UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Enunciat de la pràctica de laboratori

Gestió de Timers

Aplicació: Cronòmetre resetejable

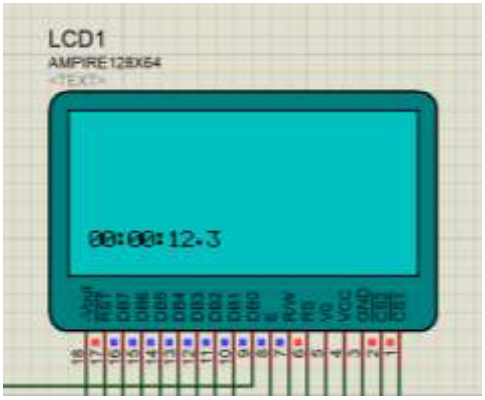
Gestió de Timers

1 Objectius

L'objectiu d'aquesta pràctica és familiaritzar-se amb l'ús dels *Interval Timers*, i comprendre com podem gestionar els diferents esdeveniments generats al microcontrolador amb la utilització del tractament d'interrupcions.

Per a tal fi programareu una aplicació de tipus temporitzador digital que mostri a la GLCD un temporitzador amb el format: hores:minuts:segons.dècimes (veure diagrama a sota). Utilitzarem un polsador per a posar en marxa i aturar el temporitzador.

pulsador
RC0



2 Descripció de la pràctica

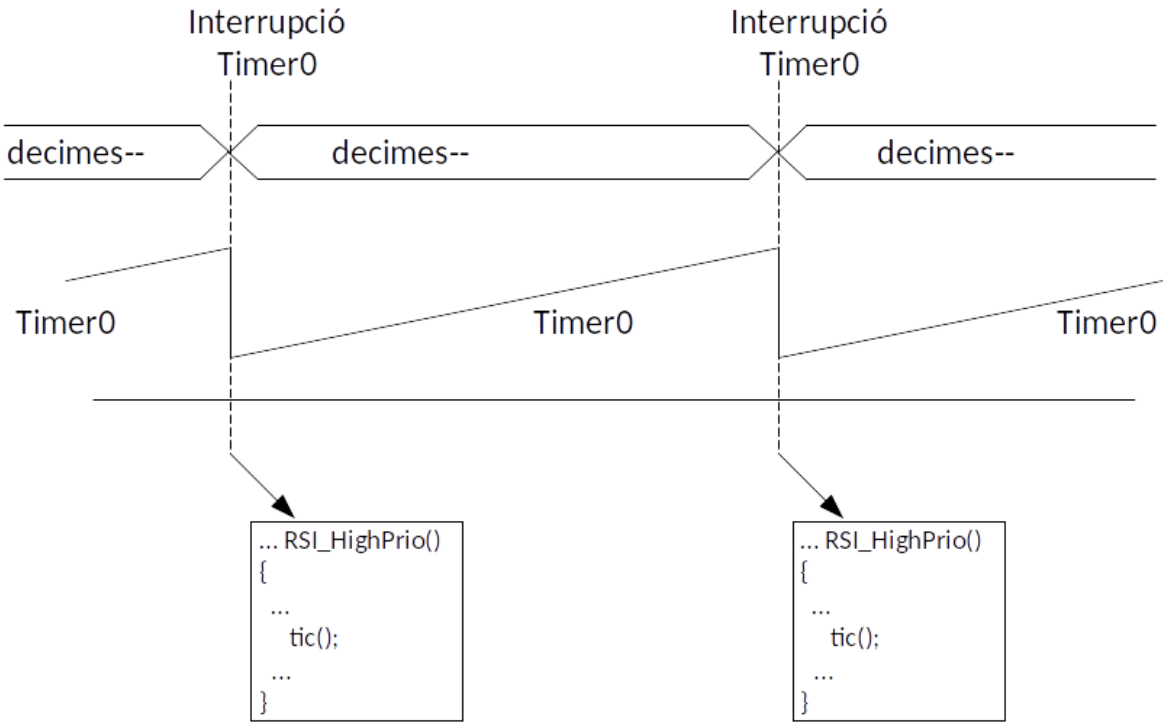
Molt habitualment necessitem comptar el temps transcorregut per tal d'executar accions periòdiques als nostres programes. Els microcontroladors ens ofereixen uns perifèrics que s'encarreguen de comptar temps de manera autònoma: els ***Timers***. El seu registre intern s'incrementa a raó d'un clock d'entrada. El PIC ens permet obtenir aquest clock d'entrada efectuant diverses manipulacions sobre l'*Instruction Cycle Clock* del microcontrolador ($F_{OSC}/4$).

