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.

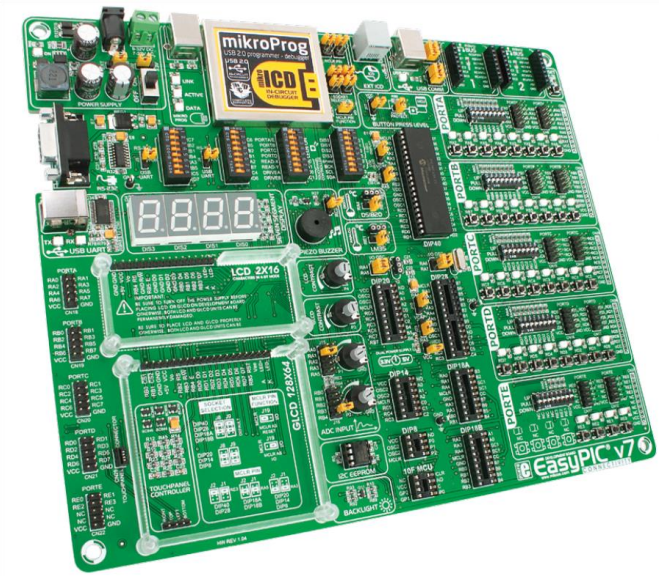


Fig1. Plataforma de desenvolupament EasyPIC v7

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**.

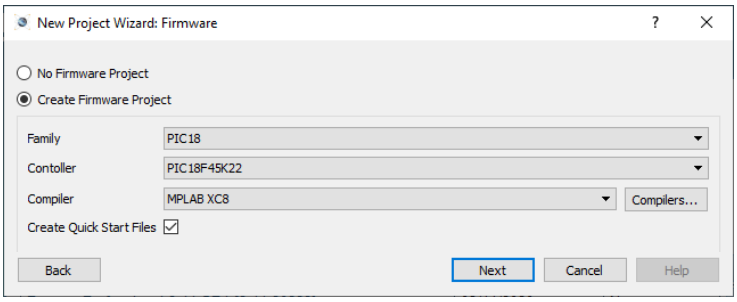


Fig2. Captura de pantalla per la generació d'un nou projecte amb Proteus.

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: 4 hores.

1. Cal entendre la configuració interna dels I/O PORTS del microcontrolador (pàgines 127-136 del *PIC18F45K22 Data Sheet*). En particular la configuració

del PORTA, PORTC i PORTE 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 (**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.

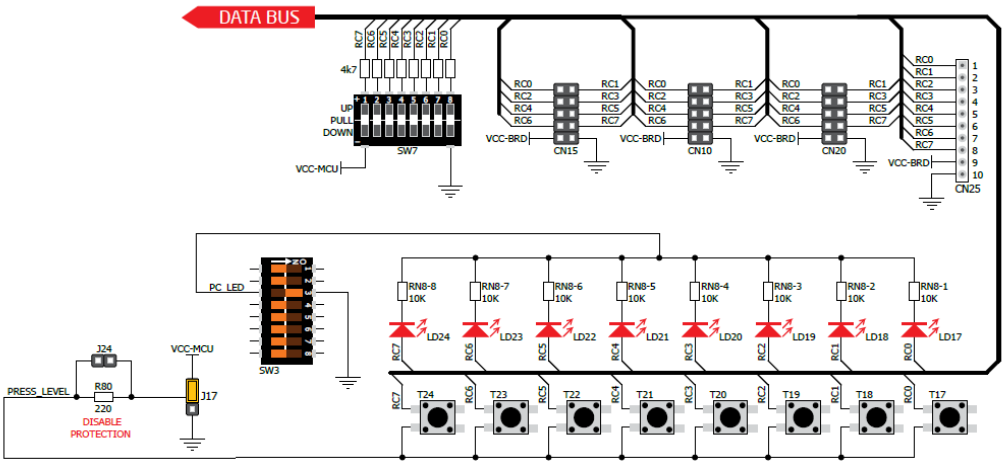


Fig 3. Esquema dels polsadors i LEDs del PORT C a la placa EasyPIC v7

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. 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 PORTE (RE0) s'apagui el LED associat al bit 0 del PORTC (RC0) 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.

