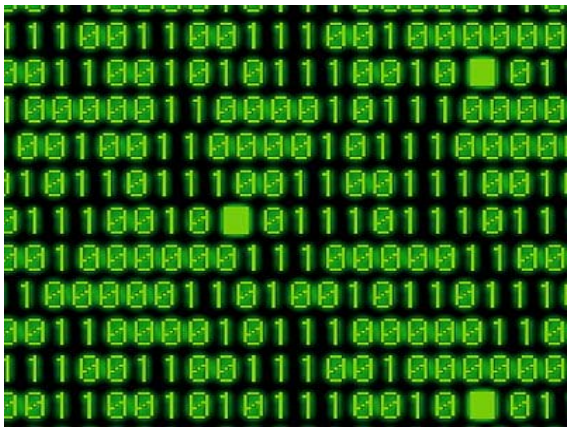


CISE IV

Circuits i Sistemes
Electrònics IV

Manual de Pràctiques

Manel Domínguez, Antoni Mas

Historial del document

Data	Ver.	Responsable	Descripció
11/04/2008	1.0	M. Domínguez, A. Mas	Versió inicial
23/04/2008	1.01	M. Domínguez	Correccions menors a apartats 4, 5 i 6.

Contingut

1. Objectiu de les Pràctiques

Introducció

L'ús de sistemes basats en microprocessador/microcontrolador requereix coneixements de programació. Els objectius de les pràctiques són:

a) introduir un entorn de programació professional, l'Eclipse,
b) practicar aspectes de programació elemental de sistemes basats en microcontrolador, i
c) analitzar cicles de bus d'un microcontrolador amb un analitzador lògic.

Aspectes clau són la gestió dels tipus d'interrupcions i la seva prioritat, el mapa de memòria d'un sistema microprocessador, el control dels perifèrics bàsics, els cicles de bus i l'adaptació de la velocitat entre el processador i els dispositius connectats en els seus busos (principalment memòries) mitjançant el control dels estats d'espera i altres recursos semblants.

Les sessions estan enfocades a objectius molt concrets. Aquest manual conté la majoria de les dades imprescindibles per comprendre el que cal fer i tenir a l'abast la informació imprescindible per realitzar el treball. En algunes ocasions serà aconsellable ampliar la informació utilitzant els fitxers addicionals com són el Manual d'usuari o el datasheet del LPC2292.

Documentació a lliurar

En acabar cada sessió, i en el termini que el professor de pràctiques estipuli, el grup de pràctiques haurà de proporcionar al professor el resultat del seu treball: ja sigui uns fitxers comprimits en format .ZIP o .RAR, o documentació escrita.

La memòria final ha de respondre a totes les qüestions plantejades a les sessions, exposar els resultats i les incidències del treball realitzat i explicar qualsevol aspecte que,a judici dels alumnes, calgui ressaltar per a la correcta avaluació del treball.

El projecte de SW o programa realitzat (més d'un si és el cas) correspondrà a les especificacions plantejades. S'ha de poder compilar sense errors i verificar el seu funcionament sobre el material de practiques. En cas contrari, no es considerarà vàlid.

Tot els treballs han de ser originals i haver estat realitzats al llarg de la sessió de pràctiques.

Avaluació

L'avaluació es farà atenent als següents criteris:

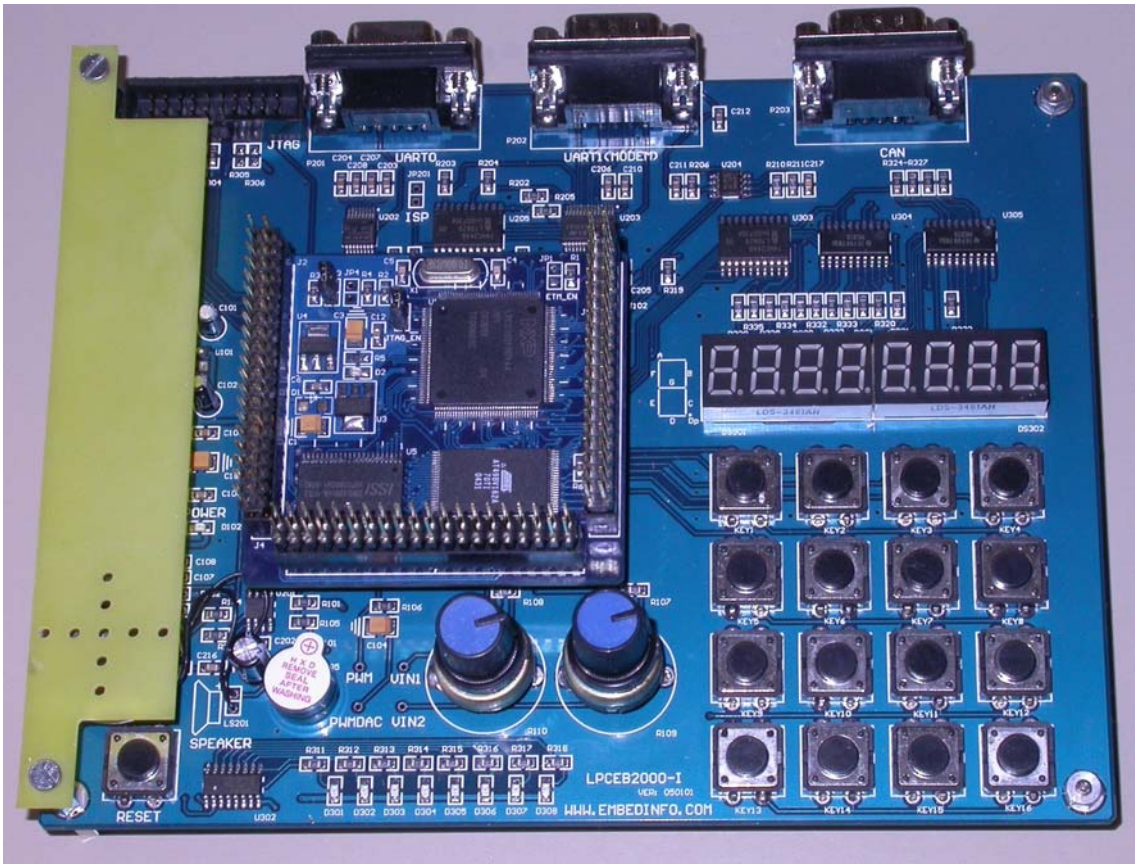
a) (30%) Compliment dels objectius específics demanats al manual de pràctiques.
b) (30%) Qualitat de la programació realitzada: fer servir bones tècniques de programació (ús mínim de variables globals, **claredat i eficiència** del programa, etc.)
c) (15%) Innovació, originalitat, i/o haver fet algun(s) apartat(s) opcional(s).
d) (15%) Memòria final.
e) (10%) Control de pràctiques.

Hi ha 4 apartats obligatoris: els 3,4,5 i 6. Es recomana fer algun dels apartats opcionals.

Es pot fer cadascun dels apartats obligatoris en una sessió (hi ha 4 sessions en total).

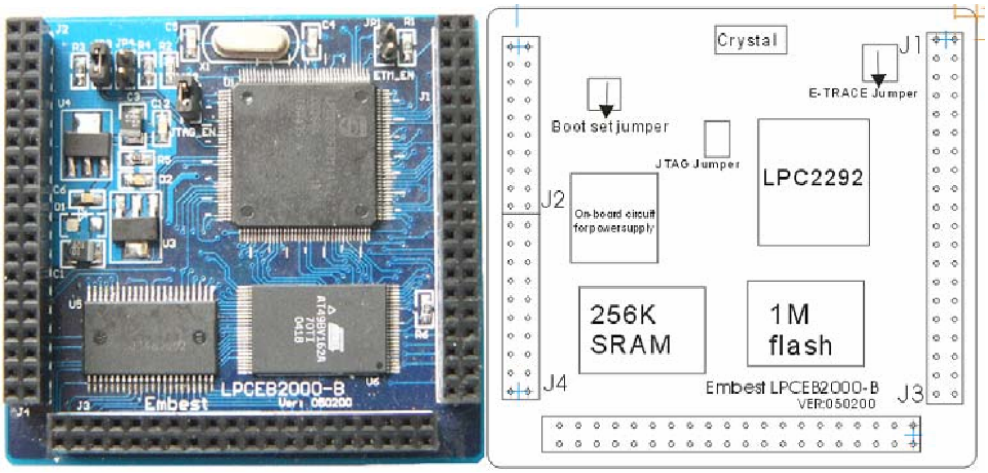
2. Breu descripció de la placa de Pràctiques

La placa d'evaluació que s'utilitzarà als treballs de laboratori és la LPCEB2000-I, de la casa Embest Info&Tech Co., LTD (http://www.armkits.com/), implementada per al desenvolupament dels microcontroladors LPC2000 de Philips que es basen en el núcli de CPU (CPU-*core*) ARM7TDMI-S de 16/32 bits. L'apecte físic i el diagrama d'aquesta placa "mare" es dóna a les següents 2 figures.



El hardware que presenta aquesta placa és el següent:

- Regulador de tensió de +5V per a una entrada d'alimentació de +9V.
- 2 ports sèrie (UARTs de comunicació sèrie asíncrona).
- Port CAN (*Controller Area Network*)
- Polsador de *Reset* i teclat 4x4.
- 8 indicadors LED.
- 2 canals per a senyal analògica (amb convertidors A/D).
- 2 canals amb sortida PWM (Pulse-Width Modulation); un a un brunzidor, l'altre a un convertidor D/A amb PWM.
- Bus serie I2C (*Inter-Integrated Circuits*)
- Visualitzador LED amb 8dígits de 8 segments.
- Sortida analògica a un altaveu.
- Port JTAG (*Joint Test Action Group*) estàndard de 20 *pins* a través del qual executarem i depurarem els codis dels nostres programes.
- 2 sòcols connectors per a expansió de la placa del mòdul microcontrolador.
- Placa "filla" del mòdul microcontrolador LPCEB2000-B, l'aspecte físic i l'enumeració de components de la qual es presenten a continuació:



- Microcontrolador LPC2292 de Philips basat en el núcli de CPU (CPU-*core*) ARM7TDMI-S de 16/32 bits.
- Cristall de quars de 10 MHz.
- Memòria *Flash* externa AT49BV162A (1M x 16 bits).
- Memòria SRAM externa IS61LV25616 (256K x 16 bits).
- Reguladors de tensió de 3,3V i 1,8V per a una entrada de alimentació de +5V.
- LED indicador de alimentació.
- 3 connexions tipus *jumper* per a JTAG (activat/desactivat), TRACE (activat/desactivat) i BOOT (des de la memòria *Flash* externa/interna).
- 4 sòcols connectors J1 – J4 (2 de 10x2 *pins* i 2 de 20x2 *pins*).

La següent figura presenta el diagrama de blocs intern del microntrolador LPC2292.

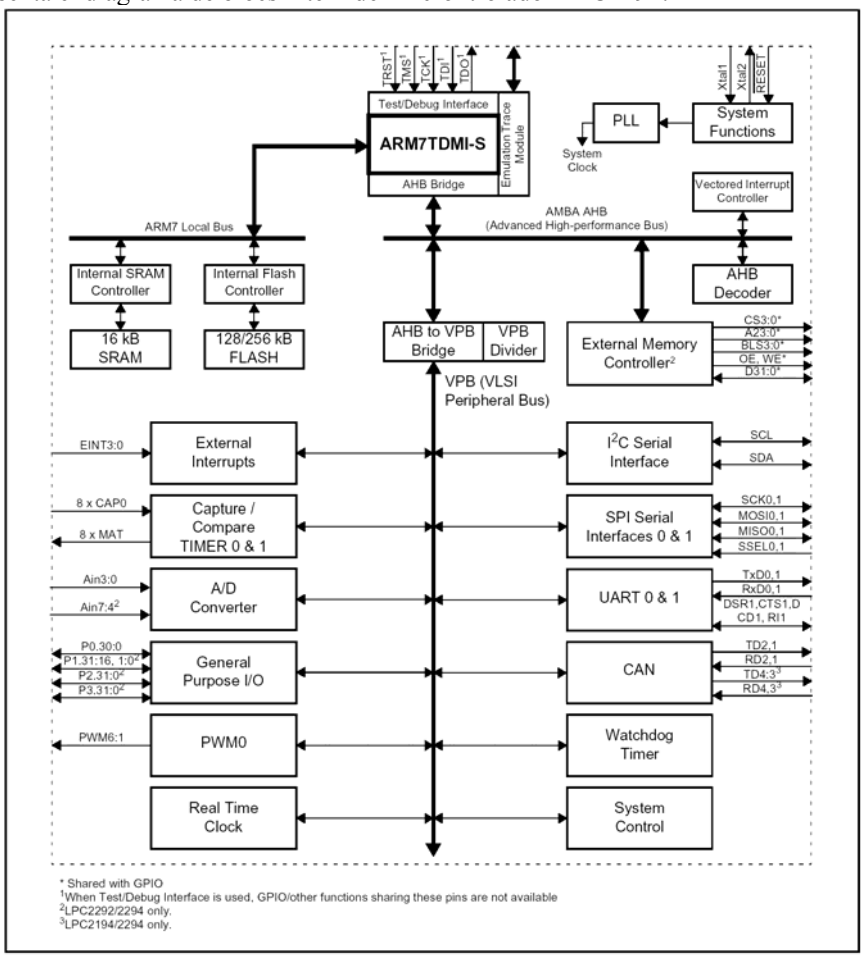


Figure 1: LPC2119/2129/2194/2292/2294 Block Diagram

3. Depuració i programació amb l'entorn ECLIPSE

Objetius concrets:

- **Entendre el funcionament bàsic de l'entorn Eclipse, aplicat a sistemes 'embedded',**
- **Compilar i executar un programa de prova a la placa.**

Compiladors creuats

Volem compilar i depurar un programa a la placa LPCEB2000. El microcontrolador que es fa servir és el LPC2292 de Philips, que es basa en el ARM7TDMI. Com que totes les eines que farem servir per compilar i depurar es troben al PC, que fa servir un microprocessador Intel Pentium 4, vol dir que hem de fer servir un compilador creuat:

Definició: un compilador creuat és un programa que s'executa a un microprocessador per tal de compilar programes que s'executaran a un altre microprocessador.

En el nostre cas el compilador és el 'arm-elf-gcc'. Podeu provar a executar a una finestra de DOS:

'arm-elf-gcc -v' i 'arm-elf-gcc --version'

Aquest compilador agafarà el nostre codi, en C, i el compilarà a un PC per tal que es pugui executar a la nostra placa, que té un microprocessador diferent. Obviament, això implica que els executables generats per aquest compilador NO es poden executar al PC.

Depuració de programes amb JTAG

El acrònim JTAG vol dir 'Joint Test Action Group', i és la norma IEEE 1149.1. S'utilitza, entre altres coses, per depurar programes a microprocessador com el ARM7TDMI. Amb aquest standard, el hardware i el software necessaris podem parar l'execució del microprocessador, carregar un programa, executar pas a pas, fer un volcat de la memòria, etc.

El connexionat és el següent:

a) Al PC es connecta un cable USB que va al JTAG Amontec Tiny (http://www.amontec.com/jtagkey-tiny.shtml, o busqueu 'Amontec tiny' a Google). Aquest dispositiu fa de pont entre el PC i el interface JTAG del microprocessador.

b) El JTAG Amontec tiny, està connectat a la placa, tal i com us indicarà el professor de pràctiques.

Programes instal·lats

L'entorn de programació que farem servir és diu Eclipse (http://www.eclipse.org). És un entorn creat per IBM a l'any 2001. A l'any 2004 IBM va fer una donació per tal que aquest projecte esdevingués software obert. Permet compilar i depurar programes a un nombre elevat de plataformes, i com que és software lliure es manté per la comunitat i té constants millores i noves versions.

