UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

Enunciat de la pràctica de laboratori

Ports d'entrada/sortida

Ports d'entrada/sortida

1. Introducció

En aquesta pràctica es prendrà contacte amb la plataforma de desenvolupament EasyPIC v7 (Fig.1) que utilitzarem al llarg de tot el curs. A partir de la documentació proporcionada, identificareu components en esquemes electrònics, coneixereu l'entorn de programació, veureu com descarregar els programes a la placa de desenvolupament i fareu unes proves senzilles per agafar confiança amb el conjunt.

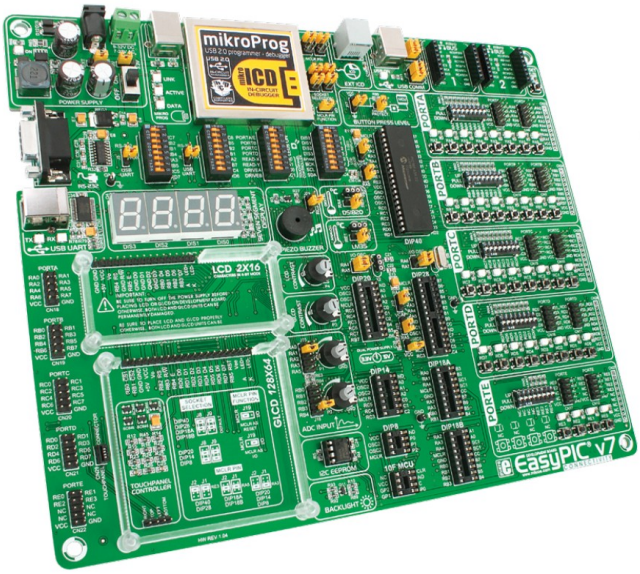

Fig1. Plataforma de desenvolupament EasyPIC v7

El *Integrated Development Environment* (IDE) utilitzat a les pràctiques serà el Proteus. Proteus integra un editor d'esquemàtics, editor de codi, compilador, assemblador i simulador en temps real. Per aquesta pràctica i fins a acabar el quadrimestre, es farà servir el **compilador XC8** (llenguatge C) disponible a:
https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers
A l'hora de crear un nou projecte amb Proteus i seleccionar el *firmware* amb el que treballareu, també haureu d'indicar el tipus de compilador; aneu amb atenció, doncs per defecte el programa proposa MPASM (*assembler*). L'heu de canviar per **MPLAB XC8**.

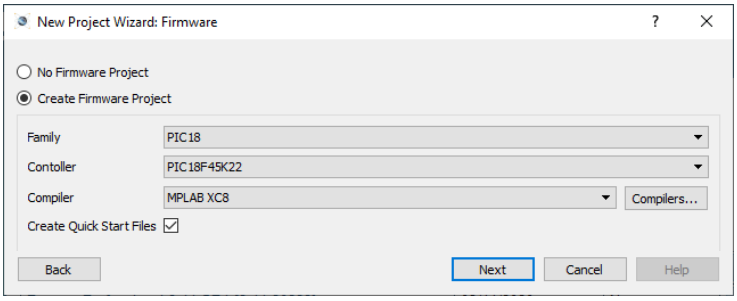

Fig2. Captura de pantalla per la generació d'un nou projecte amb Proteus.

Per fer servir la placa de desenvolupament s'ha d'afegir l'arxiu "config.h" disponible a Atenea, i guardar-lo a la carpeta del projecte on esteu treballant. Per afegir aquest arxiu, aneu a la finestra Source Code, clickeu al menú Source→Add Files i busqueu l'arxiu config.h. A dintre del main heu d'incloure #include "config.h"
Un cop hagueu dibuixat l'esquema del vostre circuit, i hagueu programat el codi que voleu que executi el PIC, es plantegen dos escenaris per a provar-ho:

- SIMULACIÓ: haureu de compilar el codi en "mode *Debug*" (a la finestra "*Source Code*" del Proteus). Fent això, el procés de compilació generarà un arxiu amb extensió ".cof" i permetrà simular l'execució del PIC en el vostre esquemàtic *hardware*.