Utilitzareu el Timer0 per tal de comptar el temps transcorregut, i quan hagi passat 1 dècima de segon, actualitzareu el valor del temporitzador que mostreu per la GLCD.

Per a saber quan ha transcorregut 1 dècima de segon, podríem consultar contínuament el valor del comptador de Timer0, però seria una estratègia bastant ineficient que consumiria massa % de CPU del PIC. Per evitar aquest problema, farem servir ***interrupcions***.

Programeu una subrutina ***tic()*** que s'executi cada dècima de segon mitjançant interrupcions periòdiques d'alta prioritat provinents del Timer0.

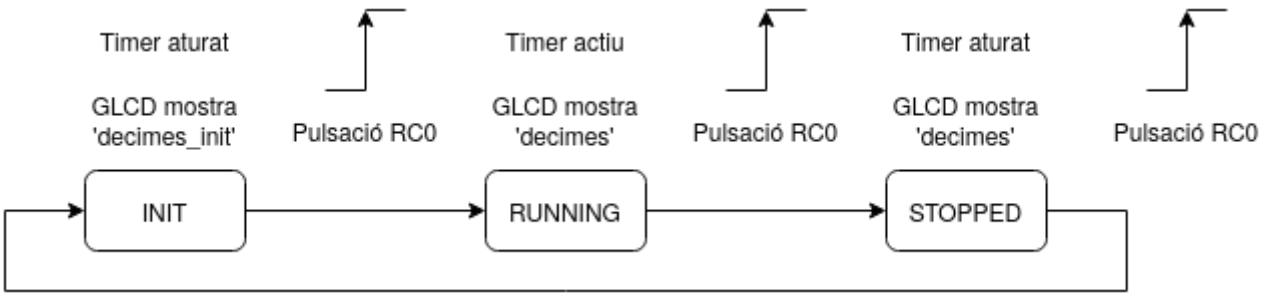
Programar un procediment que executant-se dins del programa principal (*main*) actualitzi per pantalla (GLCD) el temporitzador digital tot consultant una variable global ('decimes') que s'actualitza en la subrutina ***tic()***. El format del cronòmetre contindrà hores, minuts, segons (de 0 a 59) i dècimes de segon (HH:MM:SS.D).



El temporitzador no sempre estarà en marxa. Podrà estar en tres estats: INIT, RUNNING o STOPPED. El canvi entre estats el fareu amb el **flanc ascendent de la pulsació del botó RC0**. Gestionareu les activacions del polsador RC0 mitjançant la tècnica d'enquesta. A l'iniciar el nostre programa el deixareu en estat INIT amb el comptador a un valor definit per una constant global ('decimes_init'). Una pulsació del botó RC0 farà que passi a RUNNING i comenci a comptar (a la pantalla GLCD mostrareu el valor actualitzat del temporitzador). Si el valor del compador arriba a zero, el programa passa automaticament a STOPPED.

Durant RUNNING, una pulsació del botó RC0 farà que passi a STOPPED, en el que s'atura la comptabilització del temps. La següent pulsació del botó RC0 netejarà els comptadors al valor inicial, i passarem de nou a l'estat INIT. Com que treballeu llegint flancs, haureu d'implementar la corresponent estratègia anti-rebots.

La imatge següent mostra els canvis d'estat per polsació de RC0.



3 Treball Previ

Temps aproximat: 3 hores

1) Lectura del capítol 11 (mòdul *Timer0*) del DataSheet del PIC18F45K22
2) Respondre el qüestionari adjunt
3) Desenvolupar un programa que realitzi les funcions especificades al capítol anterior. El codi del vostre programa principal (*main*) haurà d'actualitzar per la GLCD el temps comptat pel temporitzador. Aquesta informació li vindrà donada per les variables globals que creieu necessàries ('dècimes', etc.), i que s'hauran d'actualitzar en la subrutina d'interrupcions.
4) Dissenyar l'esquema electrònic sobre PROTEUS usant els components que calguin per a poder simular la pràctica. Podeu utilitzar com a base les pràctiques anteriors on s'utilitzen la GLCD i la gestió de polsadors.
5) Realitzar l'execució, test i depurat (*debug*) del vostre programa sobre PROTEUS.