En el nostre cas els programes instal·lats són:

a) Driver d'Amontec lliure, basat en els drivers FTDI D2XX, per poder-se connectar mitjançant el bus USB a l'Amontec JTAG tiny.

b) OpenOCD: Software intermig, entre Eclipse i els drivers, per tal de poder controlar el microprocessador amb l'interface JTAG. Aquest programa va ser creat per Dominic Rath (http://OpenOCD.berlios.de/web/) i permet manegar, entre d'altres al microcontrolador LPC2292 de les pràctiques.

La versió instal·lada als ordinadors de pràctiques prové de la compilació YAGARTO: 'Yet another GNU ARM toolchain' que podeu trobar a http://www.yagarto.de.

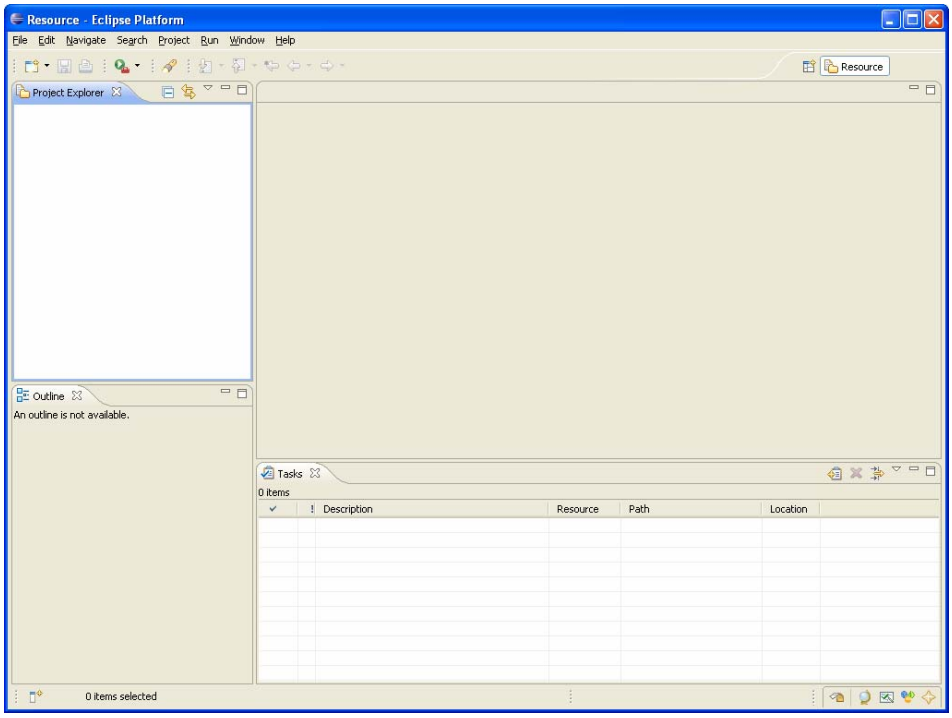
c) Entorn Eclipse: Entorn de programació global, basat en Java.

Execució entorn Eclipse

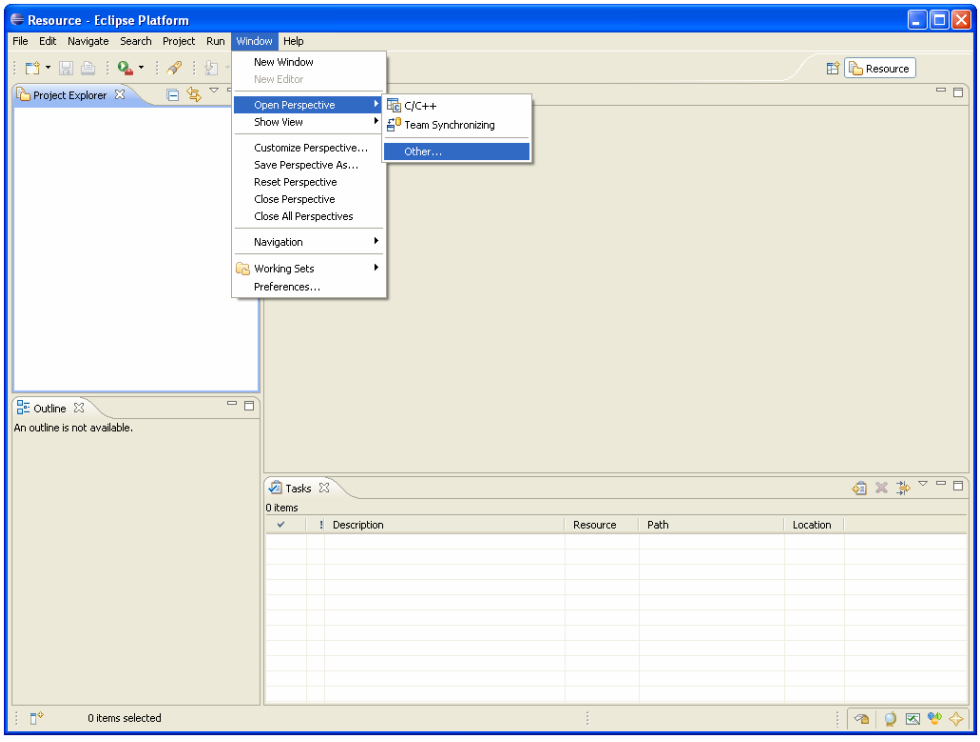
Obriu el programa Eclipse. A la següent finestra indiquem el que serà el nostre directori de treball (a la nostra zona, no local al PC). Si activem l'opció de que no torni a fer la pregunta una altra vegada, usarem sempre el mateix directori com a directori de treball (opció recomanada).

A continuació s'obrirà la finestra principal de l'Eclipse (la podeu veure a la figura). Hem d'obrir el 'Workbench'.

La finestra prendrà la següent forma:

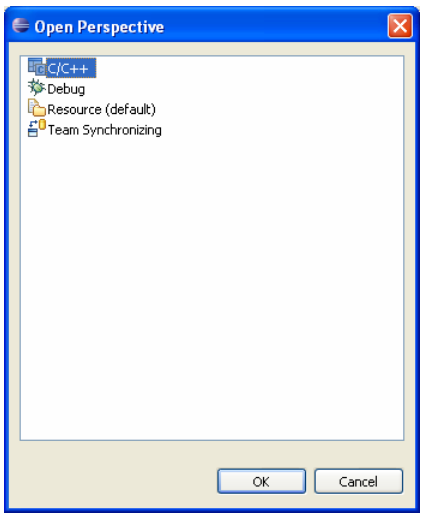


La perspectiva que ens ofereix ara l'Eclipse és la de 'recursos'. Ens interessa obrir la perspectiva de C/C++.

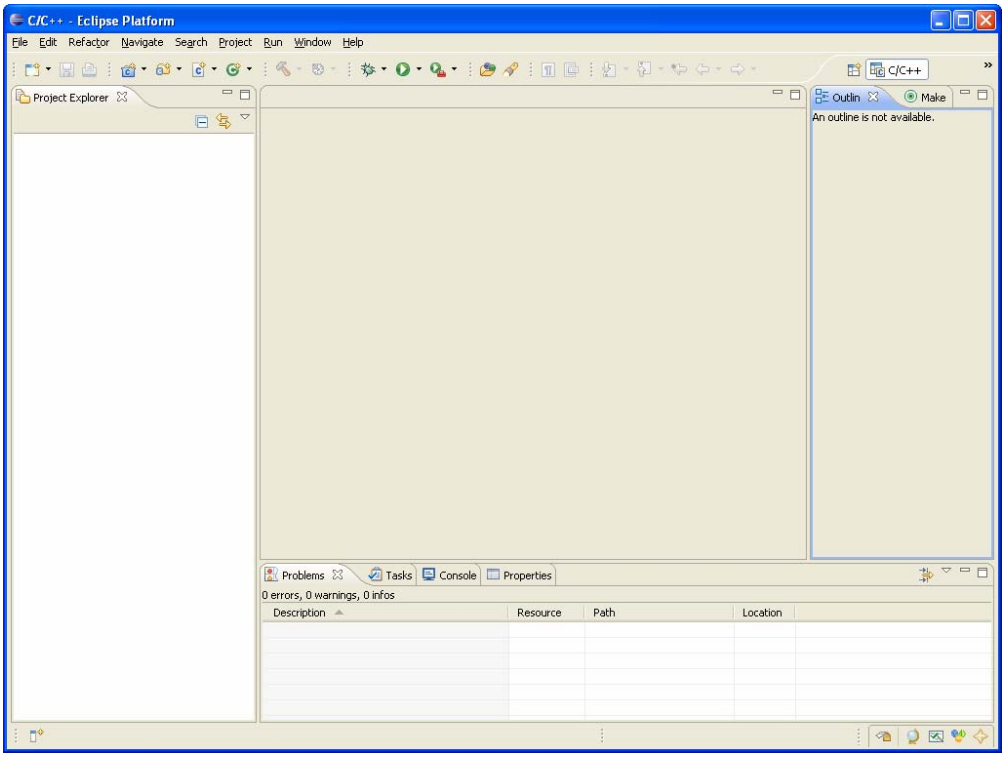


A la nova perspectiva podrem tenir projectes de programes en C, editar els fitxers i compilar-los.

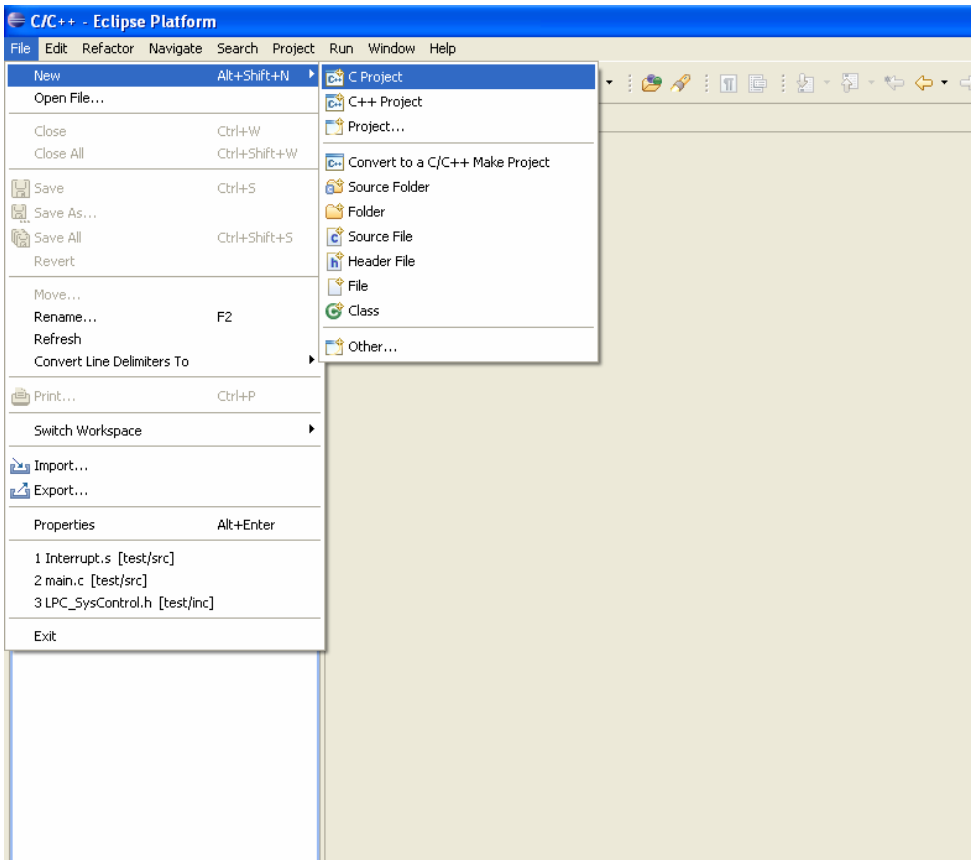
Seleccioneu C/C++:



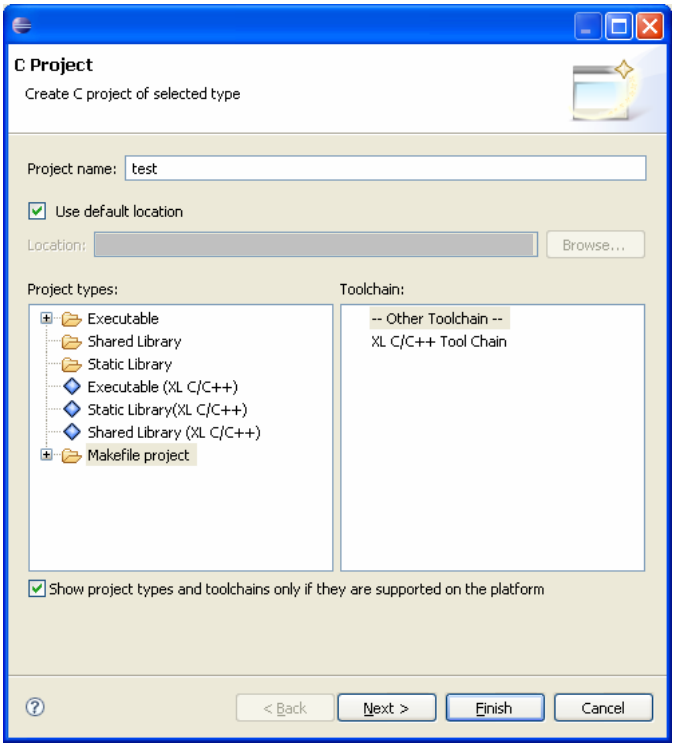
El resultat és una finestra com aquesta:



Com podem veure l'entorn està buit. No hi tenim cap projecte ni cap programa. Per tant, el que farem és crear el nostre primer projecte d'Eclipse. Aneu a File->New Project->C project, tal i com podeu veure a la següent figura:

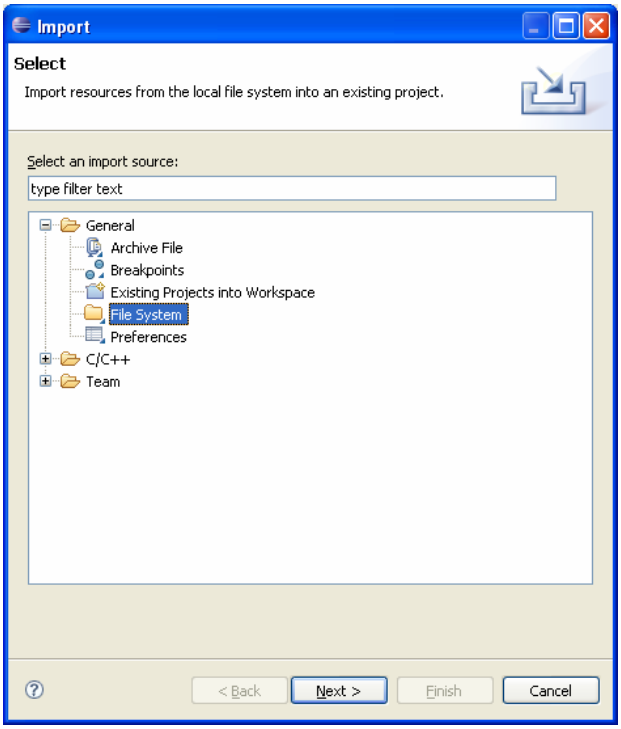


I seleccioneu que voleu un projecte C, basat en makefile i fent servir '--Other Toolchain --'. Li podeu donar el nom de 'test' a aquest nou projecte.

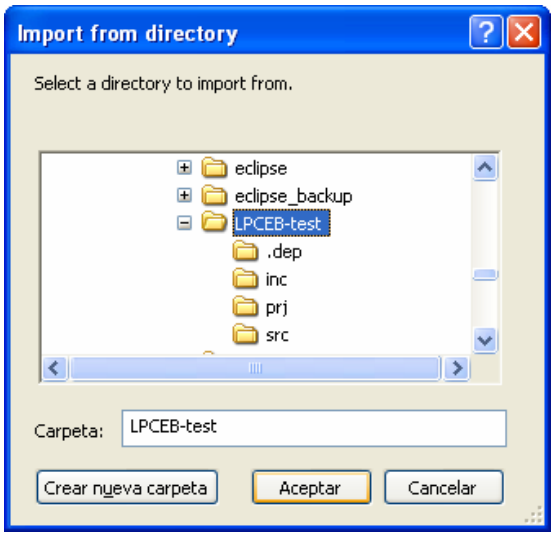


El resultat és que ara tenim un projecte nou, que està buit: no té cap fitxer C, ni res. El que farem serà agafar un projecte senzill, ja adequat per a la nostra placa, i l'importarem. Aquest projecte es troba al fitxer 'LPCEB-test.rar' que trobareu al directori de l'assignatura. L'heu de copiar a la vostra zona i descomprimir-lo.

