UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Enunciat de la pràctica de laboratori

Lab 4:
Display gràfic de cristall líquid GLCD

L4. GLCD

L4(A) Funcions bàsiques de presentació d'informació

L4(B) Rellotge amb timer

1 Objectius

L'objectiu d'aquesta pràctica és el desenvolupament de codi per presentar informació en una pantalla GLCD (Graphic Liquid Crystal Display). Els exercicis sobre la GLCD estan dividits en dues parts pràctiques. Cadascuna de les parts es farà en una sessió de laboratori diferent.

La primera part anomenada **L4(A),** és una introducció al funcionament del display gràfic i consta d'unes subrutines bàsiques per a programar la pantalla gràfica.

A la segona part de la pràctica anomenada **L4(B)**, farem servir els timers del microcontrolador per programar un rellotge que es presentarà a la pantalla gràfica.

2 Introducció

A la figura 1 podeu veure l'esquema de Proteus de la pantalla GLCD connectada al microcontrolador.

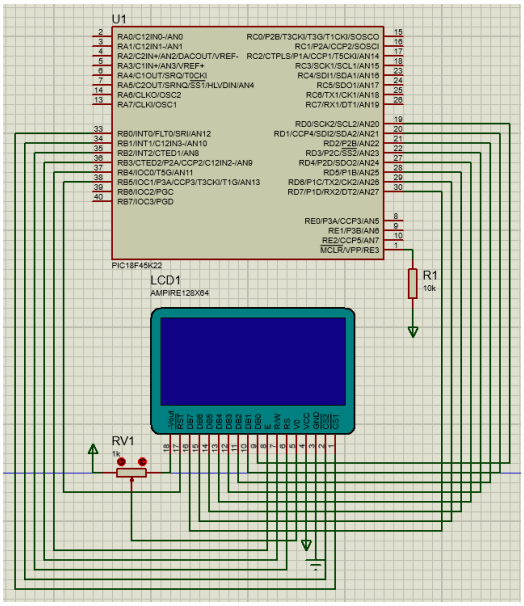

Figura 1. Connexió de la GLCD al PIC

Fixeu-vos que s'ha connectat la pantalla als ports B i D del microcontrolador. El port B té una sèrie de senyals de control de la pantalla (Enable, Reset, Read/Write...) i el port D serà un bus de 8 bits per on s'envien dades/informació a la pantalla (posició a pintar, caràcters a pintar, ordres d'esborrar,

moure cursor, etc.). Podeu veure els detalls de les connexions de la GLCD a la taula 1, extreta del datasheet de la pantalla i disponible a Atenea.

Pin No.	Symbol	Level	Description
1	$\overline{CS1}$	L	Select Segment 1 ~ Segment 64
2	$\overline{CS2}$	L	Select Segment 65 ~ Segment128
3	Vss	0V	Ground
4	V_{DD}	5.0V	Supply voltage for logic
5	V_O	(Variable)	Operating voltage for LCD
6	D/I	H/L	H: Data , L: Instruction
7	R/W	H/L	H: Read(MPU Module) , L :Write(MPU→ Module)
8	E	H	Enable signal
9	DB0	H/L	Data bus line
10	DB1	H/L	Data bus line
11	DB2	H/L	Data bus line
12	DB3	H/L	Data bus line
13	DB4	H/L	Data bus line
14	DB5	H/L	Data bus line
15	DB6	H/L	Data bus line
16	DB7	H/L	Data bus line
17	RST	L	Reset the LCM
18	VEE		Negative Voltage Output
19	A		Power supply for B/L(+)
20	K		Power supply for B/L(-)

Taula 1. Pinout de la GLCD

Com a documents annexos a aquesta pràctica, trobareu:

- Projecte base de Proteus amb la pantalla GLCD (L4_GLCD_Intro.pdsprj).
- Datasheet de la GLCD (WDG0151-TMI-V-N00-Specification.pdf).

