

Enunciat de la pràctica de laboratori

**Lab 2:**

**Entrada-Sortida**

**L2. Entrada-Sortida Assembler (A) + Entrada-Sortida XC8 (B)**

* **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.

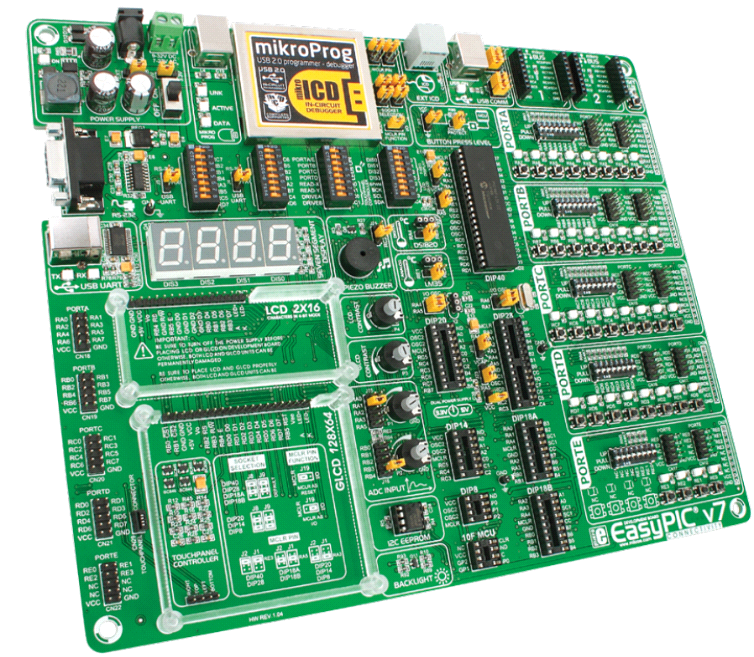


Fig. 1. Plataforma de desenvolupament EasyPIC v7.

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. 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.

A la primera part de la pràctica L2 (A) programarem en assembler amb el MPASM.

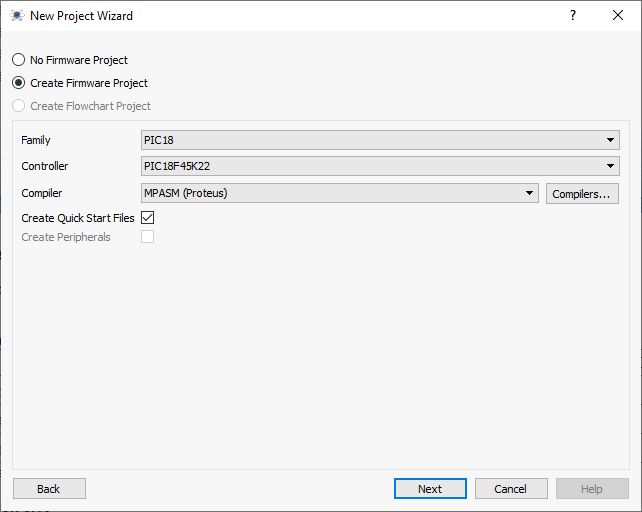


Fig. 2. Captura de pantalla per la generació d’un nou projecte

d’assembler amb el PIC18F45K22 a Proteus.

A la segona part de la pràctica L2 (B) i fins a acabar el quadrimestre, es farà servir el **compilador XC8** (llenguatge C). Aquest compilador no ve per defecte amb Proteus i s’ha d’instal·lar al vostre ordinador. El compilador està disponible a:

<https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers>

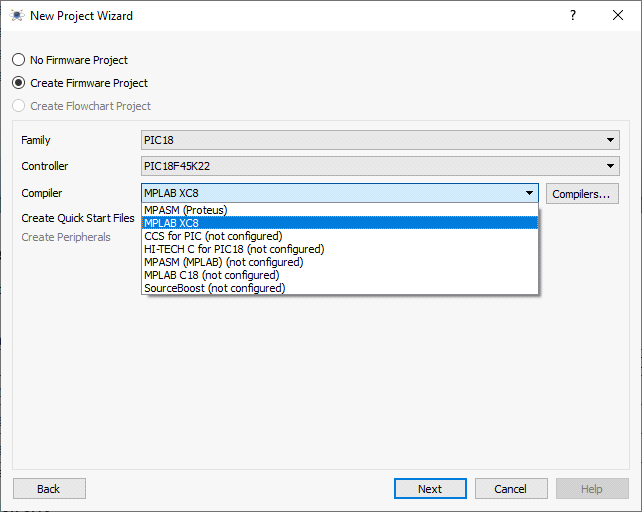


Fig. 3. Captura de pantalla per la generació d’un nou projecte en C

amb el compilador XC8 pel PIC18F45K22 a Proteus.

