UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Enunciat de la pràctica de laboratori

# Timers

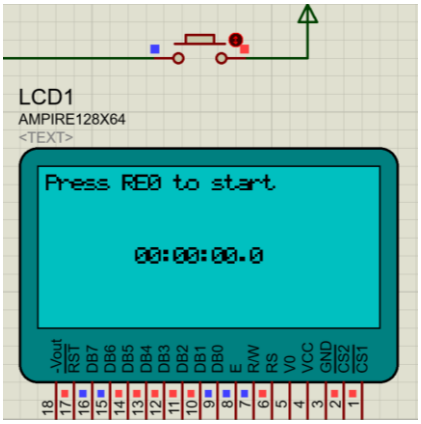
**Gestió de Timers**

# 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
RE0




**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 l'esquemàtic, fent doble click sobre el símbol del PIC i seleccionant la freqüència del clock com `Processor Clock Frequency : 8MHz`.

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 temporitzador a la simulació presenti petites desviacions respecte a la realitat.

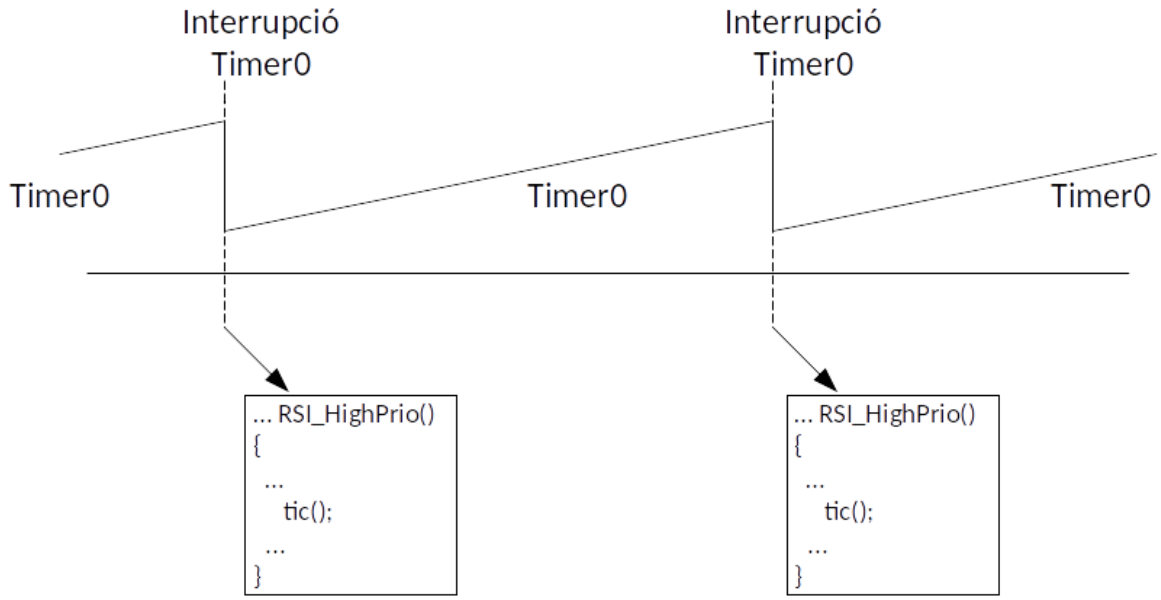
# 2 Descripció de la pràctica

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.

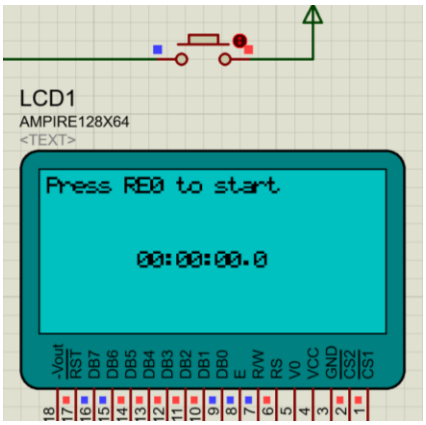
Per a saber quan ha transcorregut 1 dècima de segon, podríem consultar contínuament el valor del comptador de Timer0, però seria una estratègia bastant ineficient que consumiria massa % de CPU del PIC. Per evitar aquest problema, farem servir ***interrupcions***.