A la següent figura, es mostra l'aspecte del projecte base L4_GLCD_Intro.pdsprj carregat a Proteus:

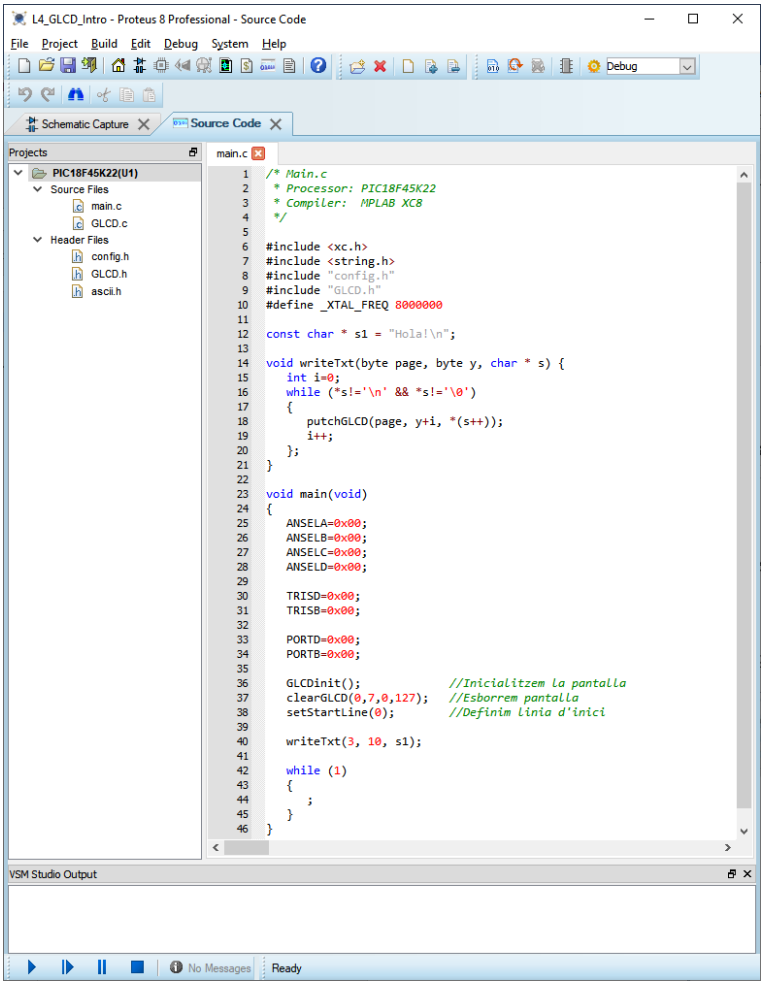

Figura 2. Captura del projecte a Proteus.

Com podeu observar, en el projecte hi han diferents fitxers .c i .h per poder usar llibreries, tenir fitxers de configuració, de dades, etc.
Concretament teniu el fitxer `main.c` amb el codi principal que configura i utilitza la pantalla, el fitxer `GLCD.c` amb les funcions de baix nivell de la pantalla, el fitxer `GLCD.h` amb les capçaleres de les funcions i el fitxer `ascii.h` on es defineixen els patrons per poder pintar caràcters a la pantalla.

Si executeu el programa, heu de veure la pantalla funcionant amb el següent text:

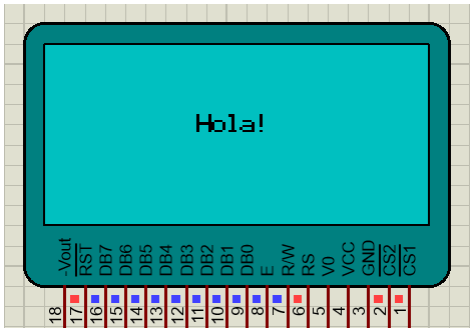

Figura 3. Demo de funcionament de l'exemple.

A partir d'aquest programa d'exemple, s'espera que desenvolupeu el vostre treball que s'especifica en les pàgines següents.

3 Treball Previ L4 (A)

1. Estudiar els diferents fitxers entregats amb la pràctica. Manuals i **sobretot el codi de la llibreria `GLCD.h` i `GLCD.c` per fer servir la pantalla**.

2. Modificar el projecte, programa i esquema base, perquè realitzi les següents funcions:

   2.1 Fer una presentació que mostri el nom de la pràctica "L4A GLCD" i el noms cognoms dels integrants del grup al centre de la pantalla.