Amb el software Proteus podreu crear interfícies entre el microcontrolador i el món exterior, programar el codi que executa el microcontrolador i simular-ho tot conjuntament. Un cop el simulador funcioni, podreu provar realment el funcionament del vostre programa a la placa de desenvolupament. Els dos escenaris de simulació i execució a la placa necessiten compilar de forma diferent a Proteus:

* 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* i també podreu debugar el contingut de les variables, registres, etc.
* 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.

En aquesta pràctica L2 es treballaran les entrades i sortides del PIC18F45K22. Per entendre el seu funcionament s’haurà de comprendre tant la part de software a programar con la part de hardware sobre la que funciona el codi programat. Per tant:

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 PORTx com a port digital mitjançant els registres ANSELx, així com la configuració dels ports com a IN o OUT mitjançant els registres TRISx.

2- Cal dissenyar sobre Proteus el circuit basat en l’esquemàtic que trobareu a la documentació lliurada sobre la placa EasyPIC v7 (i a la figura 4). 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 Clear/Reset) estigui a 1, per evitar que hi hagi resets aleatoris si queda a l’aire.

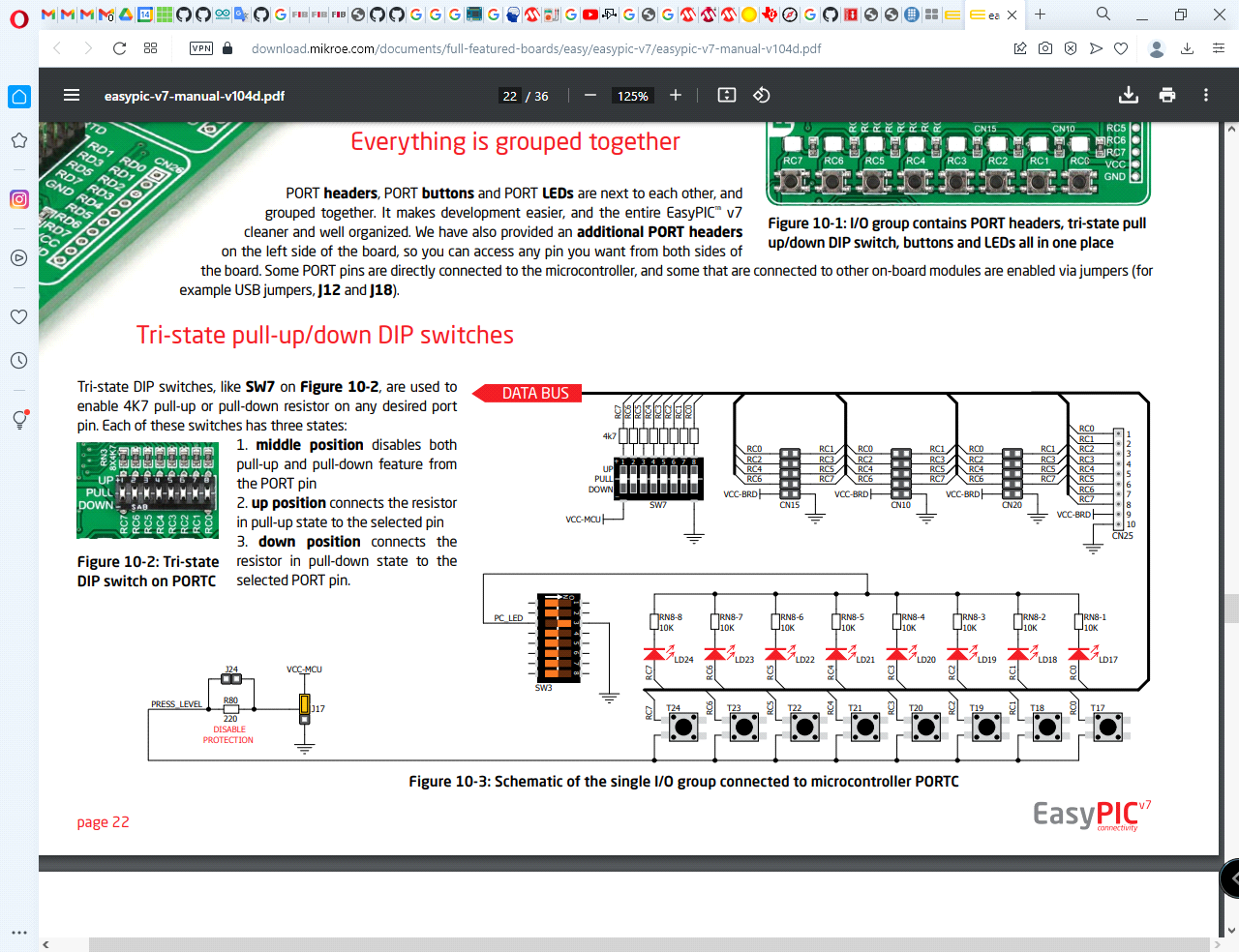


Fig. 4. Esquema dels polsadors i LEDs del PORT C a la placa EasyPIC v7.

* **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.
* control de temps i relació amb la freqüència d’oscil·lació de la EasyPIC Fosc=8MHz
* identificar els elements ‘polsador’ i ‘LED’ i com es connecten al 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.

.

* **Treball Previ Entrada-Sortida Assembler (A)**

Aquesta primera part de la pràctica L2 es farà en assembler.

Fer un programa amb Proteus que realitzi la següent funcionalitat:

-Configuri el pin RB0 com a entrada digital.

-Configuri el pin RB1 com a sortida digital.

-Llegeixi l’estat de l’entrada RB0. Si aquesta és un 0 llavors el codi haurà de posar un 1 a la sortida RB1. Si l’entrada RB0 és un 1 llavors el valor de la sortida RB1 haurà de ser un 0.

-Configuri el pin RB2 com a sortida digital.

-Canviï l’estat del pin RB2 entre 0 i 1 cada 40 cicles d’instrucció aproximadament.

Fig. 6. Cronograma del programa L2A a realitzar en assembler.

|  |
| --- |
| // A proposal for your code... Just to inspire you  // Hexadecimal numbers start with 0x  // Binary numbers start with B'  // Decimal numbers can be written without prefix  #include p18f45k22.inc  config FOSC = HSHP    ;========================================================  ; VARIABLES  ;========================================================  var1 equ 0x20  ;========================================================  ; RESET and INTERRUPT VECTORS  ;========================================================  RST code 0x0  goto Start  ;=================  =======================================  ; CODE SEGMENT  ;========================================================  PGM code  Start  movlw B'00000001'; W=0  ; Configurar PORTs  ...  Loop  ; Llegir RB0 i actualitzar RB1  ...  ; Posar el valor a RB2 i esperar uns 40 cicles  ...  goto Loop  delay\_40cicles  ... |

Fig. 7. Plantilla de codi que podeu fer servir.

**Entregueu pel Racó, ABANS de la vostra sessió de pràctiques, el projecte PROTEUS (versió actual als laboratoris, v8.13 SP0).**

* **Rúbrica treball Previ Entrada Sortida Assembler (A)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Iniciat  (0-2.5 punts) | En desenvolupament  (2.5-5.0 punts) | Aconseguit  (5.0-7.5 punts) | Exemplar  (7.5-10 punts) |
| Esquemàtic circuit entrada  (2 punts): | No funciona | Funciona en alguns casos. Hi han errors o falten connexions | Funciona però la selecció dels components no és apropiada | Funciona perfectament |
| Esquemàtic circuit sortida  (2 punts): | No funciona | Funciona en alguns casos. Hi han errors o falten connexions | Funciona però la selecció dels components no és apropiada | Funciona perfectament |
| Codi configuració  (2 punts): | No funciona | Mala configuració de ANSELx i TRISx | Mala configuració de ANSELx o TRISx | Ben configurat |
| Codi RB0RB1  (2 punts): | No funciona | No té en compte l’estat del bit sinó del PORT sencer | No funciona en alguns casos | Funciona perfectament |
| Codi delay\_40cicles  (2 punts): | No funciona | El codi està molt mal estructurat | Algunes parts del codi són difícils de justificar | Codi net, ben estructurat i fàcil d’entendre |

* **Treball Previ Entrada-Sortida XC8 (B)**

Aquesta segona part de la pràctica L2 es farà en llenguatge C fent servir el compilador XC8 que podeu trobar a la pàgina de microchip.