- EXECUCIÓ A LA PLACA EASYPIC: haureu de compilar el codi en "mode *Release*". Fent això, el procés de compilació generarà a la vostra carpeta de treball un arxiu amb extensió ".hex" que és executable sobre el PIC. Per executar el programa a la EasyPIC, haureu de descarregar aquest fitxer ".hex" a la placa a través del programador connectat al bus USB. La descàrrega del *firmware* a la placa la fareu a través del programa **mikroProg**. Un cop carregat el *firmware*, el microcontrolador provocarà un reset i executarà la vostra aplicació.

2. Objectius

L'objectiu d'aquesta pràctica és encendre i apagar uns LEDs connectats a un port (que configurarem com a sortida) a partir dels senyals dels polsadors connectats a un altre port (que configurarem com a entrada).

En acabar la pràctica l'alumne serà capaç de:

- manegar informació tècnica donada pel fabricant de la placa de desenvolupament que utilitzem a les pràctiques.
- cercar la informació necessària dins dels manuals de referència.
- identificar els elements 'polsador' i 'LED' i com es connecten al microcontrolador.
- fer un primer experiment respecte a les diverses funcionalitats dels pins del microcontrolador.
- crear i configurar un projecte amb la plataforma de desenvolupament.
- *debugar* el programa, utilitzar *breakpoints* i *tracejar* els valors de les diferents variables i registres.
- descarregar els fitxers executables ja generats a la placa de desenvolupament.
- entendre el funcionament del conjunt, veient la relació entre el codi i el comportament del microcontrolador.
- gestionar adequadament senyals d'entrada per nivell i per flanc, i utilitzar variables d'estat.

3. Treball previ

Temps estimat: 3 hores.

1- Cal entendre la configuració interna dels I/O PORTS A, B, C, D i E del microcontrolador (pàgines 127-136 del *PIC18F45K22 Data Sheet*). En particular la configuració del PORTA i PORTB com a port digital mitjançant els registres ANSELx, així com la configuració dels ports com a IN o OUT mitjançant els registres TRISx.

2- Disseny de l'esquema elèctric sobre Proteus. Cal dissenyar el circuit basat en l'esquemàtic que trobareu a la documentació lliurada sobre la placa EasyPIC v7 (i a la figura 3). Tingueu sempre present que heu de copiar de l'esquemàtic només aquells components que necessiteu per a fer la pràctica de forma similar als circuits que es demanaven a la pràctica anterior (**NO intenteu dissenyar la placa sencera**). Vigileu que l'entrada MCLR (Master Reset) estigui a 1, per evitar que hi hagi resets aleatoris si queda a l'aire.

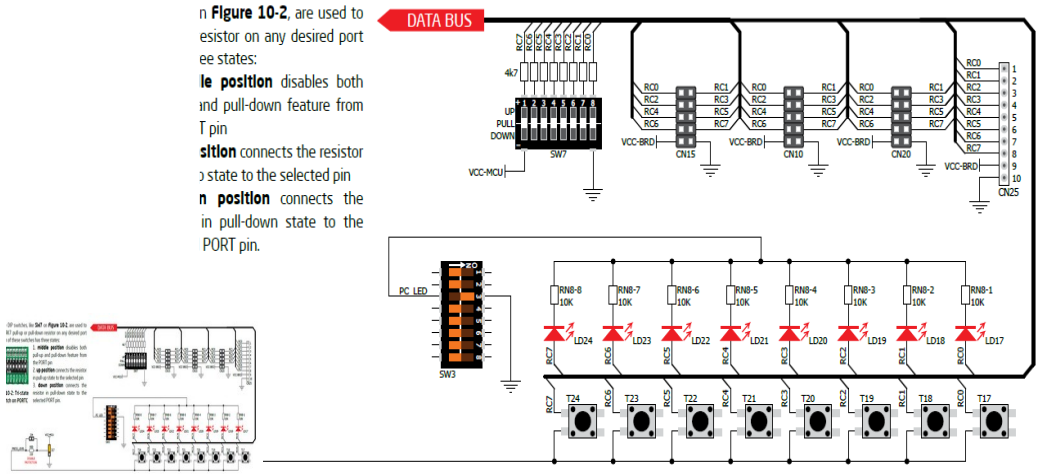

Fig 3. Esquema dels polsadors i LEDs del PORT C a la placa EasyPIC v7

RECORDEU: la figura 3 (que prové del manual de la placa) és molt exhaustiva, i intenta representar tots els components que hi ha a la EasyPIC relacionats amb el PORT C. Heu de seleccionar només els components que estan involucrats amb les vostres funcionalitats i reproduir-los al Proteus.