   2.2 Afegir **4 botons** a l'esquemàtic, en 4 pins que considereu **adequats**, i configurar-los adequadament com a pins d'entrada. Aquest botons permetran moure's per la pantalla amunt, a baix, a l'esquerra o a la dreta. Els botons per moure a dreta i esquerra **no tenen detecció de flancs**. Els botons per moure a dalt i a baix **sí tenen detecció de flanc de pujada i baixada respectivament**.

   2.3 Afegir una estratègia anti-rebots pels botons que fan el moviment cap amunt i cap a baix.

   2.4 Dibuixar un punt a la part central esquerra de la pantalla. Si apretem el botó d'anar a la dreta o a l'esquerra, el punt es mourà a la dreta o esquerra mentre estigui apretat. Si apretem el botó d'anar cap a dalt, el punt es mourà cap amunt i tornarà a la posició original com si fes un salt. De forma similar es farà al apretar el botó cap a baix.

   2.5 Afegir un text indicant la posició del punt a la part superior de la pantalla.

   Teniu un vídeo disponible a Atenea on es mostra el funcionament desitjat.

Es valorarà l'estructura del codi, l'aprofitament dels recursos del microcontrolador, el bon ús de variables i la utilització de funcions auxiliars reutilitzables.

4 Rúbrica treball Previ L4 (A)

	Iniciat (0-2.5 punts)	En desenvolupament (2.5-5.0 punts)	Aconseguit (5.0-7.5 punts)	Exemplar (7.5-10 punts)
Pantalla noms (1punt):	No hi ha	No es pot reaprofitar el codi	No està centrada	Funciona perfectament
Botons (2.5 punts):	Hw mal dissenyat i funcionament erroni	Hw mal dissenyat o funcionament erroni (apretats, flancs) o no hi ha estratègia anti-rebots	Funcionament correcte però hi han errors en l'estratègia anti-rebots.	Funciona perfectament i l'estratègia anti-rebots és modular i reutilitzable
Punt a la pantalla (5 punts):	No funciona	Funciona malament, hi han problemes amb el punt, els límits el refresc de la pantalla	Funciona malament, hi ha problemes amb el punt, o els límits o el moviment cap a dalt (salt) o cap a baix, o el refresc de la pantalla	Funciona perfectament, el punt no surt dels límits, s'actualitza la pantalla només quan és necessari, el moviment cap a dalt (salt) i cap a baix és correcte, etc.
Text part superior (1.5 punts):	No funciona bé	S'actualitza massa sovint	Codi sense optimitzar	Funciona perfectament i el codi és modular, reutilitzable i actualitza els valors només quan és necessari

5 Treball Previ L4 (B)

L'objectiu d'aquesta segona part de la 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.

Habitualment, en la programació de microcontroladors, 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***.

Tasques a realitzar

El programa a realitzar consisteix en una aplicació de tipus **compta enrera** que mostri a la GLCD un temporitzador digital amb el format: segons.dècimes (veure figura a sota). Utilitzarem un polsador per a posar en marxa, aturar i resetejar el temporitzador.

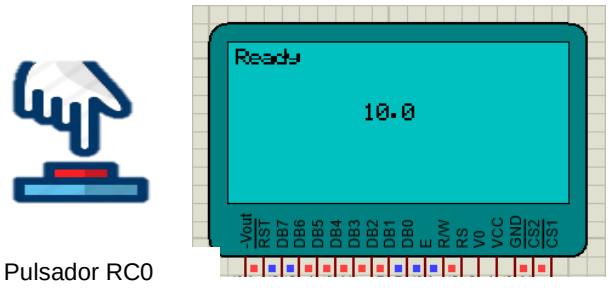

Figura 4. Captura de pantalla a Proteus amb cronòmetre i botó RC0.

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

Les tasques a realitzar són les següents:

1. Tornar a presentar la pantalla inicial que mostri el noms i cognoms dels integrants del grup i el text “L4B GLCD”.

2. Programeu una subrutina `tic()` que s’executi cada dècima de segon mitjançant interrupcions periòdiques d’alta prioritat provinents del Timer0. Aquesta rutina és l’encarregada de cridar les accions a realitzar cada 0.1 segons des de la rutina d’atenció a la interrupció del timer 0.