En aquesta part haureu de fer un programa en llenguatge C en el que cada cop que es premi el polsador associat al bit 0 del PORTB (RB0) s’apagui el LED associat al bit 1 del PORTB (RB1) i quan es deixi anar el polsador s’encengui el LED (veure figura 4). Es tracta de la mateixa funcionalitat que es va fer en assembler en la pràctica L2(A). 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. 7. Cronograma simple del programa en C.

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 RB2, el comptador s’incrementa en una unitat.

Fig 8. Cronograma amb comptador.

Afegiu al programa la següent funcionalitat:

Cada cop que es detecti un **flanc de baixada** del polsador associat a RB3, el comptador es reseteja amb el valor 0.

Fig 9. Cronograma amb reset.

A continuació, poseu un *software breakpoint* després de la detecció del flanc a RB2. 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 *ProbesVoltageReal Time BreakpointDigitalTrigger 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 Qüestionari que trobareu al final d’aquest document i també el projecte PROTEUS amb els circuits i els codis que es demanen. Per a garantir compatibilitats de versions, lliureu sempre el vostre treball salvat en una versió compatible de PROTEUS (versió actual als laboratoris, v8.13 SP0).**

|  |
| --- |
| // 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  }  } |

Fig. 10. Plantilla de codi que podeu fer servir.

Recordeu que heu d’afegir l’arxiu “config.h” que trobareu a Atenea perquè el programa funcioni correctament. Per fer-ho teniu diferents opcions, entre elles anar a la finestra de Proteus anomenada Source Code, a la part esquerra trobareu l’estructura d’arxius del projecte, clickeu amb el botó dret i afegiu un arxiu nou amb Add New File, anomeu l’arxiu com a config.h, obriu-lo i copieu el codi que hi ha a l‘arxiu d’Atenea.

També heu de configurar la simulació del PIC18F45K22 a Proteus perquè funcioni a 8MHz de tal manera que la simulació i la placa EasyPicv7 funcionin a la mateixa velocitat de rellotge. Per fer-ho heu d’anar a la pestanya de l’esquemàtic i fer doble click sobre el microcontrolador. S’obrirà una finestra per editar el component i s’haurà de modificar el valor de Processor Clock Frequency a 8MHz.