3- Fer un programa en llenguatge C en el que cada cop que es premi el polsador associat al bit 0 del PORTA (RA0) s'apagui el LED associat al bit 1 del PORTA (RA1) i quan es deixi anar el polsador s'encengui el LED (veure figura 4). S'espera que aquesta versió del codi es realitzi **per enquesta**: el vostre programa consultarà contínuament l'estat de l'entrada associada al polsador.

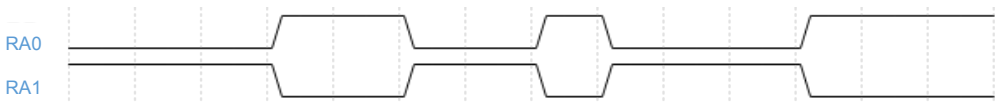

Fig 4. Cronograma simple

4- Afegiu al programa la següent funcionalitat:

Inicialitzeu un comptador de 8 bits a zero. Els 8 LEDs associats al PORTC mostraran el valor del comptador (en format binari) en tot moment. Cada cop que es detecti un **flanc de pujada** del polsador associat a RE0, el comptador s'incrementa en una unitat.

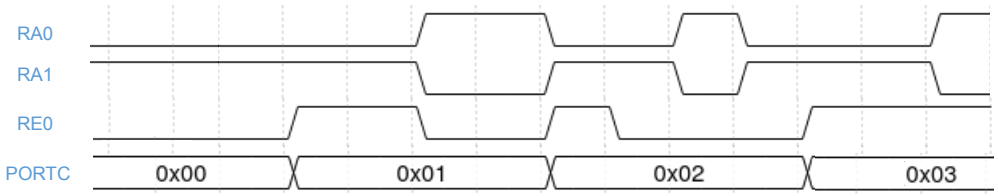

Fig 5. Cronograma amb comptador

5- Afegiu al programa la següent funcionalitat:
Cada cop que es detecti un **flanc de baixada** del polsador associat a RE1, el comptador es reseteja amb el valor 0.

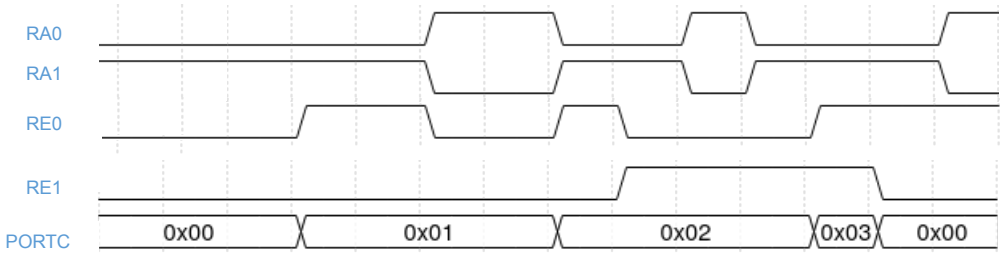

Fig 6. Cronograma amb reset

6- Afegir un programa en llenguatge C que implementi la funció

$$y = \text{not}(In1 \text{ and } \text{not}(In2)) = \overline{In1 \cdot \overline{In2}}$$

on *In1* correspongui a l'estat del polsador RB0, *In2* correspongui a l'estat del polsador RB1 i *y* correspongui a un led connectat al pin RB2.