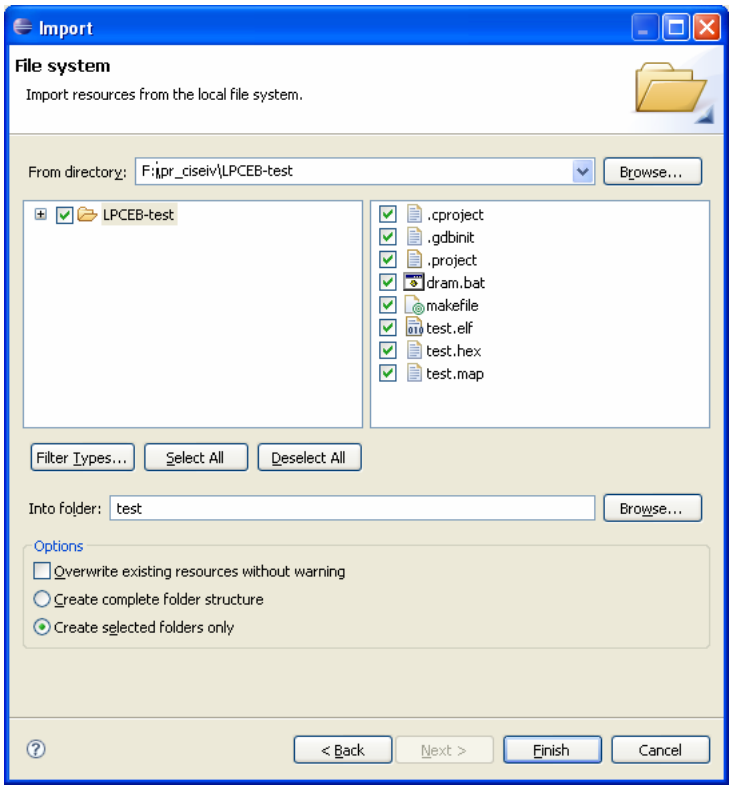
Un cop fet això, l'importem fent: File->Import. S'obrirà la següent finestra, a on indiquem que volem importar fitxers d'un sistema de fitxers.



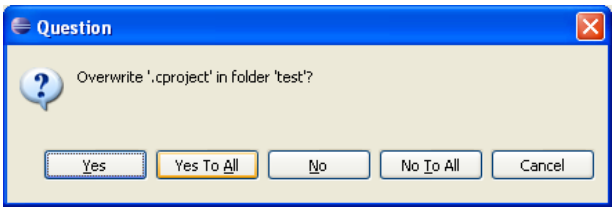
Farem servir com a projecte inicial el que es troba al fitxer LPCEB-test.rar, que trobareu a l'Atenea. Descomprimim el fitxer i a continuació seleccionem el directori a on s'hagin descomprimit els nostres fitxers:



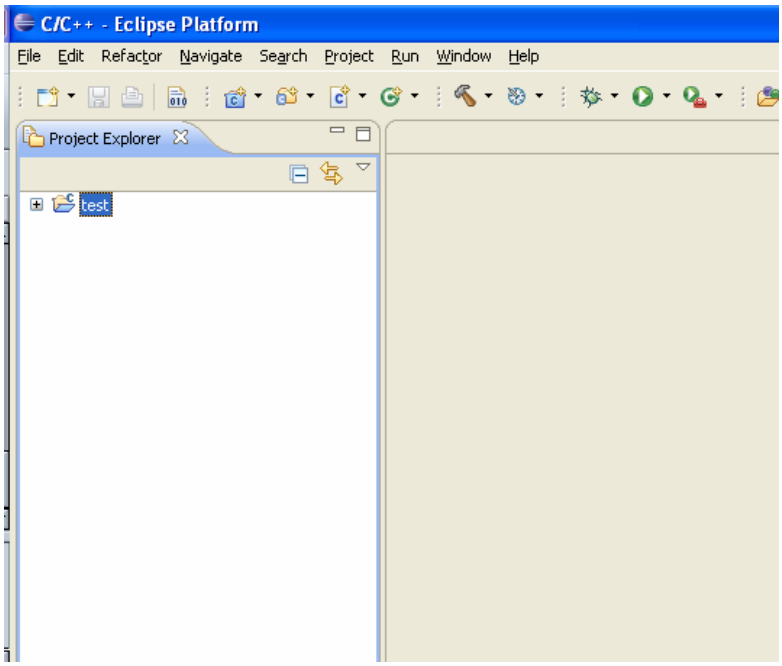
i importarem tots els fitxers que ens donen per triar. Hem d'especificar la carpeta de destí, i posem el nom del projecte que acabem de crear ('test'). Amb aquesta operació, el programa copia els fitxers, amb la seva estructura de directoris, al nostre directori de treball, i serà amb aquests amb els que treballarem (no amb els originals allà on s'hagin descomprimit).



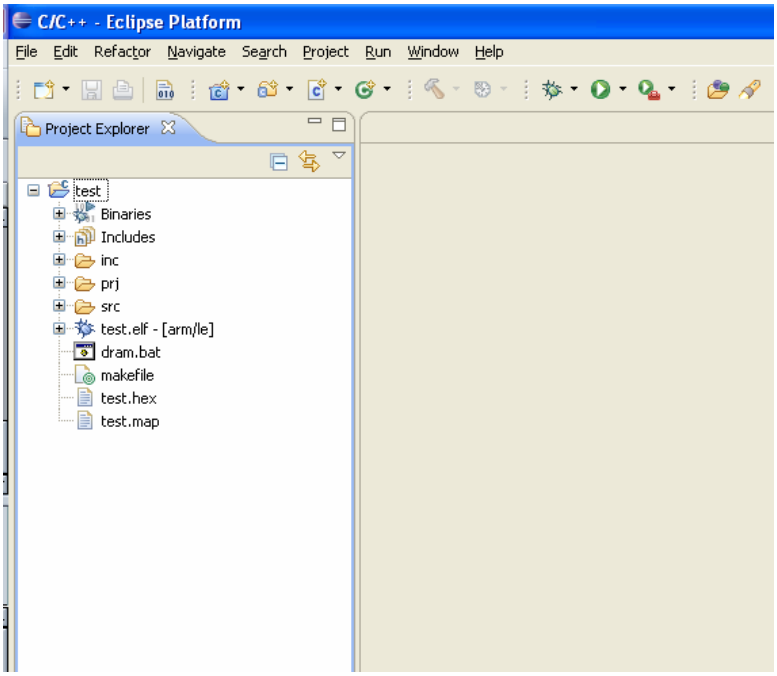
Si ens pregunten sobre si sobreescriure cap fitxer, diguem que si: 'Yes To All'.



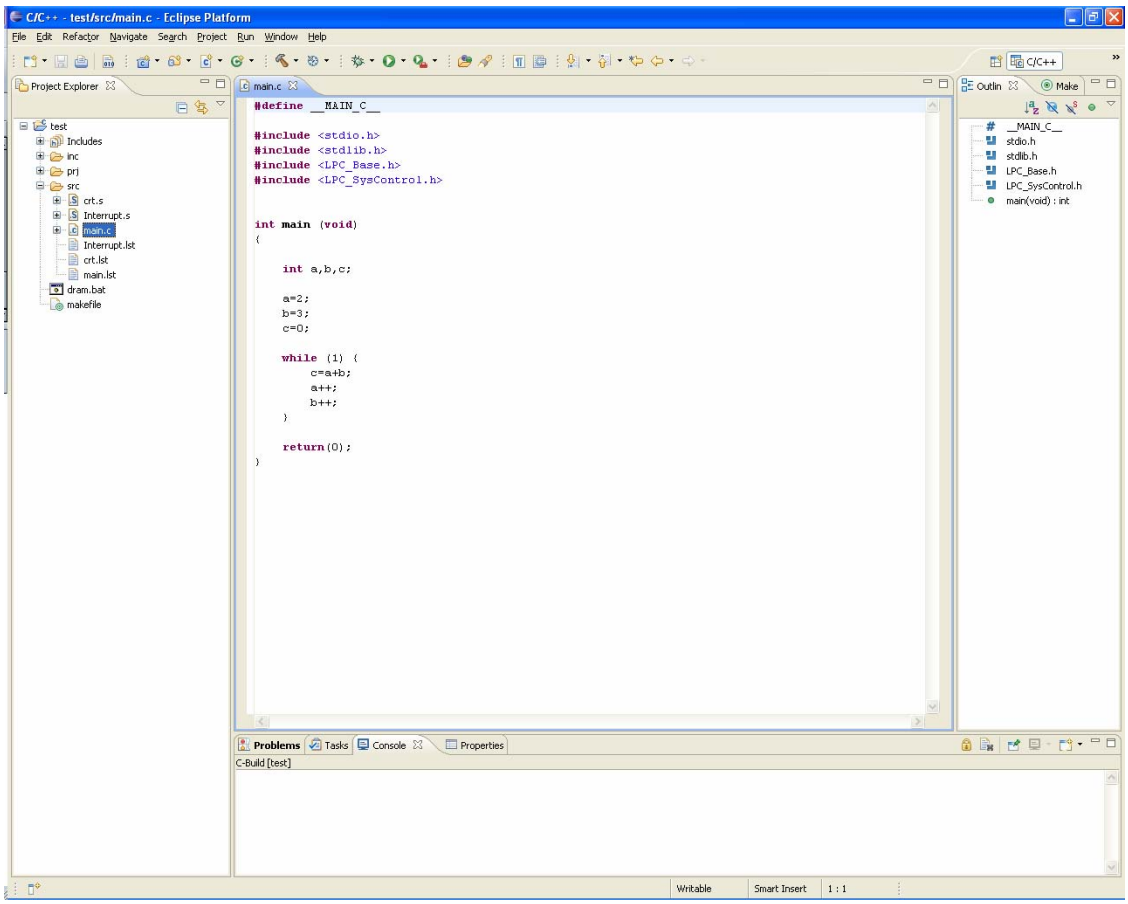
El resultat és:



Podem ‘clicar’ sobre el ‘+’ de test per veure tots els fitxers interns del projecte:

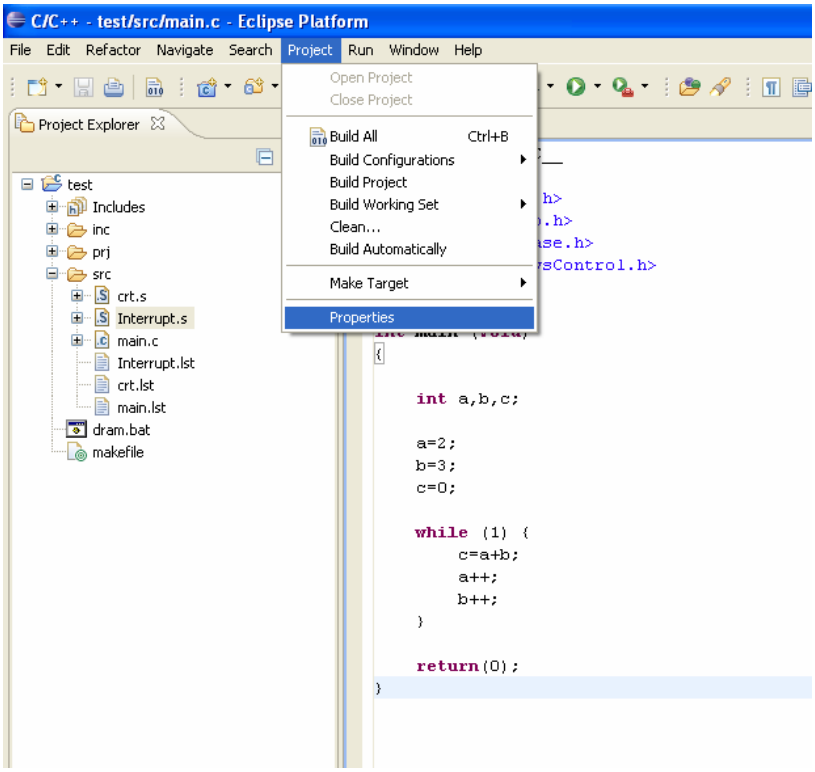


i obrir el fixer ‘main.c’, fent doble clic. Com podeu veure és un programa extremadament senzill.

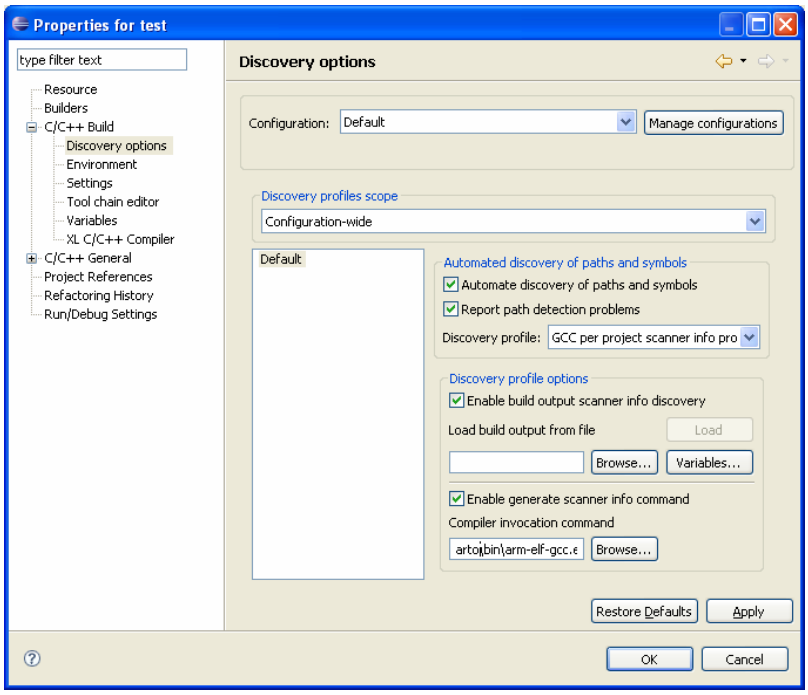


Compilació del projecte 'test'

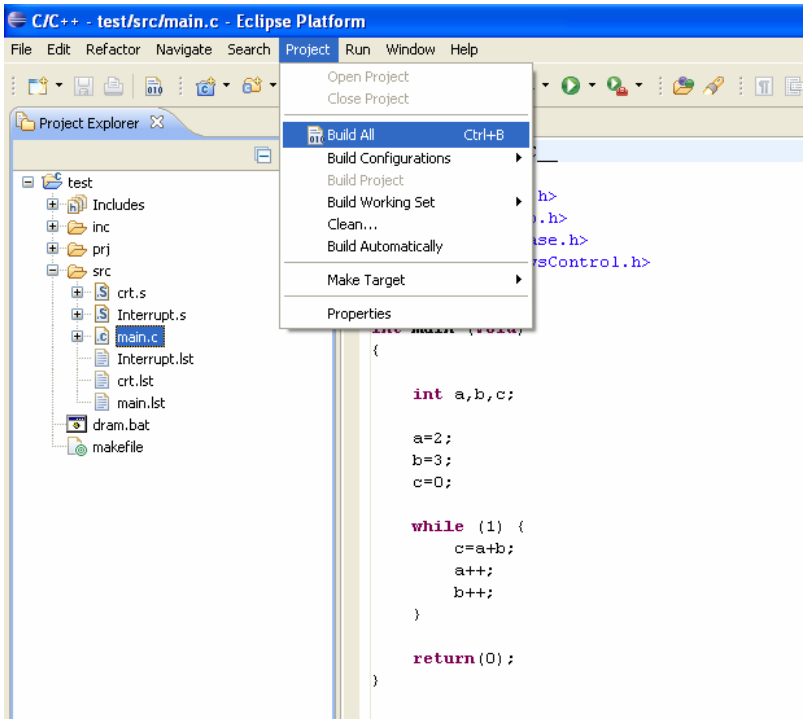
Ara volem compilar, amb el compilador creuat que hem dit a la introducció, *arm-elf-gcc*, el nostre programa. El que farem és començar assegurant-nos que el compilador configurat a l'Eclipse és l'adequat. Anem a Project->Properties.



