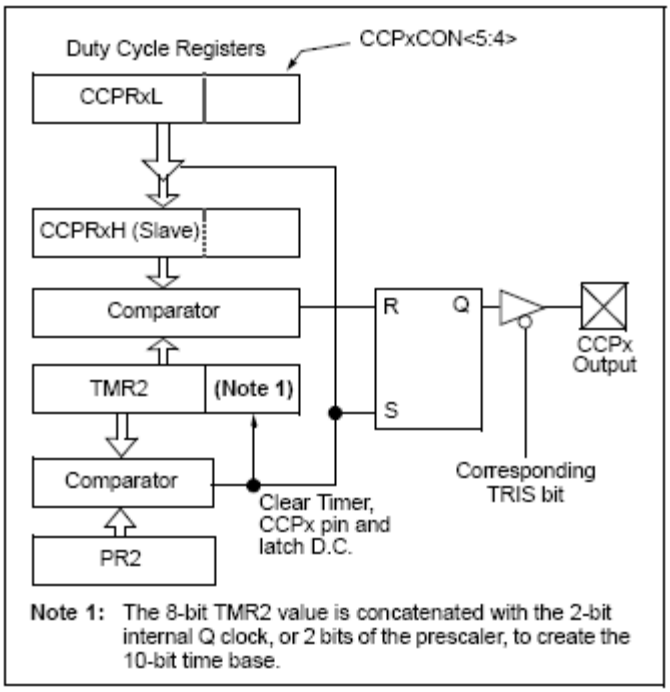
Nom i Cognoms:_____**Una possible solució**_____

1) (1'5p.) Volem generar un senyal de 100KHz. usant la unitat PWM del PIC18F4550. Si el rellotge del micro és de 8MHz, quin valor s'ha de carregar al registre PR2 ? (considereu que treballem sense prescaler)

(Per la freqüència es compten cicles d'instrucció)
PR2 = 8 MHz / 4 / 100KHz -1 = 19

Quants duty cycles diferents es poden programar amb aquestes condicions? JUSTIFICA ELS CÀLCULS

(pel duty cycle es compten cicles de rellotge, no d'instrucció)
8 MHz / 100KHz = 80



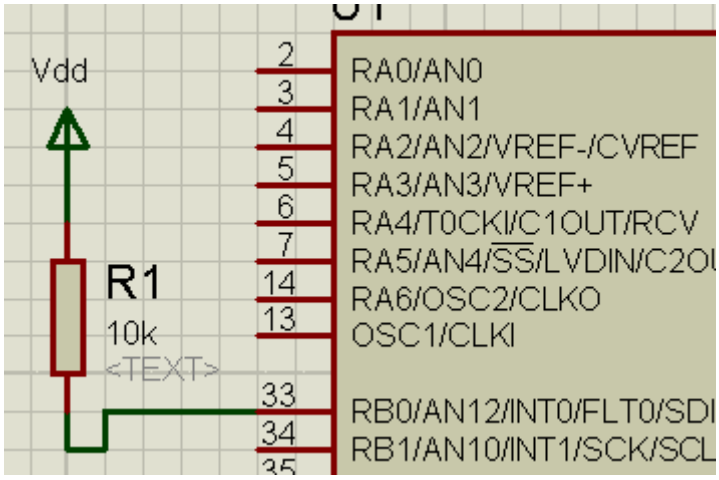
2) (1p.) Raona si és correcta la capçalera d'aquesta rutina de servei a la interrupció:
void interrupt rsi (char c)

No, una RSI no reb paràmetres, doncs no es crida desde el codi.

I aquesta? *int interrupt rsi ( )*

Tampoc. Una RSI no retorna paràmetres.

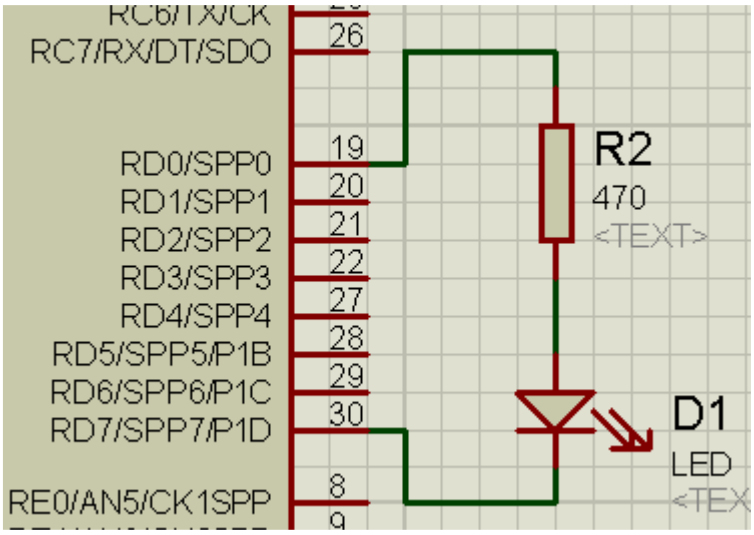
3) (1p.) Si connectem la entrada de interrupció externa INT0 d'aquesta manera:



Raoneu si existeix alguna manera de que es generin peticions d'interrupció INT0

Si, les podem forçar per soft, directament fent INT0IF=1.
També podem forçar-les provocant un flanc amb la seqüència RB0=0; RB0=1

4) (1'5p.)