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

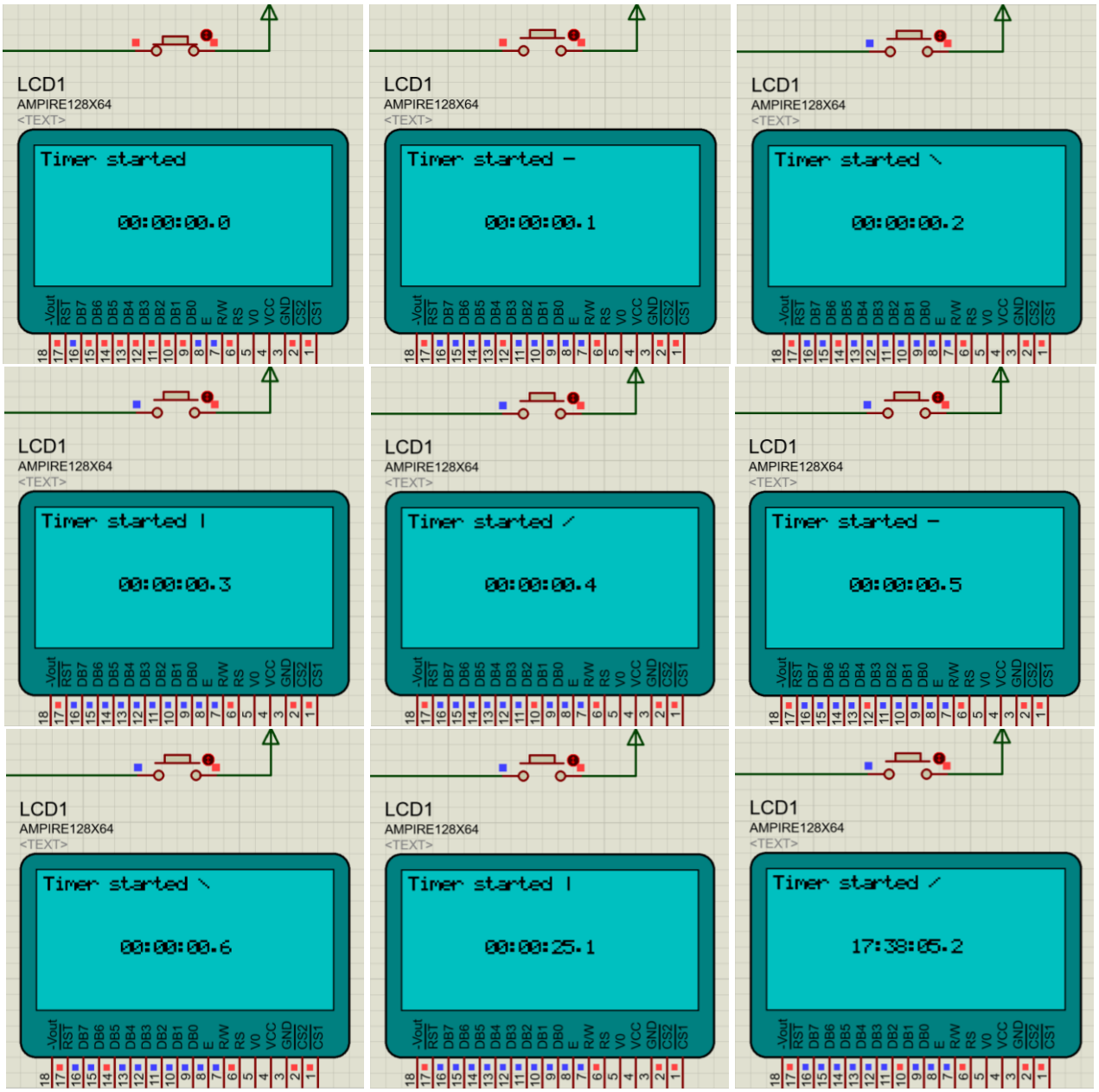



Programeu un procediment updateGLCD(·) que executant-se dins del programa principal (*main*) actualitzi per pantalla GLCD el valor del temporitzador digital tot consultant una o diverses variables globals que s'actualitzen en la subrutina ***tic()***. El format del temporitzador serà (HH:MM:SS.D).