La finestra que sortirà és la següent. Hem de veure que el 'Compiler invocation command', està dirigit cap al 'arm-elf-gcc.



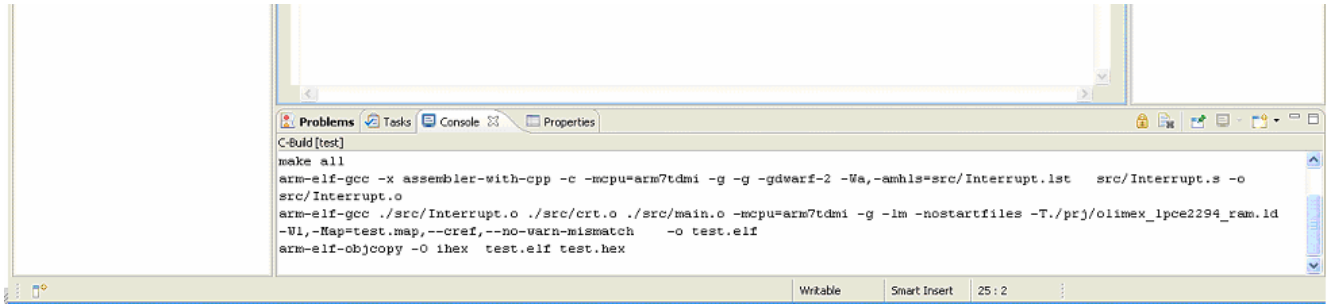
Un cop comprovat el compilador, ja podem compilar el projecte. Per això seleccionarem: Project->Build All.



I el programa es començarà a compilar. 'Make' és la utilitat que fa ser l'Eclipse per compilar, junt amb un fitxer del projecte que es diu 'makefile'. Podeu trobar-ne informació a: "Make (software)." *Wikipedia, The Free Encyclopedia*. 8 Apr 2008, 13:20 UTC. Wikimedia Foundation, Inc. 11 Apr 2008 <http://en.wikipedia.org/w/index.php?title=Make_%28software%29&oldid=204207337>. A la part de baix de l'entorn Eclipse podem veure el resultat de la compilació.

El compilador de fet segueix els passos descrits al fitxer 'makefile'. Obriu-lo i observeu-ne el contingut. Són fitxers francament complicats. Aquesta és la manera típica de compilar, per exemple, en entorns Linux. No farem res més per ara amb aquest fitxer.

El resultat de la compilació hauria de ser similar a:



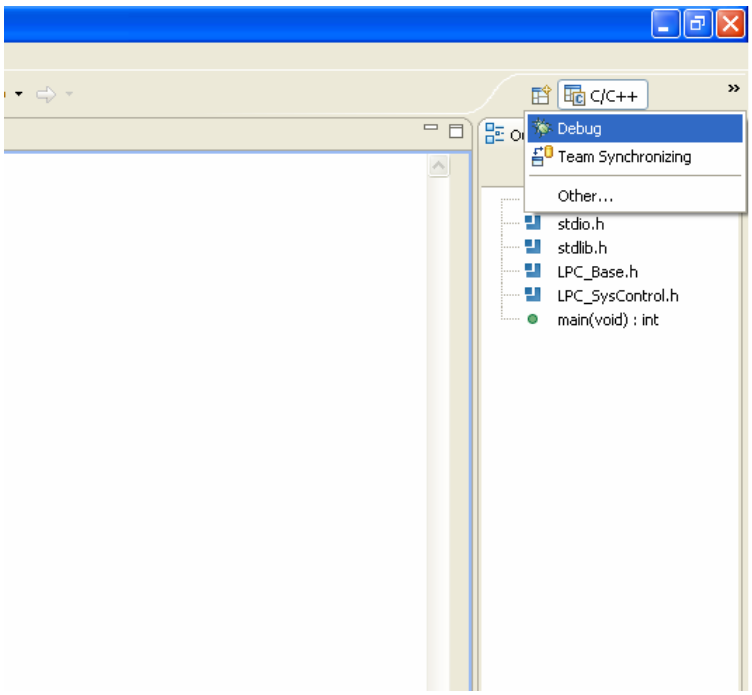
Un resum bàsic del procés que s'ha fet és:

- Compilar el fitxer main.c
- Compilar el fitxer Interrupt.s (en assemblador de l'ARM)
- Compilar el fitxer crt.s (en assemblador de l'ARM)
- 'Linkar' tots tres fitxers objectes per obtenir el fitxer final test.hex i test.elf, que es poden carregar directament a la placa i ser executats.

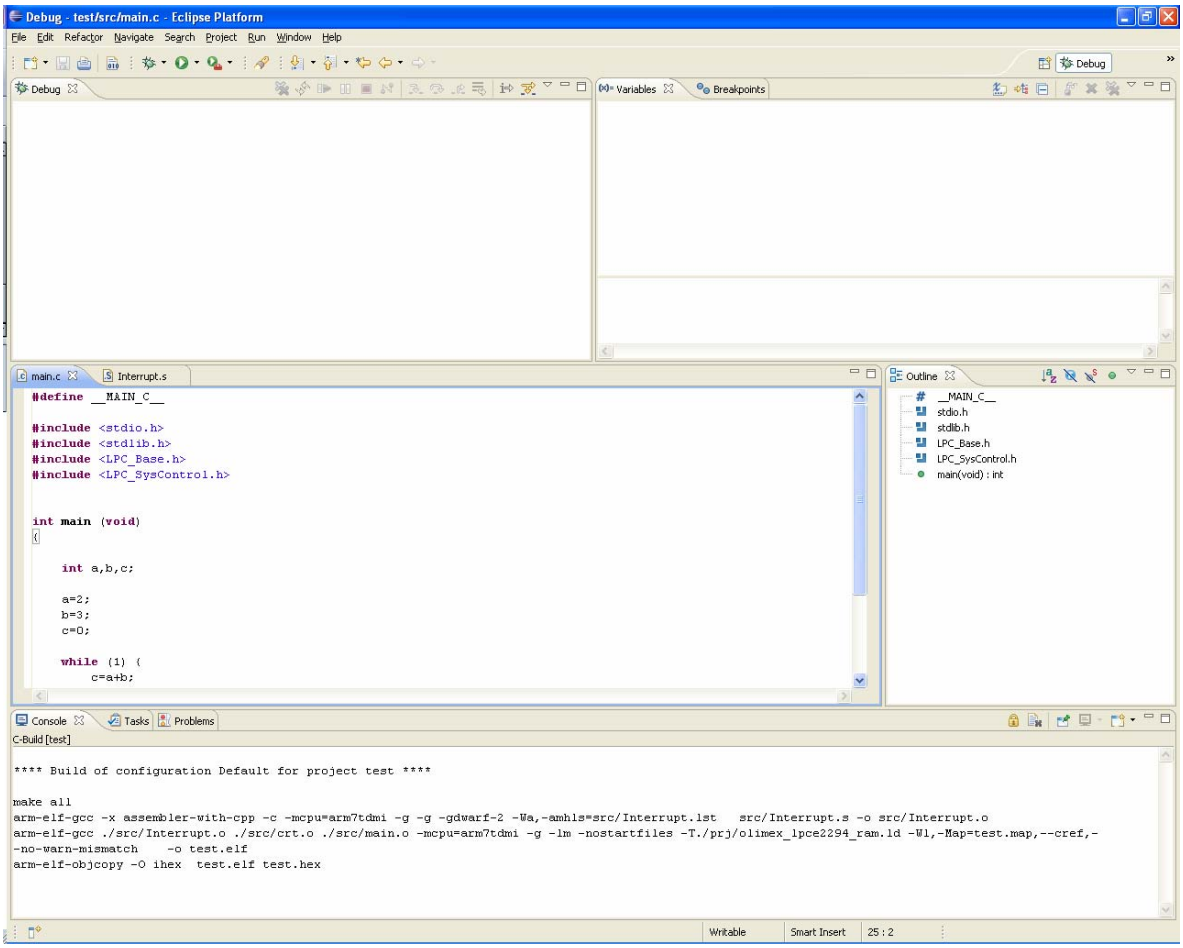
Pregunta: Trobeu el fitxer a on s'indica a partir de quina adreça es carregarà el programa a la placa.

Depuració del projecte 'test'

Ara volem depurar el programa que acabem de compilar. Per això obrirem la perspectiva de 'Debug' de l'Eclipse.



El resultat és el següent:

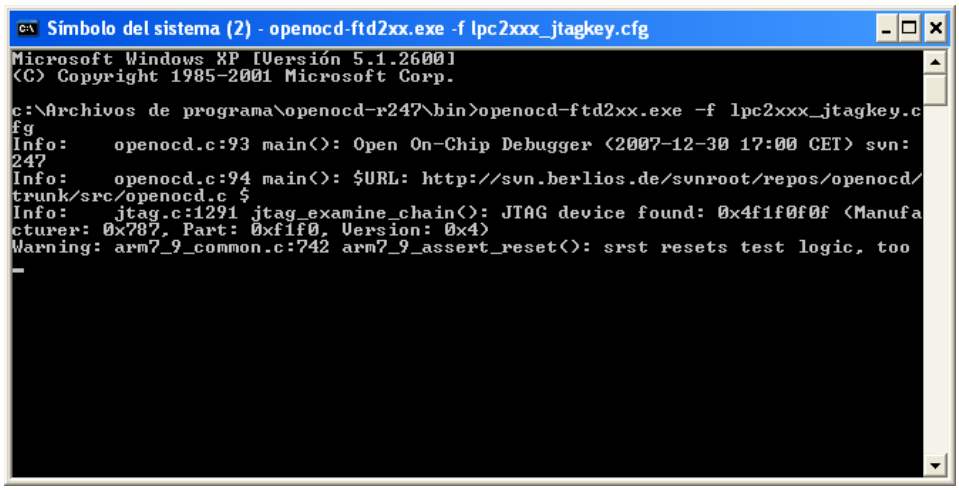


Ara, però, ens trobem amb un problema. Tot i que l'entorn Eclipse és molt potent, no està directament preparat per tractar amb cap sistema embedded en concret. Tot i així, podem fer servir més eines, en conjunció amb l'Eclipse, per tal que això sigui possible. Això implica versatilitat per part de l'Eclipse.

L'eina que farem servir és el OpenOCD (Open On-Chip Debugger). Aquest és un programa, que fa d'interface entre l'Eclipse i la nostra plataforma basada en l'ARM. La comunicació entre l'Eclipse i el OpenOCD es fa per TCP-IP, amb un port local del propi ordinador. És obvi que això pot permetre l'execució de pràctiques de manera remota. Per executar el OpenOCD, obrirem una finestra de DOS i executarem:

'openocd-ftd2xx.exe -f lpc2xxx_jtagkey.cfg'

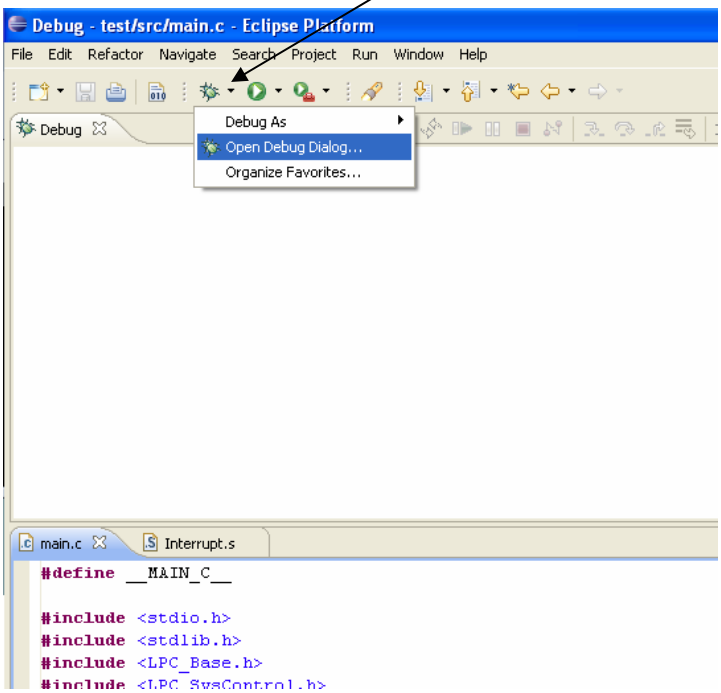
al directori a on s'hagi instal·lat el OpenOCD. Observeu el contingut del fitxer 'lpc2xxx_jtagkey.cfg'. Aquest fitxer li indica al OpenOCD certa configuració per poder-se connectar a la nostra placa mitjançant l'Amontec JTAG Tiny.



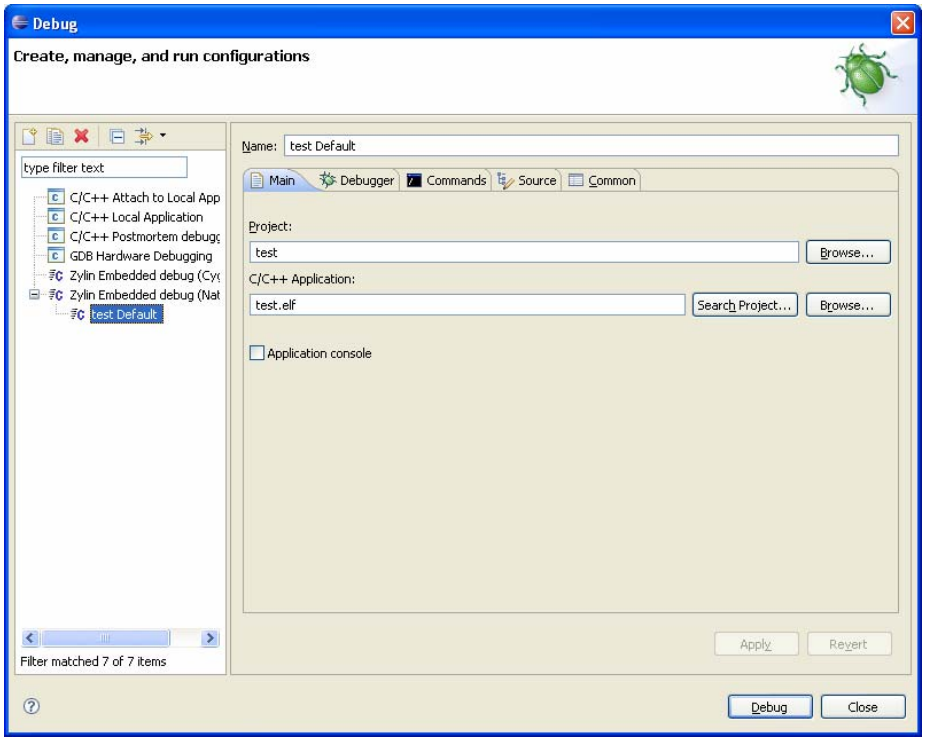
El programa es queda per tant executant-se i anirà donant missatges conforme anem depurant el nostre projecte.

Tornem a l'Eclipse. Ara, li hem de dir que s'ha de connectar via TCP-IP amb el OpenOCD, i acabar-lo de configurar. Aquesta operació només l'haurem de fer una vegada. Després el programa ja guarda tots aquests detalls.