Important:

- **Treball previ i Qüestionari: l'haureu d'entregar al Racó (entrega de pràctiques via web), abans de la vostra sessió de pràctiques.**
- **També, al finalitzar la sessió, haureu d'entregar la feina realitzada durant el laboratori al Racó.**

4 Pràctica al laboratori

El treball a realitzar al laboratori consta dels següents apartats:

1) Mostrar el correcte funcionament del cronòmetre digital sobre PROTEUS.
2) Comprovar el funcionament dels programes realitzats, ara sobre la placa EASYPIC.
3) Realitzar el treball addicional que us indicarà el vostre professor.

NOTA IMPORTANT: el correcte funcionament d'aquesta pràctica, així com la seva simulació en Proteus, depenen totalment d'una configuració correcta de la freqüència de treball del micro! Recordeu que a Proteus, aquesta freqüència s'estableix a les propietats del símbol del PIC a l'esquema electrònic. Pel que fa a la placa EasyPIC, per treballar a la freqüència correcta, cal que inclogueu al vostre codi el fitxer "config.h", que us proporcionem en sessions anteriors.

Tingueu present que la simulació temporal que fa Proteus té certes limitacions. Es probable que el temps comptat pel cronòmetre a la simulació presenti petites desviacions respecte la realitat.

```
// A proposal for main.c ..... just to inspire you
#include <xc.h>
#include "config.h"
#include "GLCD.h"
....

#define _XTAL_FREQ 8000000 // Needed for __delay_ms function
#define ESTAT_INIT    0
#define ESTAT_RUNNING 1
#define ESTAT_STOPPED 2

// Global Variables (decimes, estat del crono, etc.)
unsigned int decimes=0;
unsigned char estatCrono=ESTAT_INIT;  // initial state
....

// RSI High Priority for handling Timer0
....

// Detection of Edge on Button press
char lecturaFlancRC0() {
....
}

// Initialize PORTs and basic PIC resources
void InitPIC() {
....
}

void main(void) {
....
  InitPIC();

  ....

  // MAIN LOOP
  while (1) {
    if (lecturaFlancRC0()) { // edge of button press detected
      switch(estatCrono) {
        // depending on the current state, handle the transition
        ....
      }
    }

    // show things on the GLCD
    ....
  }
}
```

Sessió de Laboratori Timers

Cognoms i Nom Ricard Guixaró Codi GRUP 21

1) Quina és la Freqüència de Clock a la que treballa el micro de la EasyPIC?

8Mhz

2) Quant temps dura un Cicle d'Instrucció (*Instruction Cycle*)?

Cada 4 tics es produeix un cicle d'instrucció, i cada tic de rellotge és 8MHz, per tant, 1 cicle instrucció = 5ms (1*4tics*(1/8MHz)).

3) Quan escrivim un valor a TMR0H, en quin moment s'actualitza el valor del registre TMR0H?

S'actualitza quan escrivim TMR0L, ja que si volguéssim llegir TMRL0, podria donar-se el cas que TMR0H canviï després de llegir la part baixa, i és per això que es guarda en un registre. D'aquesta manera, s'actualitzen els 16 bits de cop.

4) Amb quin valor programeu el *prescaler* del Timer0? Quantes unitats haureu de comptar amb el Timer0 per tal de tenir una Interrupció cada dècima de segon? Afegiu els càlculs que heu fet, per justificar les respostes.

Prescaler = ¼, i T0PS2:T0PS0 = 001. A més, s'ha d'inicialitzar el timer amb 2^{16}-50000 = 15536.

$$0{,}1 \times \frac{1tic}{\frac{4}{2x10^6}} = 50000 < 2^{16}(timer\ 16\ bits) = 65536$$

5) Quina és la situació que fa que es generi una Interrupció de Timer0?

La interrupció del TMR0 es produeix quan hi ha un overflow.