Completa la taula indicant si el led està encès (ON) o apagat (OFF)

< B_7……B_0>	TRISD = 00000000	TRISD = 00000001	TRISD = 10000000	TRISD = 10000001
PORTD=00000000	OFF	OFF	OFF	OFF
PORTD=00000001	ON	OFF	OFF	OFF
PORTD=10000000	OFF	OFF	OFF	OFF
PORTD=10000001	OFF	OFF	OFF	OFF

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

5) (1'5p.) Analitza el codi adjunt per a la configuració i lectura de la matriu de polsadors (teclat) de la placa EASYPIC6 i corregeix el(s) error(s) existent(s)

```
void ConfigKeyPad(void) {
      ADCON1=0x0F; // Pins PORTA
digitals
      TRISD=0xFF;
}

char ScanKeyPad(void) {
  PORTD= 0x01;
  if(PORTD==0x11) return '1';
  if(PORTD==0x21) return '2';
  if(PORTD==0x41) return '3';
  if(PORTD==0x81) return 'A';
  PORTD= 0x02;
  if(PORTD==0x12) return '4';
  if(PORTD==0x22) return '5';
  if(PORTD==0x42) return '6';
  if(PORTD==0x82) return 'B';
  PORTD= 0x04;
  if(PORTD==0x14) return '7';
  if(PORTD==0x24) return '8';
  if(PORTD==0x44) return '9';
  if(PORTD==0x84) return 'C';
  PORTD= 0x08;
  if(PORTD==0x18) return '*';
  if(PORTD==0x28) return '0';
  if(PORTD==0x48) return '#';
  if(PORTD==0x88) return 'D';
  return 0;
}
```

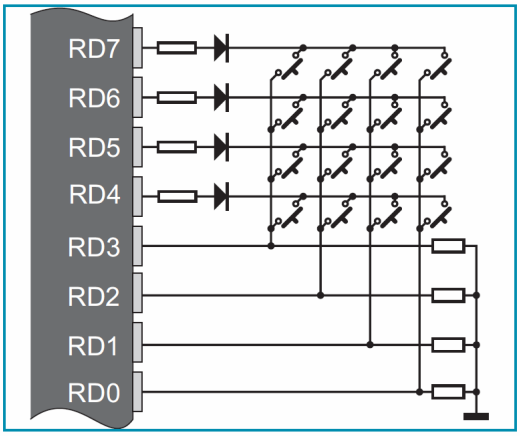


Figure 14-2: Keypad 4x4 performance

Tots els bits de PORTD están configurats com a sortida, i caldria que PORTD<3:0> estiguin configurats com a entrada, i PORTD<7:4> estiguin configurats con sortida. TRISD=0x0F

Al fer el "scan matrix", estem activant per soft els pins PORTD<3:0> i llegint els pins PORTD<7:4>, quan hauria de ser a l'inrevés.

Per tant el codi correcte hauria de ser

```
…
PORTD= 0x10;
if(PORTD==0x11) return '1';
if(PORTD==0x12) return '2';
if(PORTD==0x14) return '3';
if(PORTD==0x18) return 'A';
PORTD= 0x20;
…
```

Com s'indica que es la placa EASYPIC6, també acceptem que comentin el problema que generen les capacitats paràsites, i la necessitat de fer una espera entre la escriptura a les files i la lectura a les columnes de la matriu de tecles.

Es considera que el tractament dels rebots dels polsadors es fa fora de la rutina ScanKeyPad.

6) (1’5p.) Volem fer servir la unitat de compare del PIC18F4550, per generar un senyal periòdic pel pin CCP1, de freqüència 450Hz. L’oscil·lador del PIC és de 4MHz.
Calcula els valors a posar als registres T1CON, CCP1CON, CCPR1H i CCPR1L.

FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM

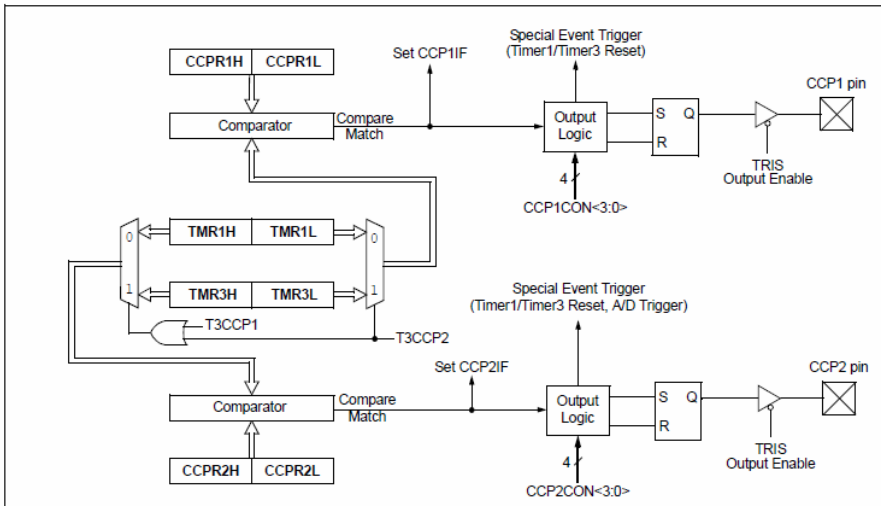


TABLE 15-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	R V on
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	
RCON	IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	
PIR1	SPPIF(2)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	
PIE1	SPPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	
IPR1	SPPIP(2)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	
PIR2	OSCFIF	CMIF	USBIF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	
PIE2	OSCFIE	CMIE	USBIE	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	
IPR2	OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0	
TMR1L	Timer1 Register Low Byte								
TMR1H	Timer1 Register High Byte								
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	
TMR3H	Timer3 Register High Byte								
TMR3L	Timer3 Register Low Byte								
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								
CCPR1H	Capture/Compare/PWM Register 1 High Byte								
CCP1CON	P1M1(2)	P1M0(2)	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	

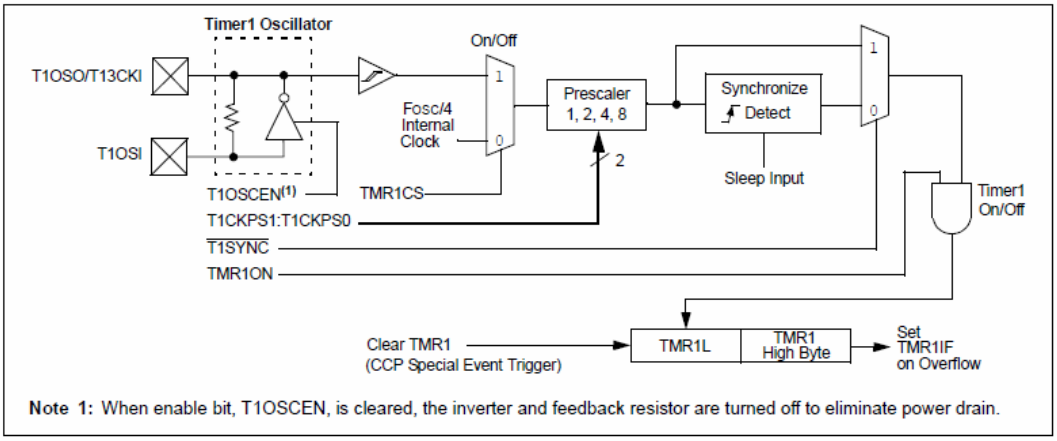
450 Hz vol dir un període de 2,22ms. Això vol dir que en 2,22ms hem de tenir el senyal a 1 i a 0.
Farem servir el mode toggle de la unitat compare amb la meitat de temps: 1’11 ms = 1110µs.
Amb un oscil·lador de 4MHz tenim fosc/4 = 1 MHz, que vol dir un període de 1µs al clock del timer.

Per tant el valor a carregar als registres CCPRH:CCPRL és 1110.
Al CCP1CON posarem: xxxx0010 mode compare i toggle.
Al T1CON hi posarem: 10000001 que configura com 16 bits, font del clock = fosc/4, no prescaler i start.

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

7) (2p.) Es vol generar una interrupció cada 1 ms, utilitzant el Timer1 del PIC18F4550. Escriu el codi C necessari per configurar el Timer1 i escriu la RSI_High per a tenir una interrupció cada 1 ms, i comptabilitzar en una variable global el nombre de milisegons des de l'inici de l'execució del codi. L'oscil·lador del PIC és de 1 MHz.

FIGURE 12-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
PIE1	SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
IPR1	SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
TMR1L	Timer1 Register Low Byte							
TMR1H	TImer1 Register High Byte							
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.
Note 1: These bits are unimplemented on 28-pin devices; always maintain these bits clear.

Fosc= 1MHz => Fcycle= 250KHz => Tcycle= 4µs

Per completar 1 ms calen 250 cicles de 4µs.

```
unsigned long milisegons=0;

void init_timer1 () {
  TMR1H=0;
  TMR1L=5;
  T1CON= ^b00000101;
  PIR1= PIR1 & 0xFE;
  IPR1= IPR1 | 0x01;
  PIE1= PIE1 | 0x01;
  INTCON= INTCON | 0xC0;
}
```

Per tant podem configurar el Timer1 com 8 bits, clock timer1= Fosc/4, prescalar 1, No Synchronize i ON.

```
void interrupt high_RSI () {
  if (TMR1IE && TMR1IF) {
        TMR1IF=0;
        TMR1L=5;
        milisegons++;
  };
}
```

No es té en compte la latència per iniciar l'execució de la RSI, ni el temps transcorregut en l'execució de la RSI fins la nova assignació al TMR1L. Per compensar aquest lapse de temps caldria que l'assignació a high_RSI fos TMR1L=12.