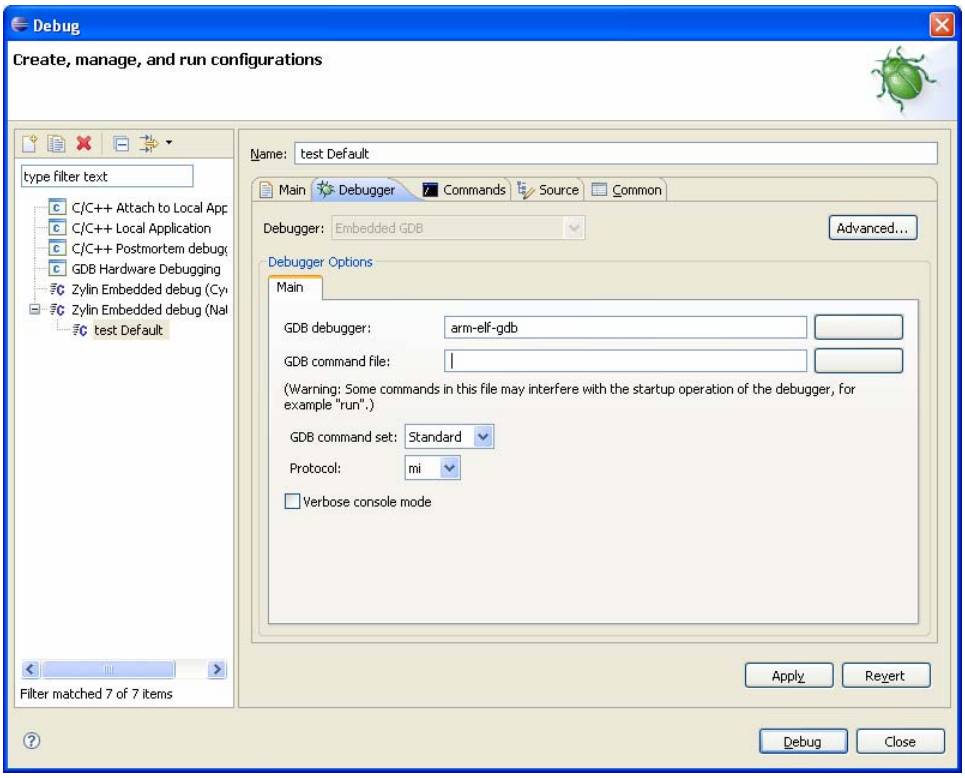
Anem a la icona del 'bug' i cliquem al triangle de la dreta (un triangle petit). Seleccionem: 'Open Debug Dialog':



S'obrirà la següent finestra:



Hem de fer doble click sobre 'Zylin Embedded debug (Native)', i apareixerà el 'test Default'. Aquest serà el fitxer de configuració de les depuracions pel projecte 'test'. En cas que no surti el nom del projecte, o el fitxer de l'aplicació, empleneu els apartats tal i com es veu a la figura anterior ('test' i 'test.elf).



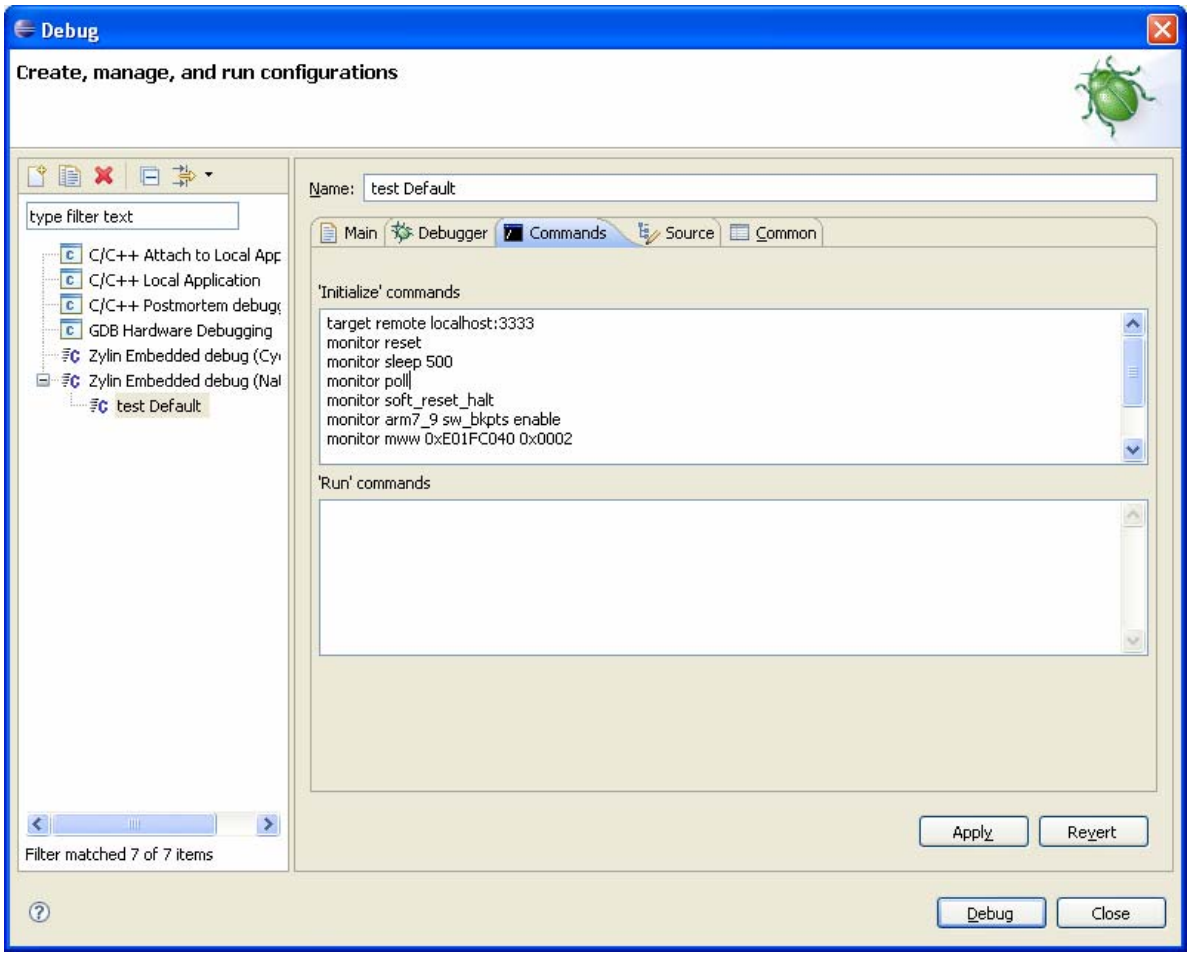
Canviem a l'apartat '**Debugger'** i deixeu la finestra tal i com apareix a la figura anterior. El debugger que farem servir és el 'arm-elf-gdb', un depurador creuat. Esborreu el nom del fitxer '.gdbinit' al 'GDB command file'.

A l'apartat '**Commands'** li hem de dir a l'Eclipse que es comunicarà amb un altre programa, en el nostre cas el OpenOCD que ja tenim obert a la finestra DOS, mitjançant l'adreça IP local: 'localhost' i el port 3333. Això vol dir que entre l'Eclipse i el OpenOCD s'estableix una comunicació TCP/IP local al mateix PC. El port que es fa servir és el 3333.

A més, hem de dir a l'Eclipse quines ordres ha d'enviar al OpenOCD. La llista d'ordres és la següent:

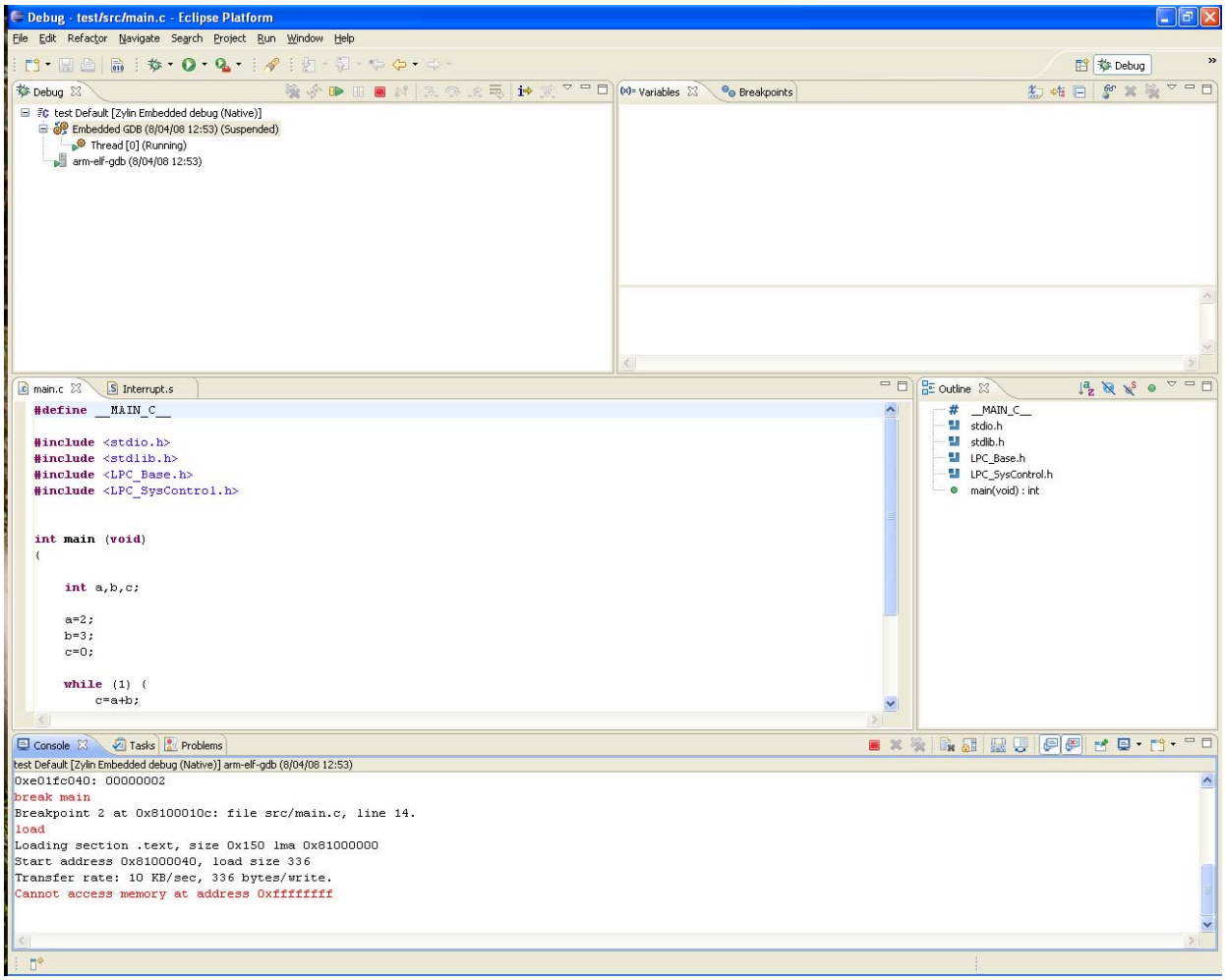
```
target remote localhost:3333
monitor reset
monitor sleep 500
monitor poll
monitor soft_reset_halt
monitor arm7_9 sw_bkpts enable
monitor mww 0xE01FC040 0x0002
monitor mdw 0xE01FC040
break main
load
```

Copiarem aquestes ordres les trobareu al fitxer 'ordres.txt' que hi ha al LPCEB-test.rar. Les heu de copiar a la finestra tal i com es pot veure a la següent figura:

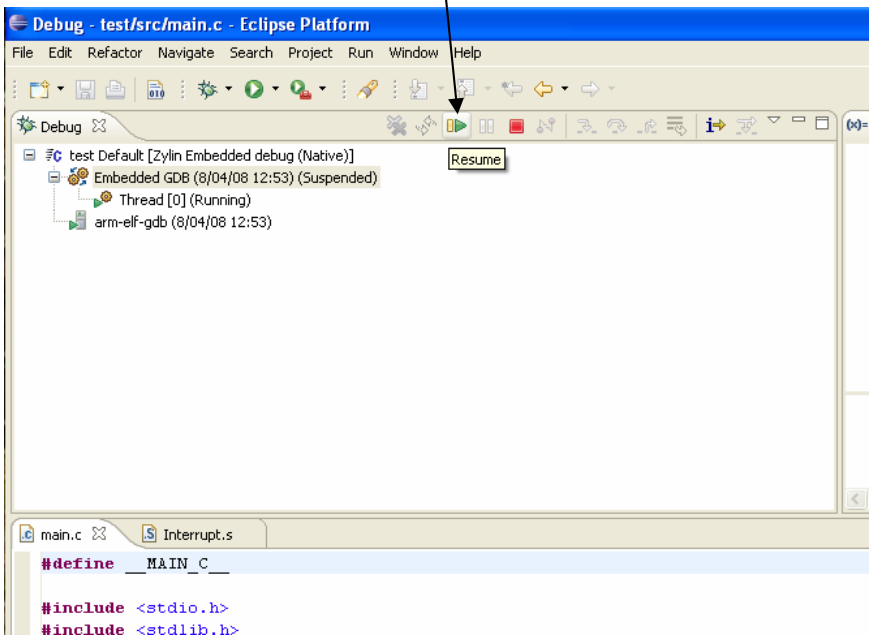


I prémer ara el botó de 'Debug'. El programa es carregarà a la placa i aniran sortint missatges a la part de baix de la finestra.

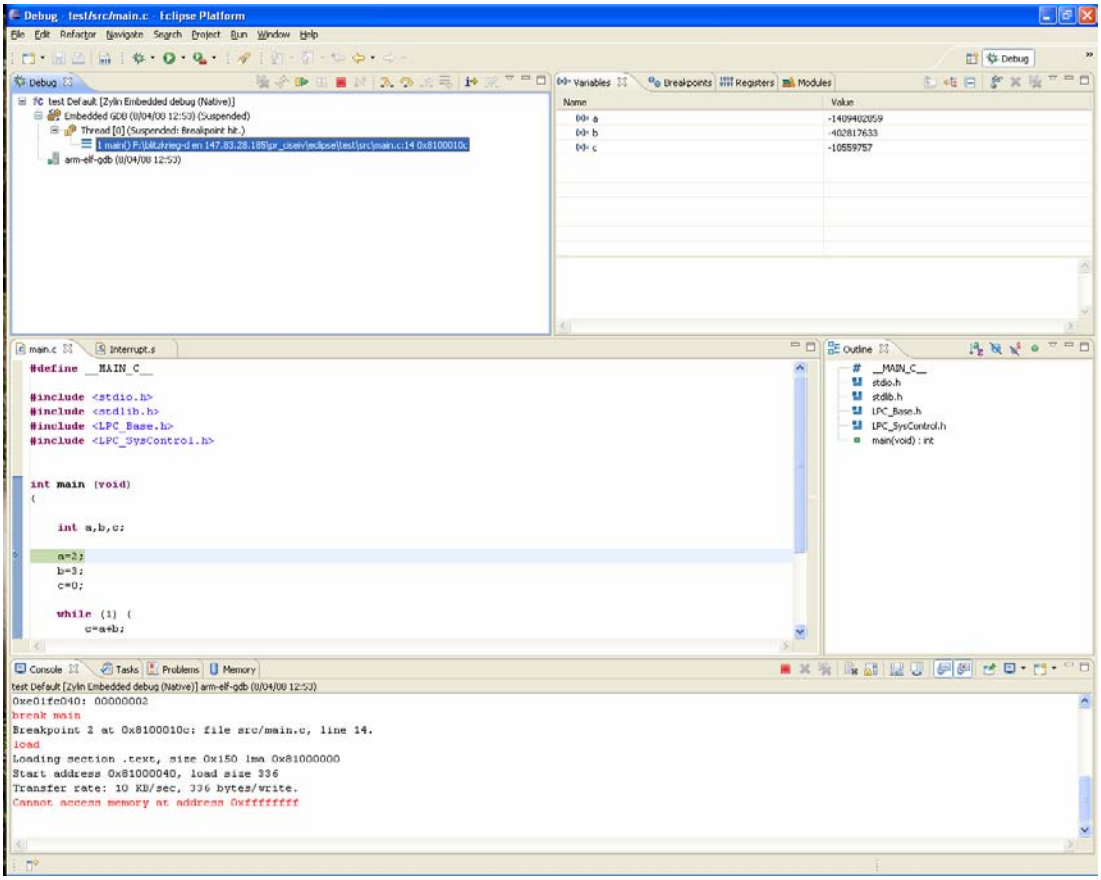
A la pantalla hauríem de veure el següent:



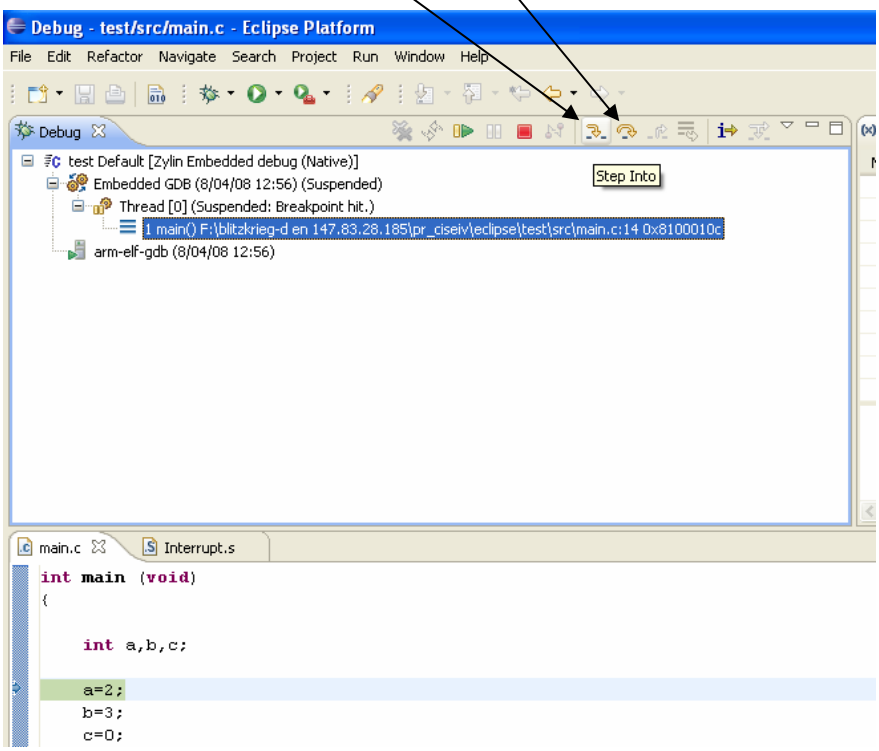
Entre les comandes que hem ficat cap el OpenOCD hi ha 'break main'. Això posa un breakpoint al 'main' del nostre programa. Per tant quan premem el símbol similar a un botó de 'play' l'execució es pararà a l'inici del nostre programa:

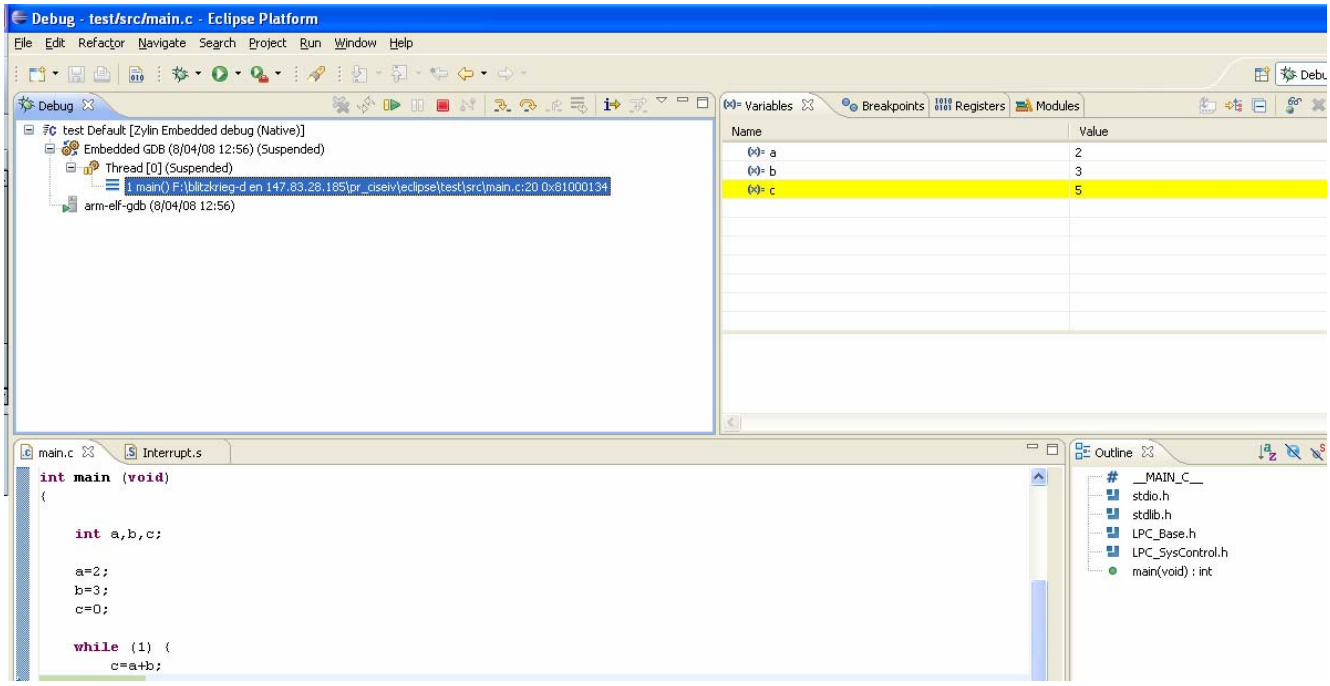


La franja de color blau de la línia de programa ‘a=2;’, vol dir que el depurador ha parat l’execució en aquest punt.



Podem executar pas a pas amb les fletxes ‘Step Into’ and ‘Step Over’ (la primera executa instrucció a instrucció entrant a les subrutines, la segona no entra a les subrutines sino que les executa directament i torna al programa principal).





Finalment, podem veure que la finestra de la dreta, a dalt, es pot veure el contingut de les variables. Podem parar la depuració prement el quadrat vermell.

Torneu a carregar el programar, tanqueu l'Eclipse i torneu-lo a obrir, familiaritzeu-vos amb l'entorn. Feu alguna modificació simple del programa, torneu a compilar i depurar, etc.

Amb aquest apartat hem assolit coneixements bàsics d'una eina genèrica, l'Eclipse, per tal de treballar amb sistemes embedded.

NOTA: De vegades conforme anem treballant, hi ha algun error a l'OpenOCD. Convé anar revisant la finestra DOS a on l'executem per verificar que no hi ha cap problema. En qualsevol cas, feu això quan l'Eclipse doni errors al comunicar-se amb la placa.

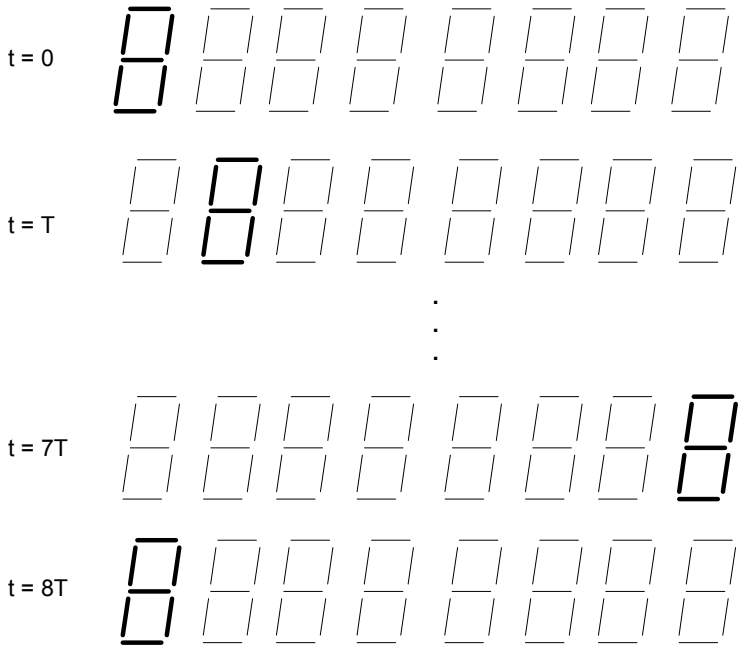
4. Control dels displays de set segments

Objetius concrets:

- **Controlar els ports d'E/S del microcontrolador,**
- **Presentar un missatge pels displays de 7 segments de la placa.**

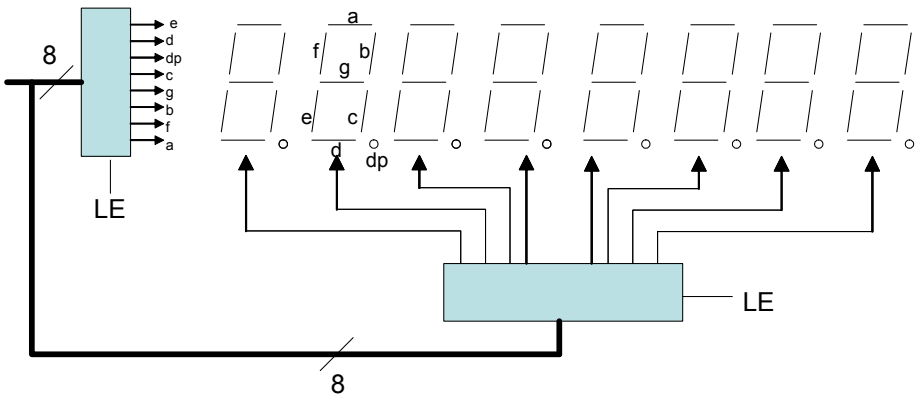
A la placa LPCEB2000-I hi ha 8 displays de 7 segments i el punt. En principi per tal de controlar-los necessitaríem 8 línies per cada dígit, i per tant un total de 8x8=64 línies. Per tal de no utilitzar tants pins del microcontrolador, una de les tècniques que es fa servir normalment és fer una visualització *multiplexada en temps*. En tot moment només tindrem com a màxim un display encès, i aquest dígit anirà canviant ràpidament per donar-nos la impressió de que tots els dígits estan encesos alhora.

És a dir, cada cert temps (T) canviarem el dígit que s'encén:



Si fem que el temps T sigui prou curt ens semblarà que tots els dígits estan encesos alhora. Per poder fer això necessitem 8 línies de control que ens permetin encendre els 7 segments més el punt d'un display, i 8 línies més d'habilitació de cada display. Per tant hem reduït el nombre de línies necessàries de 64 a 16.

Podem reduir aquest nombre encara més utilitzant elements de memòria que emmagatzemin quins segments han d'estar encesos en un moment donat i quin display ha d'estar activat. L'esquema seria semblant al de la següent figura:



En aquest esquema fem servir dos elements de memòria, latch, amb dos senyals d'habilitació (LE). En total fem servir només 10 senyals de control (els 8 per a cada latch més 2 d'habilitació). A la següent figura podem veure quines són les connexions dels displays que hi ha a la placa LPCEB2000-I.

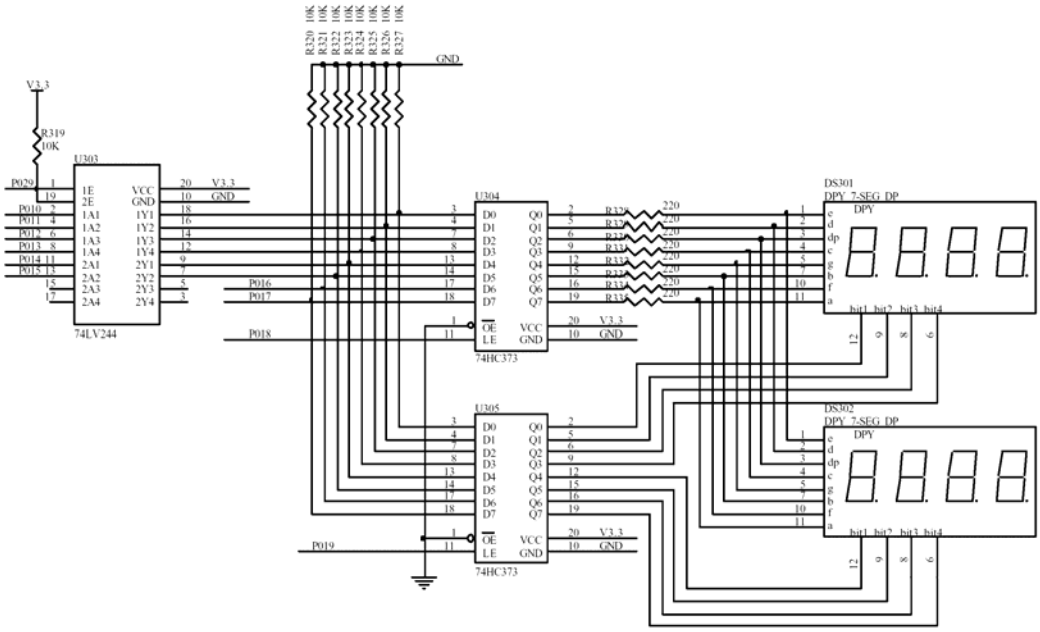


Fig 1: Esquema de connexió dels displays de 7-segments.

El primer component (74LV244) és un *buffer*. Per tenir-lo activat haurem de posar a '0' el bit P0.29. Els dos components següents (74HC373) són dos *latches*. Aquests components passen el valor Dn a Qn només quan LE=1. Si LE=0, els senyals Qn mantenen el seu valor.

NOTA: Per encendre uns sements a un dígit cal posar a '0' el senyal d'habilitació del dígit (hit1, hit2, hit3 o hit4) i posar uns a les entrades (a,b,c,d,e,f,g o dp) corresponents dels segments.

Pins de ports d'E/S involucrats (Port 0):
P0.10 – P0.17, P0.18, P0.19 i P0.29.