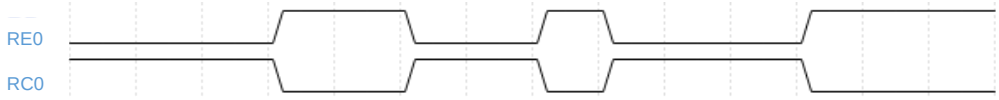


Fig 4. Cronograma simple

4. Afegiu al programa la següent funcionalitat:

Inicialitzeu un comptador de 8 bits a zero. Els LEDs associats al PORTA mostraran el valor del comptador en tot moment. Cada cop que es detecti un **flanc de pujada**

del polsador associat a RE1, el comptador s'incrementa en una unitat.

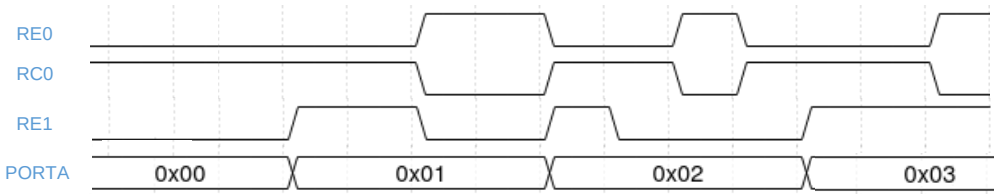


Fig 5. Cronograma amb comptador

5. Afegiu al programa la següent funcionalitat:
   Cada cop que es detecti un **flanc de baixada** del polsador associat a RE2, el comptador es reseteja amb el valor 0.

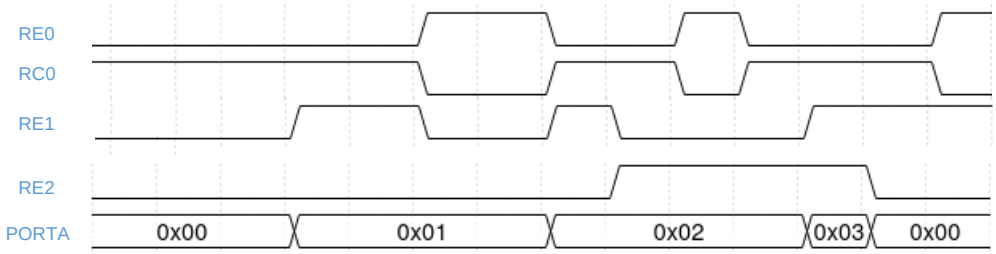


Fig 6. Cronograma amb reset

6. Execució, test i *debugat* dels vostres programes sobre Proteus.
   No us oblideu d'incloure el fitxer 'config.h' abans del vostre codi.

   En el programa, poseu un *software breakpoint* després de la lectura RE0 quan aquest canvia d'estat. Observeu el valor del PORTx, TRISx, LATx, 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 PORTx, TRISx, i LATx 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.

7. Abans de l'inici del laboratori haureu d'entregar al racó el *Full d'entrega* adjunt al final d'aquest document.

Entregueu el projecte PROTEUS (inclou diagrama esquemàtic, codi font implementat i el fitxer executable ".hex") pel Racó, ABANS de la vostra sessió de pràctiques. 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. Pràctica al laboratori

A l'inici de la sessió, l'alumne ha de mostrar el programa de l'apartat 5 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 i PORTE (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.