7- Execució, test i *debugat* dels vostre programa sobre Proteus.
No us oblideu d'incloure el fitxer 'config.h' abans del vostre codi.
A continuació, poseu un *software breakpoint* després de la lectura RE1 quan aquest canvia d'estat. Observeu el valor del ANSELx, TRISx, LATx i PORTx, un cop s'atura l'execució.
Poseu un *hardware breakpoint* que s'activi quan s'encengui el LED associat a RC0. Observeu el valor del ANSELx, TRISx, LATx i PORTx un cop s'atura l'execució. Observeu com varien aquests valors a mesura que seguim executant instruccions del programa.
Podeu afegir el *hardware breakpoint* a partir de la barra d'eines situada a l'esquerra del Proteus anomenada *Probes* → *Voltage* → *Real Time Breakpoint* → *Digital* → *Trigger Value=1* o afegint un component *RTVBREAK* a partir de la llibreria Debugging Tools.

Entregueu pel Racó, ABANS de la vostra sessió de pràctiques:

- el Full d'entrega que trobareu al final d'aquest document

- el projecte PROTEUS (valideu abans d'enviar que al obrir-lo hi apareix el codi, l'esquema, ...). Per a garantir compatibilitats de versions, lliureu sempre el vostre treball salvat en una versió compatible de PROTEUS (versió actual als laboratoris, v8.4 SP0).

4- Rúbrica treball Previ Lab2 Entrada Sortida

	Iniciado (0-2.5 puntos)	**En desarrollo (2.5-5.0 puntos)**	**Conseguido (5.0-7.5 puntos)**	**Ejemplar (7.5-10 puntos)**
Proteus circuit 1 negador (1 punts):	Hw i Sw tenen errors	Hw funciona però Sw falla	Sw funciona però Hw falla	Sw i Hw funcionen
Proteus circuit 2 comptador (1.5 punts):	Hw i Sw tenen errors	Hw funciona però Sw falla	Sw funciona però Hw falla	Sw i Hw funcionen. El flanc de pujada està correctament implementat.
Proteus circuit 3 reset (1.5 punts):	Hw i Sw tenen errors	Hw funciona però Sw falla	Sw funciona però Hw falla	Sw i Hw funcionen. El flanc de baixada està correctament implementat.
Proteus circuit 4 y = not(In1 and not(In2)) (2 punts):	Hw i Sw tenen errors	Hw funciona però Sw falla	Sw funciona però Hw falla	Sw i Hw funcionen
Proteus sw breakpoint (1 punts):	Pregunta 2 full d'entrega incorrecta i breakpoint no funcional a proteus	Pregunta 2 full d'entrega correcta i breakpoint no funcional a proteus	Pregunta 2 full d'entrega incorrecta i breakpoint funcional a proteus	Pregunta 2 full d'entrega correcta i breakpoint funcional a proteus
Proteus hw breakpoint (1 punts):	Pregunta 3 full d'entrega incorrecta i breakpoint no funcional a proteus	Pregunta 3 full d'entrega correcta i breakpoint no funcional a proteus	Pregunta 3 full d'entrega incorrecta i breakpoint funcional a proteus	Pregunta 3 full d'entrega correcta i breakpoint funcional a proteus
Diagrama flux full d'entrega (2 punts):	El diagrama és incongruent	El diagrama no clarifica el funcionament dels circuits	Presenta de forma clara la majoria de circuits	Presenta de forma clara els estats de tots els circuits

5. Pràctica al laboratori

A l'inici de la sessió, l'alumne ha de mostrar el programa de l'apartat 3 corrent sobre Proteus. Un cop comprovat el funcionament correcte, la pràctica consistirà en els següents apartats:

1- Configuració de la placa de forma que:

- Estiguin actius només els LEDs de sortida del PORTA, PORTB i PORTC (interruptors SW3).
- Els polsadors d'entrada del PORTE generin un 0 al estar lliures i un 1 a l'estar premut (*jumper* J17 i interruptors SW9). Veure figura 7.