Per configurar aquests pins com a sortides hem de programar adequadament els registres següents (Capítol 9 del Manual d'usuari del LPC2292: GPIO):

*PINSEL0 (configurar com a GPIO els pins P0.10-P0.15)

Table 58: Pin Function Select Register 0 for LPC2119/2129/2292 (PINSEL0 - 0xE002C000)

PINSEL0	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.0	GPIO Port 0.0	TxD (UART0)	PWM1	Reserved	00
3:2	P0.1	GPIO Port 0.1	RxD (UART0)	PWM3	EINT0	00
5:4	P0.2	GPIO Port 0.2	SCL (I^2C)	Capture 0.0 (TIMER0)	Reserved	00
7:6	P0.3	GPIO Port 0.3	SDA (I^2C)	Match 0.0 (TIMER0)	EINT1	00
9:8	P0.4	GPIO Port 0.4	SCK (SPI0)	Capture 0.1 (TIMER0)	Reserved	00
11:10	P0.5	GPIO Port 0.5	MISO (SPI0)	Match 0.1 (TIMER0)	Reserved	00
13:12	P0.6	GPIO Port 0.6	MOSI (SPI0)	Capture 0.2 (TIMER0)	Reserved	00
15:14	P0.7	GPIO Port 0.7	SSEL (SPI0)	PWM2	EINT2	00
17:16	P0.8	GPIO Port 0.8	TxD UART1	PWM4	Reserved	00
19:18	P0.9	GPIO Port 0.9	RxD (UART1)	PWM6	EINT3	00
21:20	P0.10	GPIO Port 0.10	RTS (UART1)	Capture 1.0 (TIMER1)	Reserved	00
23:22	P0.11	GPIO Port 0.11	CTS (UART1)	Capture 1.1 (TIMER1)	Reserved	00
25:24	P0.12	GPIO Port 0.12	DSR (UART1)	Match 1.0 (TIMER1)	Reserved	00
27:26	P0.13	GPIO Port 0.13	DTR (UART1)	Match 1.1 (TIMER1)	Reserved	00
29:28	P0.14	GPIO Port 0.14	CD (UART1)	EINT1	Reserved	00
31:30	P0.15	GPIO Port 0.15	RI (UART1)	EINT2	Reserved	00

*PINSEL1 (configurar com a GPIO els pins P0.16-P0.19 i el P0.29)

Table 61: Pin Function Select Register 1 for LPC2119/2129/2292 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.16	GPIO Port 0.16	EINT0	Match 0.2 (TIMER0)	Reserved	00
3:2	P0.17	GPIO Port 0.17	Capture 1.2 (TIMER1)	SCK (SPI1)	Match 1.2 (TIMER1)	00
5:4	P0.18	GPIO Port 0.18	Capture 1.3 (TIMER1)	MISO (SPI1)	Match 1.3 (TIMER1)	00
7:6	P0.19	GPIO Port 0.19	Match 1.2 (TIMER1)	MOSI (SPI1)	Match 1.3 (TIMER1)	00
9:8	P0.20	GPIO Port 0.20	Match 1.3 (TIMER1)	SSEL (SPI1)	EINT3	00
11:10	P0.21	GPIO Port 0.21	PWM5	Reserved	Capture 1.3 (TIMER1)	00
13:12	P0.22	GPIO Port 0.22	Reserved	Capture 0.0 (TIMER0)	Match 0.0 (TIMER0)	00
15:14	P0.23	GPIO Port 0.23	RD2 (CAN Controller 2)	Reserved	Reserved	00
17:16	P0.24	GPIO Port 0.24	TD2 (CAN Controller 2)	Reserved	Reserved	00
19:18	P0.25	GPIO Port 0.25	RD1 (CAN Controller 1)	Reserved	Reserved	00
21:20	P0.26	Reserved				00
23:22	P0.27	GPIO Port 0.27	AIN0 (A/D Converter)	Capture 0.1 (TIMER0)	Match 0.1 (TIMER0)	01
25:24	P0.28	GPIO Port 0.28	AIN1 (A/D Converter)	Capture 0.2 (TIMER0)	Match 0.2 (TIMER0)	01
27:26	P0.29	GPIO Port 0.29	AIN2 (A/D Converter)	Capture 0.3 (TIMER0)	Match 0.3 (TIMER0)	01
29:28	P0.30	GPIO Port 0.30	AIN3 (A/D Converter)	EINT3	Capture 0.0 (TIMER0)	01
31:30	P0.31	Reserved				00

***IODIR** (configurar els pins P0.10-P0.19 i P0.29 com a sortides).

GPIO Direction Register (IO0DIR - 0xE0028008, IO1DIR - 0xE0028018)

This register is used to control the direction of the pins when they are configured as GPIO port pins. Direction bit for any pin must be set according to the pin functionality.

Table 72: GPIO Direction Register (IO0DIR - 0xE0028008, IO1DIR - 0xE0028018)

IODIR	Description	Value after Reset
31:0	Direction control bits (0 = INPUT, 1 = OUTPUT). Bit 0 in IO0DIR controls P0.0 ... Bit 31 in IO0DIR controls P0.31	0

Per exemple, la següent rutina configura els Pins P0.10:P0.19 i el Pin P0.29 com a sortides:

```
void initdisplay(void)
{
  *PINSEL0&=0x000FFFFF;                                  // Pins P0.10-15 son GPIO
  *PINSEL1&=0xF3FFFF00;                                  // Pins P0.16-19 i P0.29 son GPIO
  *IODIR0|=P10+P11+P12+P13+P14+P15+P16+P17+P18+P19+P29;  // Pins P0.10:19 i P0.29 son sortides

  *IOCLR0=P29;                                           // Posem a '0' bit P0.29.
}
```

Les etiquetes de registres (PINSEL0, PINSEL1, IODIR0, etc.) estan definides al fitxer LPC_Base.h. Les definicions de P10, P11, etc., (o B10, B11, etc.) es troben a LPC_SysControl.h. Aquestes etiquetes són molt útils perquè ens permeten posar a 0 bits d'un registre d'una manera molt simple:

Per exemple, si volem posar a 0 els bits 7, 16, 21 i 28 d'un registre (apuntat per 'reg') podem fer:

```
*reg &=~(B7+B16+B21+B28);
```

Si volem posar a '1' els bits 4, 13, i 19, podríem fer (es pot fer també amb *Pn*):

```
*reg |=B4+B13+B19;
```

Finalment, cal notar com funcionen els registres ***IOCLR0, *IOSET0 i *IOPIN0**.

GPIO Pin Value Register (IO0PIN - 0xE0028000, IO1PIN - 0xE0028010)

This register provides the value of the GPIO pins. This value reflects any outside world influence on the pins.

Note: for test purposes, writing to this register stores the value in the output register, bypassing the need to use both the IOSET and IOCLR registers. This feature is of little or no use in an application because it is not possible to write to individual bytes in this register.

Table 69: GPIO Pin Value Register (IO0PIN - 0xE0028000, IO1PIN - 0xE0028010)

IOPIN	Description	Value after Reset
31:0	GPIO pin value bits. Bit 0 in IO0PIN corresponds to P0.0 ... Bit 31 in IO0PIN corresponds to P0.31	Undefined

GPIO Output Set Register (IO0SET - 0xE0028004, IO1SET - 0xE0028014)

This register is used to produce a HIGH level output at the port pins if they are configured as GPIO in an OUTPUT mode. Writing 1 produces a HIGH level at the corresponding port pins. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOSET has no effect.

Reading the IOSET register returns the value in the GPIO output register, as determined by previous writes to IOSET and IOCLR (or IOPIN as noted above). This value does not reflect the effect of any outside world influence on the I/O pins.

Table 70: GPIO Output Set Register (IO0SET - 0xE0028004, IO1SET - 0xE0028014)

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit 0 in IO0SET corresponds to P0.0 ... Bit 31 in IO0SET corresponds to P0.31	0

GPIO Output Clear Register (IO0CLR - 0xE002800C, IO1CLR - 0xE002801C)

This register is used to produce a LOW level at port pins if they are configured as GPIO in an OUTPUT mode. Writing 1 produces a LOW level at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOCLR has no effect.

Table 71: GPIO Output Clear Register (IO0CLR - 0xE002800C, IO1CLR - 0xE002801C)

IOCLR	Description	Value after Reset
31:0	Output value CLEAR bits. Bit 0 in IO0CLR corresponds to P0.0 ... Bit 31 in IO0CLR corresponds to P0.31	0

Quan escrivim una dada de 21 bits a *IOCLR0 o *IOSET0, només canviem els bits, configurats com a sortides GPIO amb *PINSEL0 o *PINSEL1, en els que escrivim uns. Es a dir, si tot el port 0 està configurat com a sortida i fem:

```
*IOCLR0=P1+P7;
*IOSET0=P8+P15;
```

Estem posant, amb la primera instrucció, els bits P0.1 i P0.7 a ‘0’, i amb la segona, els bits P0.8 o P0.15 a ‘1’. La resta continua amb els valor que tenien. **Noteu que no és necessari, ni de fet recomanable, executar instruccions com ‘*IOSETn | =P8+P15’ (fent una OR), amb els registres IOSETn o IOCLRn. Per què ?**

Per llegir quins valors tenen els bits, ja estiguin configurats com a sortides o entrades llegirem el registre *IOPIN0:

```
variable=*IOPIN0;.
```

L.1.1 Feu un programa que escrigui un missatge als displays.

Caldrà fer una rutina que configuri adequadament els pins P0.10:P0.19 i P0.29 com a sortides, i multiplexi la visualització dels caràcters als displays. Els retards (T) els podeu fer amb bucles d’espera:

```
int i;
float a;

for (i=0, a=0.0;i<10000;i++) a++;   // Generem un retard
```

Més endavant farem servir funcions molt més adequades que faran ús dels temporitzadors.

5. Timer0 i Timer1 + Vectored Interrupt Controller

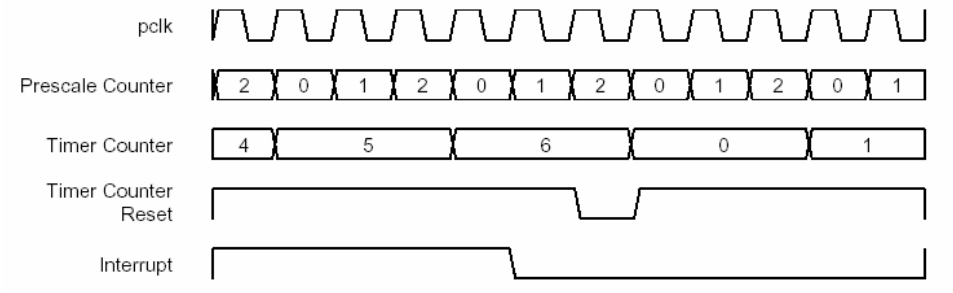
Objetius concrets:

- **Entendre el funcionament de les interrupcions vectoritzades als µC LPC2XXX,**
- **Entendre el funcionament dels temporitzadors,**
- **Programar els Timers per a generar interrupcions a una determinada freqüència.**

Els temporitzadors del LPC2292 segueixen l'estructura habitual que podem trobar a la majoria de microcontroladors. Bàsicament són uns comptadors que es van incrementant periòdicament fins que arriben a un valor que el programador ha decidit. Quan el comptador arriba a aquest valor passa a '0' i es pot generar una interrupció.

Per tenir més flexibilitat a l'hora de programar els períodes d'interrupció normalment s'afegeix un comptador de prescalat, el qual quan arriba a un determinat valor és el que fa que el comptador anterior s'incrementi en una unitat.

Es a dir, suposem que el comptador de prescalat està programat per incrementar el comptador principal quan arriba a 2, i el comptador principal està programat per passar a 0 un cop arriba a 6. Veuríem la següent evolució:



Cada cop que el comptador de prescalat arriba a 2 (el límit que s'ha programat en aquest cas) augmentem el comptador del timer. Un cop el comptador arriba a 6, es demana interrupció i el comptador passa a 0 (quan així ho indica el comptador de prescalat). El rellotge *pclk* (peripheral clock) és de 10MHz, per defecte.

Els registres per configurar el timer0 d'aquesta manera són (Capítol 15 del Manual d'usuari del LPC2292):

*TIMER0_TCR: Timer0 Control Register

Table 159: Timer Control Register (TCR: TIMER0 - T0TCR: 0xE0004004; TIMER1 - T1TCR: 0xE0008004)

TCR	Function	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0

És necessari fer primer reset als Timers i després, un cop configurats posar-los en marxa.

*TIMER0_PR: Timer0 Prescaler Register

Prescale Register (PR: TIMER0 - T0PR: 0xE000400C; TIMER1 - T1PR: 0xE000800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

*TIMER0_MR0: Timer0 Match Register

Match Registers (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

*TIMER0_MCR: Timer 0 Match Control Register

Table 160: Match Control Register (MCR: TIMER0 - T0MCR: 0xE0004014; TIMER1 - T1MCR: 0xE0008014)

MCR	Function	Description	Reset Value
0	Interrupt on MR0	When one, an interrupt is generated when MR0 matches the value in the TC. When zero this interrupt is disabled.	0
1	Reset on MR0	When one, the TC will be reset if MR0 matches it. When zero this feature is disabled.	0
2	Stop on MR0	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. When zero this feature is disabled.	0
3	Interrupt on MR1	When one, an interrupt is generated when MR1 matches the value in the TC. When zero this interrupt is disabled.	0
4	Reset on MR1	When one, the TC will be reset if MR1 matches it. When zero this feature is disabled.	0
5	Stop on MR1	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. When zero this feature is disabled.	0
6	Interrupt on MR2	When one, an interrupt is generated when MR2 matches the value in the TC. When zero this interrupt is disabled.	0
7	Reset on MR2	When one, the TC will be reset if MR2 matches it. When zero this feature is disabled.	0
8	Stop on MR2	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. When zero this feature is disabled.	0
9	Interrupt on MR3	When one, an interrupt is generated when MR3 matches the value in the TC. When zero this interrupt is disabled.	0
10	Reset on MR3	When one, the TC will be reset if MR3 matches it. When zero this feature is disabled.	0
11	Stop on MR3	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. When zero this feature is disabled.	0

A continuació teniu un exemple de configuració del Timer0 per interrompre a una certa freqüència.

```
void initimer0(void)
{
  *TIMER0_TCR    =       B1;     // Timer Reset
  *TIMER0_PR     =       0x02F;  // Valor pel registre de prescalat
  *TIMER0_MR0    =       0xF0;   // Valor del reg. de match0 per fer reset comptador
  *TIMER0_MCR    =       B0+B1;  // Interrupcio i reset quan MR0
  *TIMER0_TCR    =       B0;     // El posem en marxa
}
```

Aquest exemple no és complert perquè tot i que configurem el timer0 per generar interrupcions no les activem ni indiquem quina serà la rutina de servei d'interrupció. Això ho veurem a la següent secció.

NOTA: El Timer compta una unitat amb el rellotge *pclk* (peripheral clock), que per defecte és de 10MHz. Les definicions de *Bn*, es troben a *LPC_SysControl.h*, del projecte *test*: $Bn=2^n$.

VIC: Vectored Interrupt Controller del LPC2292

El microcontrolador LPC2292 té 3 maneres diferents de respondre a interrupcions (Capítol 6 del Manual d'usuari del LPC2292):

a) Interrupcions ràpides: Fast Interrupt reQuest (FIQ)
b) Interrupcions vectoritzades: Vectored IRQ (16 canals com a màxim)
c) Interrupcions no vectoritzades: Non-vectored IRQ.

Les interrupcions que farem servir són les IRQ vectoritzades. Hi ha 28 fonts d'interrupció degudes a perifèrics interns o interrupcions externes:

Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
TIMER0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
TIMER1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I^2C	SI (state change)	9
SPI0	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	10
SPI1	SPI Interrupt Flag (SPIF) Mode Fault (MODF)	11
PLL	PLL Lock (PLOCK)	12
RTC	Counter Increment (RTCCIF) Alarm (RTCALF)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16
System Control	External Interrupt 2 (EINT2)	17
A/D	A/D Converter	18
CAN	CAN and Acceptance Filter 1 ORed CAN, LUTerr int CAN1 and CAN2: 2x(Tx int, Rx int) LPC2119/2129/2292/2294 CAN3 and CAN4: 2x(Tx int, Rx int) LPC2194/2292/2294 only	19 20-23 24-27

Fig 2: Fonts d'interrupció al LPC2292.

Com podem veure la interrupció generada pel Timer0 és la 4. Volem que sigui vectoritzada, i això ho indiquem amb el registre *VICIntSelect:

VICIntSelect	Function	Reset Value
31:0	1: the interrupt request with this bit number is assigned to the FIQ category. 0: the interrupt request with this bit number is assigned to the IRQ category.	0

i per tant haurem d'escriure un '0' al bit 4 d'aquest registre. El LPC2292 té fins a 16 canals d'interrupció vectoritzada (0-15). Podem configurar un canal qualsevol (per example el 0) per controlar les interrupcions del Timer0. Cada canal

té associat un registre de control. Si volem que el canal 0 controli les interrupcions del Timer0 hem de configurar el registre *VICVectCntl0 adequadament:

Table 49: Vector Control Registers (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)

VICVectCntl0-15	Function	Reset Value
5	1: this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
4:0	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower-numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0

Per exemple, per utilitzar el canal 0 per al Timer0 executarem:

```
*VICVectCntl0 = B5 + 0x4;  // Activem Canal 0 d'IRQ vectoritzada per la int. 4
```

El següent pas és donar l'adreça de la RSI al canal 0 del VIC (Vectored Interrupt Controller):

```
*VICVectAddr0 = (int)TIMER0_VectISR;
```

on estem dient que la rutina `TIMER0_VectISR` serà la RSI que atendrà la interrupció del Canal 0 que hem associat al Timer0. Les rutines de servei d'interrupció es caracteritzen perquè han de salvar tots els registres al començar i recuperar-los al finalitzar. D'aquesta manera el programa principal que ha estat interromput no es veu afectat per l'execució de la RSI. Veurem amb més detall en el següent punt com es fa això al nostre sistema.