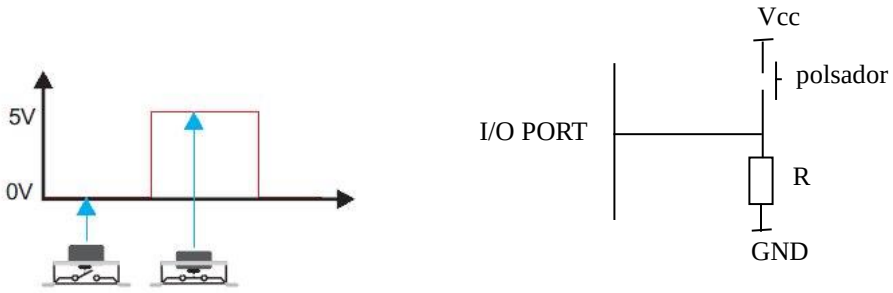


Fig 7. Configuració dels polsadors

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. 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"
```

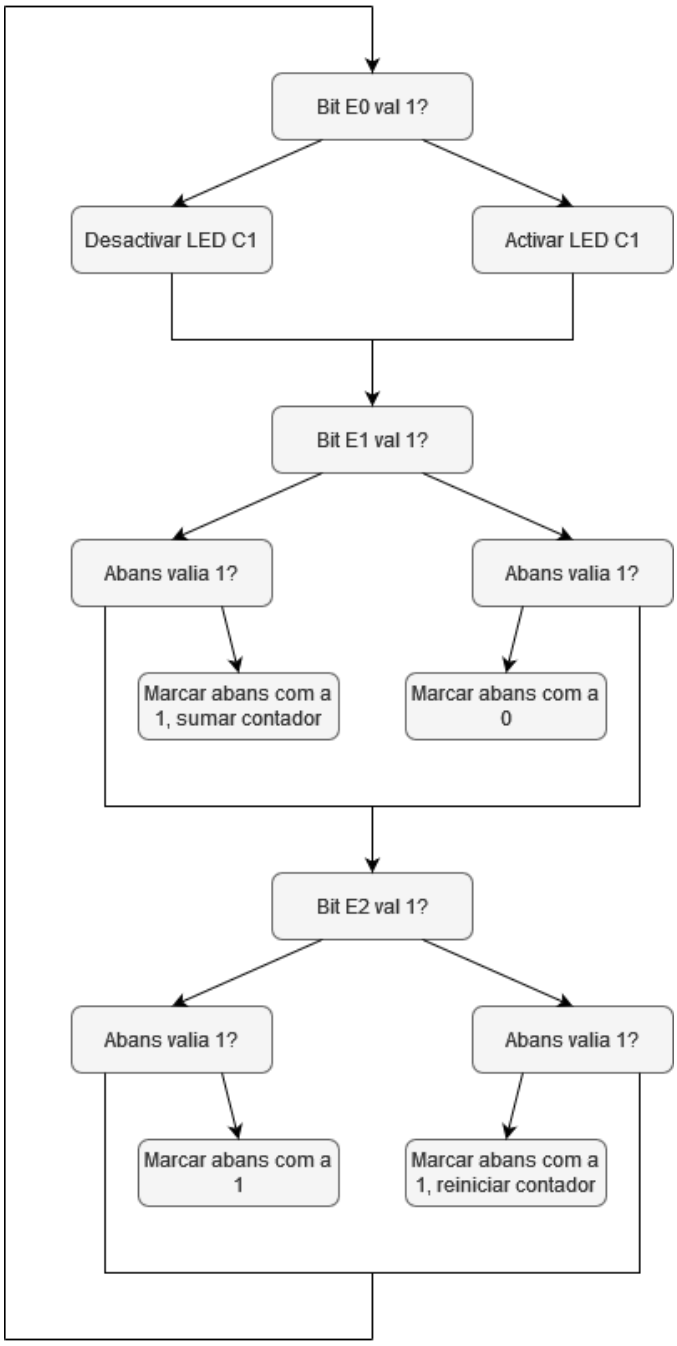
```
void configPIC(){
  ANSELA=...;  // All pins as digital
  ANSELB=...;
  ANSELC=...;
  ANSELD=...;
  ANSELE=...;

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

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

Full d'entrega

1) Dibuixeu els diagrames de flux del programa implementat a l'apartat 5 (comptador amb reset). Podeu mirar al següent enllaç per a més informació:



2) Que conté el fitxer 'config.h'?

Al fitxer podem trobar assignacions en diversos paràmetres per la configuració de la placa.

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

ANSELA= 0x00	PORTA= 0x00	TRISA= 0x00	LATA= 0x00
ANSELB= 0x00	PORTB= 0x00	TRISB= 0xFF	LATB= 0x00
ANSELC= 0x00	PORTC= 0x00	TRISC= 0x00	LATC= 0x01
ANSELD= 0xFF	PORTD= 0x00	TRISD= 0xFF	LATD= 0x00
ANSELE= 0x00	PORTE= 0x00	TRISE= 0x07	LATE= 0x00

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

ANSELA= 0x00	PORTA= 0x00	TRISA= 0x00	LATA= 0x00
ANSELB= 0x00	PORTB= 0x00	TRISB= 0xFF	LATB= 0x00
ANSELC= 0x00	PORTC= 0x00	TRISC= 0x00	LATC= 0x01
ANSELD= 0xFF	PORTD= 0x00	TRISD= 0xFF	LATD= 0x00
ANSELE= 0x00	PORTE= 0x00	TRISE= 0x07	LATE= 0x00