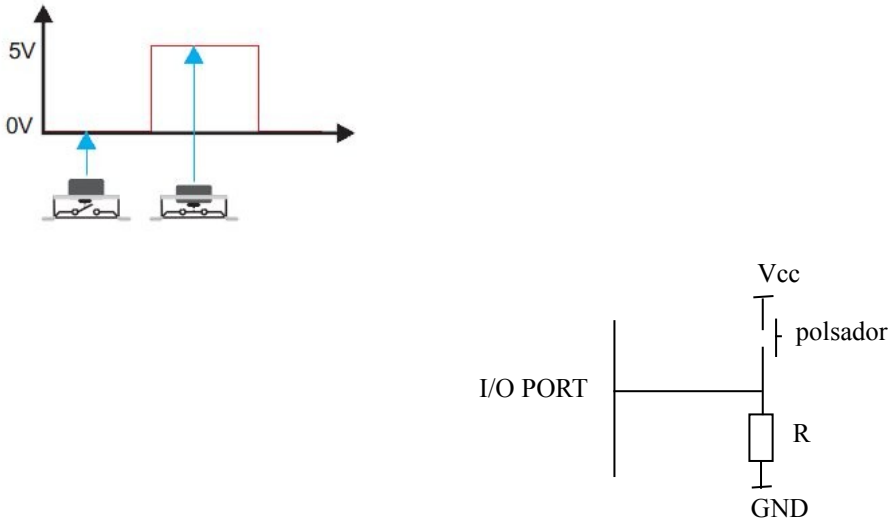

Fig 7. Configuració dels polsadors

2- Per poder fer/comprovar la correcta configuració de la placa de prototipus, cal entendre bé el connexionat dels LEDs i dels polsadors als ports d'entrada i sortida del microcontrolador (pàgines 22 i 23 del *EasyPIC User's Manual*). En particular cal vigilar la configuració del *Switch* 3 (SW3), el *Jumper* 17 (J17).

3- També cal conèixer la configuració de *switches* de la placa en relació als *pull-ups* i *pull-downs* de les entrades. En el cas del PORTE s'encarreguen els interruptors SW9. (pàgina 22 del *EasyPIC User's Manual*).

4- Baixar el vostre primer programa a la placa fent servir **mikroProg** (pàgines 10-12 del *mikroProg User's Manual*). Provar que funciona. Si no és així corregir la posició dels *switches* i *jumpers*. Quan funcioni ensenyeu l'execució del codi al professor.

5- Baixar el vostre segon programa a la placa. Provar que funciona. Si no és així no cal que corregiu la posició dels *switches* i *jumpers*, que serà correcta si funciona el vostre primer programa. El problema caldrà buscar-lo en el vostre codi. Seguiu les indicacions del professor. Quan funcioni ensenyeu l'execució del codi al professor.

6- En aquest punt el professor us donarà un nou enunciat "in-situ". Implementeu les modificacions de codi i/o configuració de la placa que el professor us demani.

```
// A proposal for your code... Just to inspire you

#include <xc.h>
#include "config.h"

void configPIC(){
  ANSELA=...;  // All pins as digital
  ANSELB=...;
  ANSELC=...;
  ANSELD=...;
  ANSELE=...;

  TRISA=...;
  TRISB=...;
  TRISC=...;
  TRISD=...;
  TRISE=...;
}

void main(void){
  configPIC();
  while(1) {
    // your code here
  }
}
```

Noms:_____GRUP:
_____

Full d'entrega

1) Dibuixeu els diagrames de flux del programa FINAL implementat (l'aconseguit al final del punt 3.7). Podeu mirar al següent enllaç per a més informació:
https://ca.wikipedia.org/wiki/Diagrama_de_flux

2) Indica els valors dels registres després del *software breakpoint*

ANSELA=	TRISA=	LATA=	PORTA=
ANSELB=	TRISB=	LATB=	PORTB=
ANSELC=	TRISC=	LATC=	PORTC=
ANSELD=	TRISD=	LATD=	PORTD=
ANSELE=	TRISE=	LATE=	PORTE=

3) Indica els valors dels registres després del *hardware breakpoint*

ANSELA=	TRISA=	LATA=	PORTA=
ANSELB=	TRISB=	LATB=	PORTB=
ANSELC=	TRISC=	LATC=	PORTC=
ANSELD=	TRISD=	LATD=	PORTD=
ANSELE=	TRISE=	LATE=	PORTE=