable bit<br>1 = Timer1 oscillator is enabled<br>0 = Timer1 oscillator is shut off<br>The oscillator inverter and feedback resistor are turned off to eliminate   |
| bit 2   | <b>T1SYNC:</b> Timer1 External Clock Input Synchronization Select bit<br><u>When TMR1CS = 1:</u><br>1 = Do not synchronize external clock input<br>0 = Synchronize external clock input<br><u>When TMR1CS = 0:</u><br>This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0. |
| bit 1   | <b>TMR1CS:</b> Timer1 Clock Source Select bit<br>1 = External clock from RC0/T1OSO/T13CKI pin (on the rising edge)<br>0 = Internal clock (Fosc/4)  |
| bit 0   | <b>TMR1ON:</b> Timer1 On bit<br>1 = Enables Timer1<br>0 = Stops Timer1   |