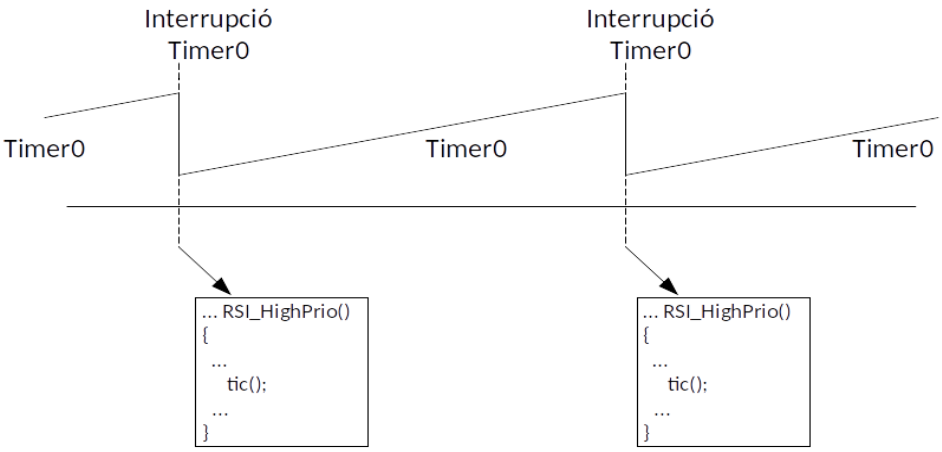

Figura 5. Esquema de funcionament del comptador del timer, la generació d’interrupcions i la crida de la funció `tic()` a la rutina d’atenció de la interrupció.

3. 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 (ss.d), i tal i com apareix a la Fig. 6 es situarà al mig de la pantalla GLCD.

4. El temporitzador no sempre estarà en marxa. Podrà estar en tres estats: *Ready*, *Running* o *Stopped*. El canvi entre estats el fareu amb el **flanc descendent de la pulsació del botó RC0**. Gestionareu les activacions del polsador RC0 mitjançant la tècnica d’enquesta. Implementeu una estratègia anti-rebots que eviti la detecció de flancs no desitjats degut a problemes mecànics del botó.

Al iniciar el programa, el sistema estarà en estat READY i apareixerà la següent pantalla amb el text “Ready” i el temporitzador inicialitzat a deu:

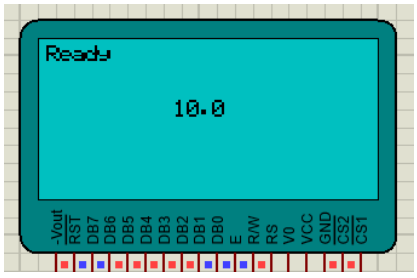

Figura 6. Estat inicial

Una pulsació del botó RC0 farà que passi a RUNNING i comenci a comptar. A la pantalla GLCD mostrareu el text “Running...”. També s’haurà de mostrar el valor actualitzat del temporitzador:

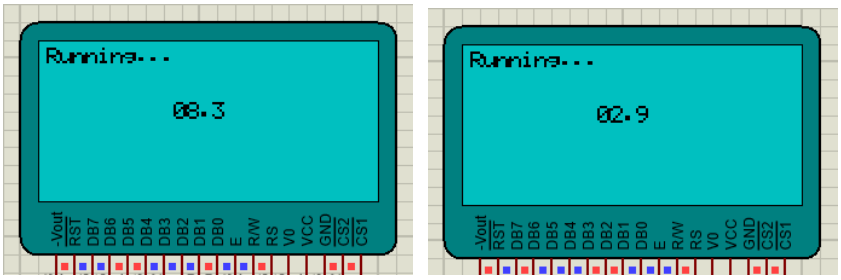

Figura 7. Estat running

Una nova pulsació del botó RC0 farà que el programa passi automàticament a STOPPED. S’ha d’indicar amb el text “Stopped!”. També s’arriba a l’estat Stopped si acaba el compta enrere quan arriba a 00.0.