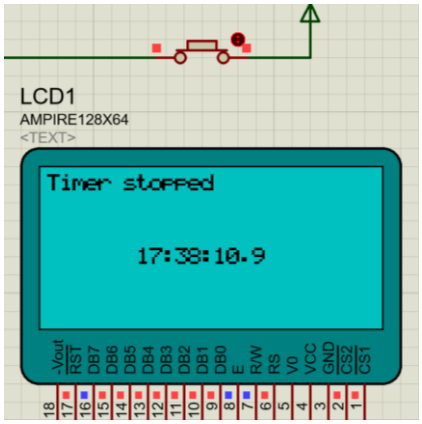
El temporitzador no sempre estarà en marxa. Podrà estar en tres estats: INIT, RUNNING o STOPPED. El canvi entre estats el fareu amb el **flanc ascendent de la pulsació del botó RE0**. Gestionareu les activacions del polsador RE0 mitjançant la tècnica d'enquesta. Com que treballeu llegint flancs, haureu d'implementar la corresponent estratègia anti-rebots.
A l'iniciar el programa, el sistema estarà en estat INIT i apareixerà la següent pantalla amb el text "Press RE0 to start" i el temporitzador inicialitzat a cero:



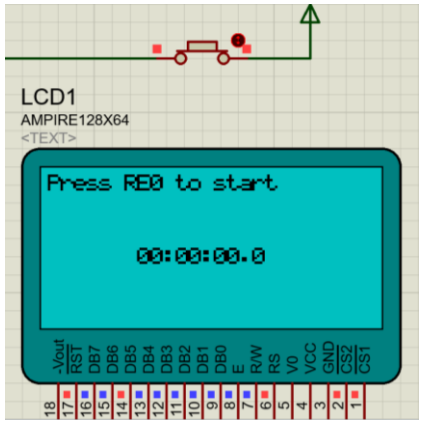
Una pulsació del botó RE0 farà que passi a RUNNING i comenci a comptar. A la pantalla GLCD mostrareu el text “Timer started”, els caràcters “-”, “\”, “|” i “/” canviant constantment per indicar que el timer està actiu, i el valor actualitzat del temporitzador:



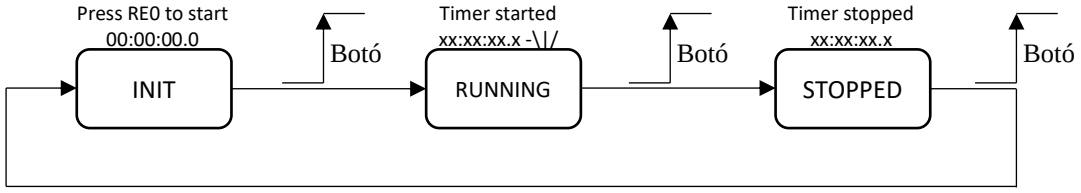
Una nova pulsació del botó RE0 farà que el programa passi automàticament a STOPPED. S'ha d'indicar amb el text "Timer stopped" i mostrar el valor del temporitzador.




La següent pulsació del botó RE0 netejarà els comptadors al valor inicial, i passarem de nou a l'estat INIT. S'ha de mostrar per pantalla el text "Press RE0 to start":




La imatge següent mostra els canvis d'estat per polsació de RE0.



# 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 el projecte sencer de proteus i el qüestionari, 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 temporitzador 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.

```
// 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 GRUP 13

Bernat Borràs Civil
Miquel Torner Viñals

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

   $F_{osc}$ = 8MHz.

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

   $t_{ci}$ = 4*ticks*4/8*$10^6$Hz = 2*$10^{-6}$s = 2µs

3) Indica els càlculs realitzats per configurar el timer0 amb una periodicitat de 0.1 segons.

   $t_{tick}$ = 4/(8*$10^6$)s = 0,5 µs

   $n_{ticks}$ = 100.000 µs / 0,5 µs = 200.000 ticks $>2^{16}$

   Prescaler = 4: 200.000/4 = 50.000 ticks $<2^{16}$

   Inicialització: $2^{16}$ – 50.000 = 15.536 = 0x3CB0

4) Indica el valor amb què has configurat els registres següents i una breu descripció de la seva funcionalitat.

```
T0CONbits.T08BIT  = 0
T0CONbits.T0CS    = 0
T0CONbits.T0SE    = 1
T0CONbits.PSA     = 0
T0CONbits.T0PS2   = 0
T0CONbits.T0PS1   = 0
T0CONbits.T0PS0   = 1
TMR0H             = 0x3C
TMR0L             = 0xB0
INTCONbits.TMR0IF = 0
INTCONbits.T0IE   = 0
T0CONbits.TMR0ON  = 1
```

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

   Quan TMR0 es produeixi un overflow: TMR0 = 0xFFFF + 1