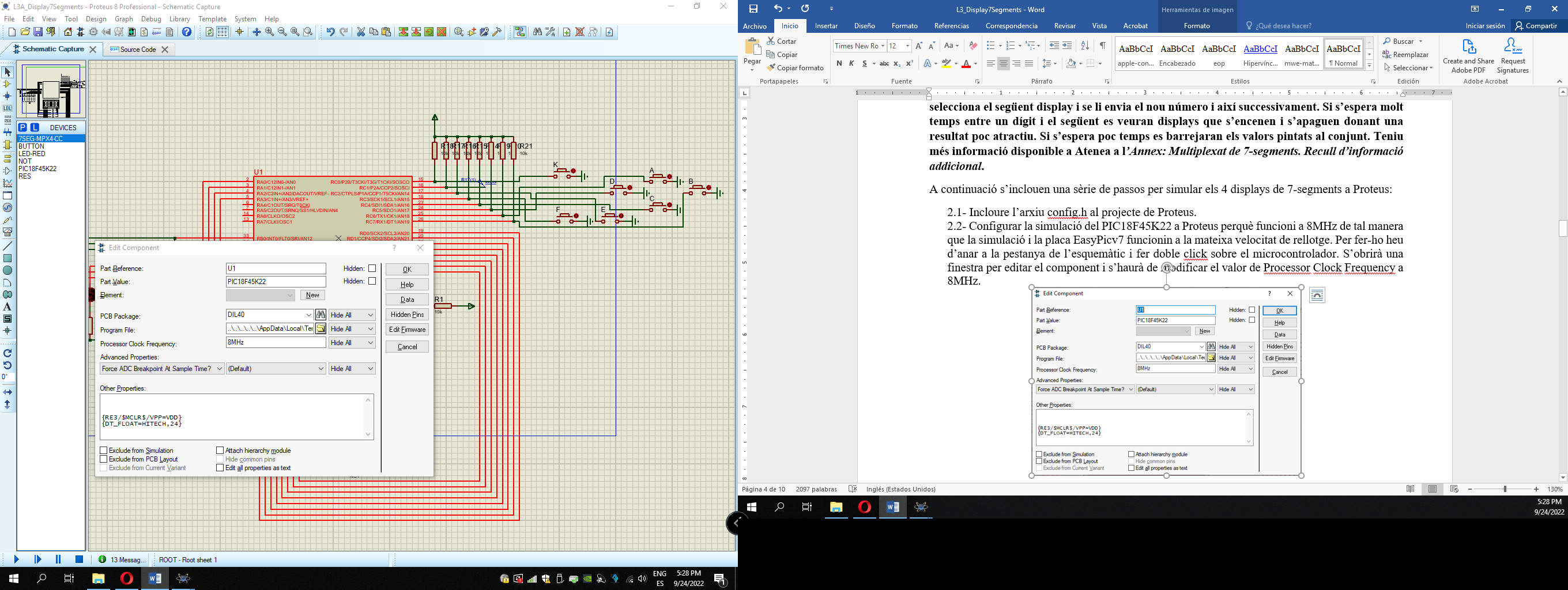


Fig. 11. Captura de pantalla pel canvi de la velocitat del clock.

* **Rúbrica treball Previ Entrada Sortida XC8 (B)**

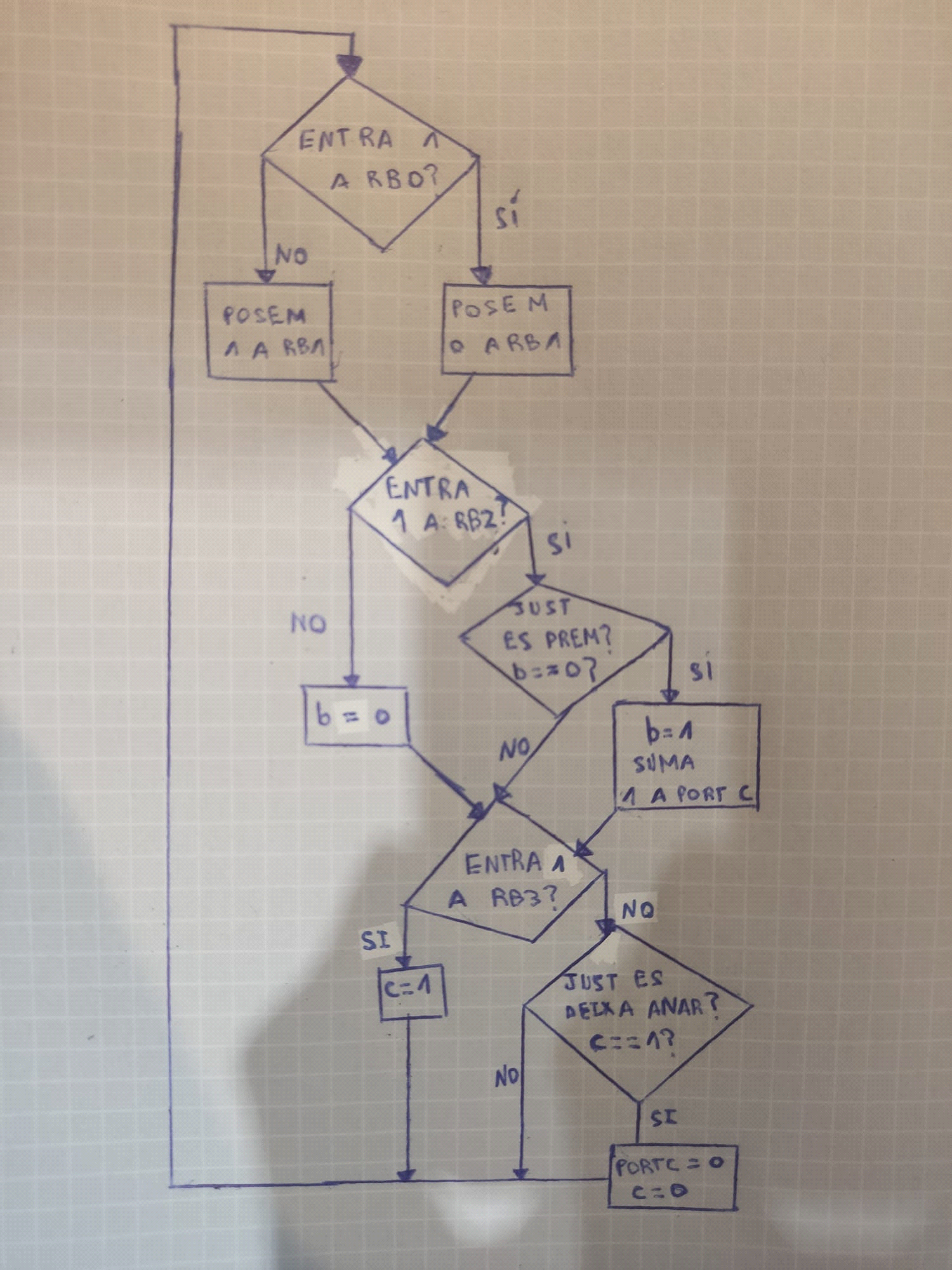
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Iniciat  (0-2.5 punts) | En desenvolupament  (2.5-5.0 punts) | Aconseguit  (5.0-7.5 punts) | Exemplar  (7.5-10 punts) |
| Proteus circuit 1  negador  (1 punt): | Hw i Sw  tenen errors | Hw funciona però  Sw falla | Sw funciona però  Hw falla | Sw i Hw funcionen |
| Proteus circuit 2  comptador  (2 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  (2 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. |
| Diagrama flux  circuits  (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 tots els circuits |
| Proteus sw breakpoint  ( 1 punt): | Pregunta 2 qüestionari  incorrecta i breakpoint no funcional a proteus | Pregunta 2 qüestionari  correcta i breakpoint no funcional a proteus | Pregunta 2 qüestionari  incorrecta i breakpoint funcional a proteus | Pregunta 2 qüestionari  correcta i breakpoint funcional a proteus |
| Proteus hw breakpoint  (1 punt): | Pregunta 3 qüestionari  incorrecta i breakpoint no funcional a proteus | Pregunta 3 qüestionari  correcta i breakpoint no funcional a proteus | Pregunta 3 qüestionari  incorrecta i breakpoint funcional a proteus | Pregunta 3 qüestionari  correcta i breakpoint funcional a proteus |
| Explicació ANSEL, TRIS, LAT, PORT  (1 punt): | Preguntes 5, 6, 7 i 8 del qüestionari  incorrectes | Dos o més respostes incorrectes a les preguntes 5, 6, 7 ó 8 del qüestionari | Una resposta incorrecta a les preguntes 5, 6, 7 ó 8 del qüestionari | Resposta correcte a les preguntes 5, 6, 7 i 8 del qüestionari |