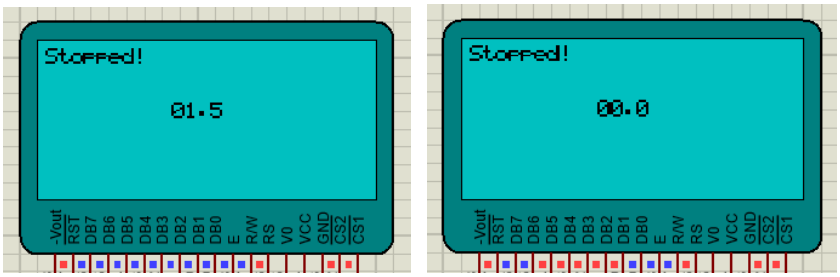

Figura 8. Estat stopped

La següent pulsació del botó RC0 netejarà els comptadors al valor inicial, i passarem de nou a l’estat READY. S’ha de mostrar per pantalla el text “Ready”:

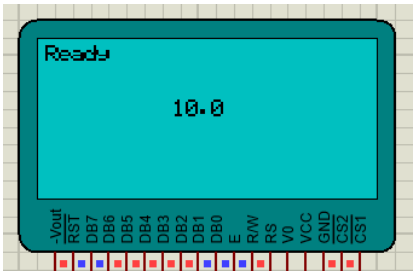

Figura 9. Tornada a l’estat inicial

```
// 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

typedef enum{
   Ready=0,
   Running,
   Stopped
}state_t;

// Global Variables
unsigned int decimes=0;
....

void tic(void)
{
    //Coses a fer periòdicament
}

// RSI High Priority for handling Timer0
...    if (timer0_flag) {
           tic();
       }


// Falling edge detection and debouncing
??? inputDetector(·) {
....
}

// Initialize PORTs, timer0 and basic PIC resources
void configPIC() {
....
}

void main(void) {
....
  configPIC();

  ....

  // MAIN LOOP
  while (1) {
    if ( inputDetector(·) ) { // check falling edge
      switch(estatCrono) {
        // depending on the current state, handle the transition
        ....
      }
    }

    updateGLCD(·);// show things on the GLCD smartly!
    ....
  }
}
```

6 Rúbrica treball Previ L4 (B)

	Iniciat (0-2.5 punts)	En desenvolupament (2.5-5.0 punts)	Aconseguit (5.0-7.5 punts)	Exemplar (7.5-10 punts)
Pantalla noms (0.5 punts):	No hi ha	No es pot reaprofitar el codi	No està centrada	Funciona perfectament
Transició botó RC0 (flanc de baixada) (0.5 punts):	Hw mal dissenyat i funcionament erroni	Hw ben dissenyat però funcionament erroni	Funcionament correcte però no hi ha estratègia anti-rebots	Funcionament correcte amb estratègia anti-rebots reutilitzable
Programació Timer0 (2.5 punts)	Hi han més de 3 errors en la programació dels registres del timer i interrupcions	Hi han entre 2 i 3 errors en la programació dels registres del timer i interrupcions	Hi han entre 1 i 2 errors en la programació dels registres del timer i interrupcions	No hi han errors en la programació dels registres del timer i interrupcions
Màquina d'estats (1.5 punt):	No hi ha estats ni codi per fer les transicions	No hi ha estats o no hi ha codi per fer les transicions	O els estats o les transicions tenen algun error	Estats i transicions perfectament implementats
Pantalla GLCD (3 punts)	La pantalla no presenta la informació correctament, apareixen punts on no toca, s'actualitza la pantalla en llocs del codi no adients	La pantalla no presenta la informació correctament, o apareixen punts on no toca, o s'actualitza la pantalla en llocs del codi no adients	La pantalla funciona bé però es fan més actualitzacions de les necessàries o l'estructura del codi no està ben raonada	Funciona perfectament i s'aprofiten bé tots els recursos i l'estructura del codi és correcta
Qüestionari (2 punts)	Pregunta 1 = 0.2 punts. Pregunta 2 = 0.2 punts. Pregunta 3 = 0.2 punts. Pregunta 4 = 0.2 punts. Pregunta 5 = 0.2 punts. Pregunta 6 = 1.0 punts.			

FULL DE RESPOSTES–L4 (B) GLCD

(s'ha d'entregar en format electrònic com a treball previ de la L4 (B))

Nom i Cognoms: _____Grup LAB: _____

_____

1) Quina és la freqüència de clock a la que treballa el micro de la EasyPIC?

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

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

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

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

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

6) Dibuixa el diagrama de blocs del codi per fer el compta enrere.