Pràctica 5:

Com a primera part d'aquesta pràctica hem preparat tal com es demanava un programa que encengui i apagui un LED de forma intermitent i vagi canviant el LED encès cada determinat temps. S'ha fet:

- amb un sol timer (base de temps)
- amb dos timers

1- Amb un sol timer

Amb un sol timer utilitzem les variables globals 'contador', 'encendido', posició i 'pbyReg' que utilitzarem com a punter per escriure sobre els registres que ens calgui.
Posició indicarà el LED (del primer al vuitè) que estem encenent en aquell moment i contador ens servirà per regular el temps que romandrà encès cada LED ja que base de temps provoca una interrupció cada 2.05 ms (de fet el que estem fent es utilitzar la base de temps com un pre-scaler).

```
int contador = 0;
int encendido = 0;
BYTE posicion = 1;
BYTE far *pbyReg;
```

Necessitarem inicialitzar el timer de la base de temps. Ho fem amb la següent rutina:

```
void IniTimers(void)
{
   BYTE far *pbyReg;                          //punter a un registre

                                              //substituir el vector d'interrupció de la base de temps :
  disable();                                        //no permetre cap INT
  setvect(INTBASETEMPS, RSIBaseTemps);              //indiquem la ISR que atendrà la interrupció
  enable();

  pbyReg = (BYTE far *)MK_FP(RSEG, PRC);            //crear el punter al registre de control del
                                                    //processador PRC

  *pbyReg = 0x04;                             //modificar el registre PRC per a una int cada 2.05 ms
                                              //RAM interna deshabilitada
                                              //Fclk de 4 MHz

  pbyReg = (BYTE far *)MK_FP(RSEG, TBIC);           //apuntar al registre TBIC
  *pbyReg &= 0xBF;                                  //activar interrupcions BT
}
```

La RSI que utilitzarem per atendre cadascuna de les interrupcions que generi aquesta base de temps es la següent:

```
void interrupt RSIBaseTemps(void)
{
    disable();

    if (contador%50 == 0) {
          if (encendido == 0) encendido = 1;
           else encendido = 0;

          if (encendido == 1) SetP2(posicion);
          else SetP2(0);
    }

    if (contador >= 500) {
          posicion = posicion<<1;
          if (posicion == 0) posicion = 1;
           contador = 0;
    }

     contador++;

     enable();
     FINT;
}
```

Cada vegada que es produeix una interrupció l'atenem amb aquesta RSI, on incrementem la variable contador, de manera que sabem quantes vegades hem atès la interrupció i podem controlar els temps com ens convingui.

Així, cada 50 vegades que es generi la interrupció encendrem/apagarem el LED posicio-èsim LED. Tenint en conte que es genera una interrupció cada 2.05 ms, cada 0.2 segons s'encendrà un LED (0.1 segon encès, 0.1 segon apagat).

Quan el comptador arriba a 500 canviem el LED que estàvem encenent i reiniciem el contador. Així aconseguim que cada LED s'encengui i s'apagui durant 1 segon (és a dir: abans de que canvii el LED que encenem i apaguem, el LED en qüestió s'encendrà i apagarà 5 vegades).

2**- Amb dos timers**

Bàsicament dividirem la feina que feia la RSI de la base de temps en 2 rutines d'interrupció. Una rutina per cada timer: una que encendrà i apagarà el posició-èsim LED i l'altra que modificarà aquesta posició.

```
void interrupt RSITimer1(void) {
   disable();

   posicion = posicion<<1;
    if (posicion == 0) posicion = 1;
    enable();       //permetre altres INT.
    FINT;
}
```

```
void interrupt RSITimer0(void) {
  disable();

  if (encendido == 0) encendido = 1;
  else encendido = 0;

  if (encendido == 1) SetP2(posicion);
  else  SetP2(0);

  enable();       //permetre altres INT.
  FINT;
}
```

El registre MD0 d'un dels timers serà aproximadament 10 vegades menor que el de l'altra timer per tal d'aconseguir que es cridi 10 vegades a una de les rutines per cada vegada que cridem a l'altra. La rutina cridada amb més freqüència serà l'encarregada de encendre i apagar el LED, deixant a l'altra la tasca de modificar la posició (aconseguim que cada LED s'encengui i s'apagui 5 vegades abans de que es produeixi la interrupció que canviarà de LED).

Així, els timers s'inicialitzaran tal com s'indica a continuació:

```
void IniTimers(void) {

BYTE far *pbyReg;                                   //punter a un registre

   // Iniciar el timer 0

  pbyReg = (BYTE far *)MK_FP(RSEG, MD0);            //indiquem a MD0 quin valor ha de decrementar
  pbyReg[0] = 0;
  pbyReg[1] = 13;

  pbyReg = (BYTE far *)MK_FP(RSEG, TMC0);           //permetem interrupció i escollim que decrementi
  *pbyReg = 0x80 | 0x40;                            //MD0 amb una freq de Fclk/128

  disable();
  setvect(INTCOUNTER0, RSITimer0);                  //indiquem la rutina que atendrà la interrupció
  enable();

  pbyReg = (BYTE far *)MK_FP(RSEG, TMIC0);
  *pbyReg &= 0xBF;                                  //activar interrupcions timer0


  // Iniciarel timer 1

  pbyReg = (BYTE far *)MK_FP(RSEG, MD1);
  pbyReg[0] = 0;
  pbyReg[1] = 128;

  pbyReg = (BYTE far *)MK_FP(RSEG, TMC1);
  *pbyReg = 0x80 | 0x40;

  disable();
```

```
  setvect(INTCOUNTER2, RSITimer1);
  enable();

  pbyReg = (BYTE far *)MK_FP(RSEG, TMIC2);
  *pbyReg &= 0xBF;                                    //activar interrupcions timer1
}
```

NOTA: MD0 és un registre de 16 bits! Hem estat hores barallant amb els timers ja que si modifiquem el byte de menor pes, al estar inicialitzat a 1's el registre en qüestió els bits de major pes fan que aquesta modificació que nosaltres fèiem passés desapercebuda, donant la impressió de que els timers no estan ben inicialitzats. Cal fer dos accessos i guardar dos bytes o bé fer un accés a WORD (nosaltres com es pot veure al codi hem optat per la primera opció).