**QÜESTIONARI–L2 Entrada-Sortida (B)**

***(s’ha d’entregar en format electrònic com a treball previ de la L2 (B) )***

Nom i Cognoms: Àlex Lafuente, Gabriel Grup LAB: 34

1) Dibuixeu els diagrames de flux del programa FINAL implementat a la pràctica L2(B). 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= \0 TRISA= 0xFF LATA= 0 PORTA= 0

ANSELB= \0 TRISB= 0xFD LATB= 6 PORTB= 0x06

ANSELC= \0 TRISC= 0x00 LATC= 0 PORTC= 0

ANSELD= \0 TRISD= 0xFf LATD= 0 PORTD= 0

ANSELE= \0 TRISE= 0x87 LATE= 0 PORTE= 0

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

ANSELA= \0 TRISA= 0xFF LATA= 0 PORTA= \0

ANSELB= \0 TRISB= 0xFD LATB= 6 PORTB= 0x06

ANSELC= \0 TRISC= 0 LATC= 1 PORTC= \0

ANSELD= \0 TRISD= 0xFF LATD= 0 PORTD= \0

ANSELE= \0 TRISE= 0x87 LATE= 0 PORTE= \0

4) Per a què serveix el registre ANSEL?

Serveix per a escollir si els pins del micro s'interpreten com a digitals, o analògics.

5) Per a què serveix el registre TRIS?

Serveix per a escollir si els pins del micro s'interpreten com a input o output.

6) Per a què serveix el registre LAT?

Serveix per a posar un valor de sortida als pins del micro.

7) Per a què serveix el registre PORT?

Serveix per a llegir els valors d'input als pins del micro.