Finalment cal activar les interrupcions del Timer0, amb el registre ***VICIntEnable**:

```
*VICIntEnable |= B4;    // Habilitem les interrupcions del Timer (font 4).
```

Interrupt Enable Register (VICIntEnable - 0xFFFFF010, Read/Write)

This register controls which of the 32 interrupt requests and software interrupts contribute to FIQ or IRQ.

Table 44: Interrupt Enable Register (VICINtEnable - 0xFFFFF010, Read/Write)

VICIntEnable	Function	Reset Value
31:0	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See the VICIntEnClear register (Table 45 below), for how to disable interrupts.	0

Tot i que no ho farem servir explícitament ara, per deshabilitar interrupcions podríem fer servir el següent registre:

Interrupt Enable Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

This register allows software to clear one or more bits in the Interrupt Enable register, without having to first read it.

Table 45: Software Interrupt Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

VICIntEnClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request. 0: writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	0

Què hem fet fins ara ?

a) Hem activat el Timer0, li hem donat un valor al registre de prescalat i al registre de 'match' per tal de produir una interrupció un cop el Timer0 arribi a aquest valor. Això ho hem fet donant valors adequats als registres:

TIMER0_TCR:	*Reset del Timer0*
TIMER0_PR:	*Donem valor al registre de prescalat*
TIMER0_MR0:	*Valor del registre de Match 0 (MR0)*
TIMER0_MCR:	*Fem que demani int. (4), quan el comptador arribi a MR0, i es posi a 0.*
TIMER0_TCR:	*Habilitem el Timer0*

b) Un cop fet això, no es produirà cap interrupció. Això és degut a que no hem configurat el Controlador d'Interrupcions Vectoritzades (VIC) del LPC2292. És a dir, el perifèric pot generar la petició d'interrupció però ningú no en farà cas.

Per tal que el VIC tingui en compte les peticions d'interrupcions, hem configurat un canal dels 16 que té el VIC, per tal que dugui a terme la interrupció demanada pel Timer0. Hem triat el canal 0. Això ho hem fet configurant els registres:

VICIntSelect:	*Seleccionem int. 4 com a vectoritzada*
VICVectCntl0:	*Activem canal 0 VIC per a la int. 4*
VICVectAddr0:	*Donem adreça RSI de canal 0 de VIC*
VICIntEnable:	*Activem interrupcions 4 (del Timer0)*

I, ara, què ens queda per fer ?

a) Totes les Rutines de Servei d'Interrupció han de salvar els registres a la pila i recuperar-los en acabar. En el nostre cas, ho hem de fer en assemblador.

b) Escriure la RSI.

Ho veurem a continuació...

Rutines de Servei d'Interrupció al LPC2292

Una rutina de servei d'interrupció ha de salvar els registres del microprocessador al començar i ha de recuperar-los al finalitzar. En el nostre entorn de programació ho hem de fer en ensamblador, i per això farem servir una Macro que està definida al fitxer **Interrupt.s** del nostre projecte:

```
.macro IRQHandle in_handle,out_handle
        .extern \in_handle
        .global \out_handle
\out_handle:
        stmdb    sp!, {r0-r11, ip, lr}        # Guardar registres a la pila
        ldr      r0,  =\in_handle
        mov      lr,  pc
        bx       r0                           # Saltem a RSI de l'usuari
        ldmia    sp!, {r0-r11, ip, lr}        # Recuperem registres de la pila
        subs     pc,  r14, #4
.endm
```

Aquesta macro té dos paràmetres d'entrada: `in handle` i `out handle`. La crida a la macro es fa de la següent manera dins el fitxer **Interrupt.s**:

```
    IRQHandle TIMER0_ISR,TIMER0_VectISR
```

El fet que això sigui una macro vol dir que la sentència anterior és idèntica a haver escrit:

```
         .extern TIMER0_ISR
         .global TIMER0_VectISR
TIMER0_VectISR:
        stmdb    sp!, {r0-r11, ip, lr}        # Guardar registres a la pila
        ldr      r0,  TIMER0_ISR
        mov      lr,  pc
        bx       r0                           # Saltem a RSI de l'usuari
        ldmia    sp!, {r0-r11, ip, lr}        # Recuperem registres de la pila
        subs     pc,  r14, #4                 # Tornem a programa principal
```

on TIMER0_VectISR és definida com una etiqueta global (pot ser referenciada a un altre fitxer del projecte) i l'etiqueta TIMER0_ISR es suposa que és externa (està definida a un altre fitxer del projecte). L'etiqueta TIMER0_VectISR serà l'adreça que passarem al Controlador d'interrupcions, i TIMER0_ISR serà la nostra rutina de servei d'interrupcions, la de l'usuari.

Per tant al nostre fitxer C podem executar l'ordre que hem vist abans (`*VICVectAddr0 = (int)TIMER0_VectISR;`), a on diem al controlador que l'adreça de la RSI que respondrà al canal 0.

La rutina de servei d'interrupció pròpiament dita pot incrementar una variable (*tics*) per generar retards. En qualsevol cas cal, just abans de sortir de la RSI, desactivar la petició d'interrupció del Timer0. Si no ho féssim, en sortir, continuaríem tenint pendent la petició d'interrupció que va generar l'execució de la RSI. Això seria un problema, perquè executaríem la RSI més d'un cop. Per desactivar la interrupció disposem del registre TIMER0_IR

Interrupt Register (IR: TIMER0 - T0IR: 0xE0004000; TIMER1 - T1IR: 0xE0008000)

The Interrupt Register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 158: Interrupt Register (IR: TIMER0 - T0IR: 0xE0004000; TIMER1 - T1IR: 0xE0008000)

IR	Function	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.	0
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.	0

i cal dir-li al controlador d'interrupcions vectoritzades que s'ha acabar la RSI. Això fem escrivint un zero al registre *VICVectAddr*. Això és el típic **EOI (End of Interrupt)** del VIC. Com es pot llegir al 'VIC usage notes' (p. 101):

Following the completion of the desired interrupt service routine, clearing of the interrupt flag on the peripheral level will propagate to corresponding bits in VIC registers (VICRawIntr, VICFIQStatus and VICIRQStatus). Also, before the next interrupt can be serviced, it is necessary that write is performed into the VICVectAddr register before the return from interrupt is executed. This write will clear the respective interrupt flag in the internal interrupt priority hardware.

Aquesta pot ser la RSI del Timer0:

```
void TIMER0_ISR(void)
{
  tics++;

  *TIMER0_IR |= 0x1;            //anulem peticio int.
  *VICVectAddr = 0x00;          //EOI del VIC
                                //(end of int.)
}
```

Ara ja estem en condicions de fer un exemple complert amb el Timer0:

Exemple control Timer0

```
extern void TIMER0_VectISR(void);

unsigned int tics=0;

void retard(int n)
{
  tics=0;
  while (tics<n);
}
void TIMER0_ISR(void)
{
  tics++;

  *TIMER0_IR |= 0x1;            //anulem peticio int.
  *VICVectAddr = 0x00;          //EOI del VIC
                                //(end of int.)
}
void initimer0(void)
{
  *TIMER0_TCR    =       B1;     // Timer0 Reset
  *TIMER0_PR     =       0x02F;  // Prescaler
  *TIMER0_MR0    =       0xF0;   // Match0
  *TIMER0_MCR    =       B0 + B1;
  *TIMER0_TCR    =       B0;     // Timer0 ON


  *VICIntSelect&=~B4;   // Timer0 Int. es IRQ
  *VICVectAddr0 = (int)TIMER0_VectISR;
  *VICVectCntl0 = B5 + 0x4;
  *VICIntEnable|=B4;
}
```

Al fitxer *Interrupt.s* posarem:

```
.macro IRQHandle in_handle,out_handle
        .extern \in_handle
        .global \out_handle
\out_handle:
        stmdb    sp!, {r0-r11, ip, lr}
        ldr      r0,  =\in_handle
        mov      lr,  pc
        bx       r0
        ldmia    sp!, {r0-r11, ip, lr}
        subs     pc,  r14, #4
.endm

    IRQHandle TIMER0_ISR,TIMER0_VectISR
```

Estudi previ:

EP2.1: El rellotge que governa les transicions del registre de prescalat dels timers és el 'pclk' (periphery clock). Aquest rellotge està programat per defecte a 10 MHz. Tenint en compte això, quina és la freqüència d'interrupció del *Timer0*, quin temps triga a executar-se la crida '***retard(20)***' ?

Treball de laboratori:

L2.1 Amplieu el programa del projecte test per configurar una Interrupció del Timer1 a 100Hz.

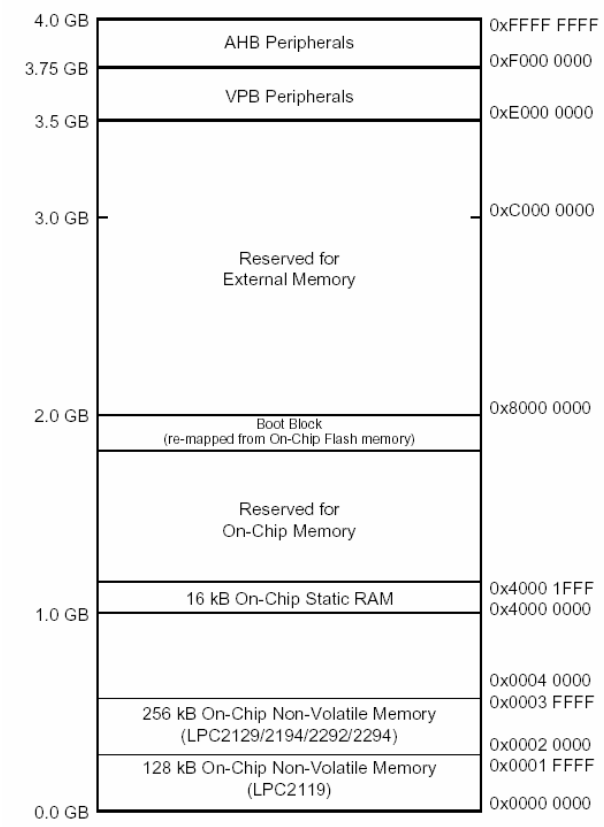
L2.2: Canvieu el codi de la secció anterior, L1.1, que controlava el display, per tal que el display sigui multiplexat des de la RSI del Timer0.

6. Cicles de bus externs al LPC2292

Objetius concrets:

- **Observar a l'analitzador lògic les formes d'ona dels cicles de bus a memòria externa del LPC2292.**
- **Programar diversos cicles d'espera i observar els resultats a l'analitzador lògic.**

El LPC2292 a l'igual que mols altres microcontroladors moderns disposa de memòria Flash i RAM internes, però també té un controlador de memòria externa (EMC, Capítol 3 del Manual d'usuari del LPC2292). Aquest controlador permet configurar fins a 4 bancs de memòria independents, cadascun de fins a 16Mbytes. El mapa de memòria general del microcontrolador és el següent:



Com podem veure hi ha 1.5Gbytes reservats per a memòria externa. Tot i així només es fan servir 4x16Mbytes amb el controlador de memòria externa:

Bank	Address Range	Configuration Register
0	8000 0000 - 80FF FFFF	BCFG0
1	8100 0000 - 81FF FFFF	BCFG1
2	8200 0000 - 82FF FFFF	BCFG2
3	8300 0000 - 83FF FFFF	BCFG3

Table 5: Address Ranges of External Memory Banks (LPC2292/2294only)

Els senyals de l'interface són:

Pin Name	Type	Pin Description
D[31:0]	Input/ Output	External memory data lines.
A[23:0]	Output	External memory address lines.
OE	Output	Low-active Output Enable signal.
BLS[3:0]	Output	Low-active Byte Lane Select signals.
WE	Output	Low-active Write Enable signal.
CS[3:0]	Output	Low-active Chip Select signals.

Table 6: External Memory Controller Pin Description

El LPC2292 disposa de 4 senyals de chip select (CS[3:0]) que es poden connectar directament als xips de memòria RAM/ROM. A més, cada CS[3:0] es correspon a un únic banc de memòria dels que sortien a la taula anterior (Table 5). Són controlats amb els següents registres:

Name	Description	Access	Reset Value (see Table 9)	Address
BCFG0	Configuration register for memory bank 0	Read/Write	0x0000 FBEF	0xFFE00000
BCFG1	Configuration register for memory bank 1	Read/Write	0x2000 FBEF	0xFFE00004
BCFG2	Configuration register for memory bank 2	Read/Write	0x1000 FBEF	0xFFE00008
BCFG3	Configuration register for memory bank 3	Read/Write	0x0000 FBEF	0xFFE0000C

Table 7: External Memory Controller Register Map

Cada registre BCFGn té el següent format:

BCFG0-3	Name	Function	Reset Value
3:0	IDCY	This field controls the minimum number of "idle" CCLK cycles that the EMC maintains between read and write accesses in this bank, and between an access in another bank and an access in this bank, to avoid bus contention between devices. The number of idle CCLK cycles between such accesses is the value in this field plus 1.	1111
4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9:5	WST1	This field controls the length of read accesses, except for subsequent reads from a burst ROM. The length of such read accesses, in CCLK cycles, is the value in this field plus 3.	11111
10	RBLE	This bit should be 0 for banks composed of byte-wide or non-byte-partitioned devices, so that the EMC drives the BLS3:0 lines High during read accesses. This bit should be 1 for banks composed of 16-bit and 32-bit wide devices that include byte select inputs, so that the EMC drives the BLS3:0 lines Low during read accesses.	0
15:11	WST2	For SRAM banks, this field controls the length of write accesses, which consist of: • one CCLK cycle of address setup with CS, BLS, and WE high, • (this value plus 1) CCLK cycles with address valid and CS, BLS, and WE low, and • one CCLK cycle with address valid, CS low, BLS and WE high. For burst ROM banks, this field controls the length of subsequent accesses, which are (this value plus 1) CCLK cycles long.	11111
16:23	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
24	BUSERR	The only known case in which this bit is set is if the EMC detects an AMBA request for more than 32 bits of data. The ARM7TDMI-S will not make such a request.	0
25	WPERR	This bit is set if software attempts to write to a bank that has the WP bit 1. Write a 1 to this bit to clear it.	0
26	WP	A 1 in this bit write-protects the bank.	0
27	BM	A 1 in this bit identifies a burst-ROM bank.	0
29:28	MW	This field controls the width of the data bus for this bank: 00=8 bit, 01=16 bit, 10=32 bit, 11=reserved	see Table 9
31:30	AT	Always write 00 to this field.	00

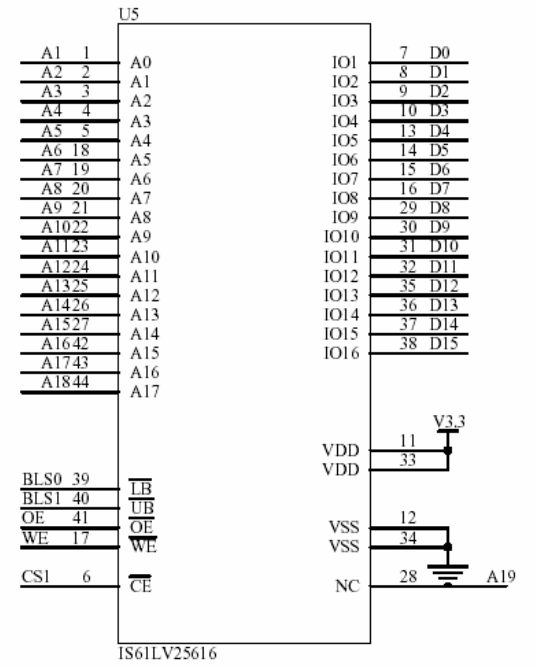
Table 8: Bank Configuration Registers 0-3 (BCFG0-3 - 0xFFE00000-0C)

Els bits que ens interessen són els RBLE, IDCY, WST1 i WST2:

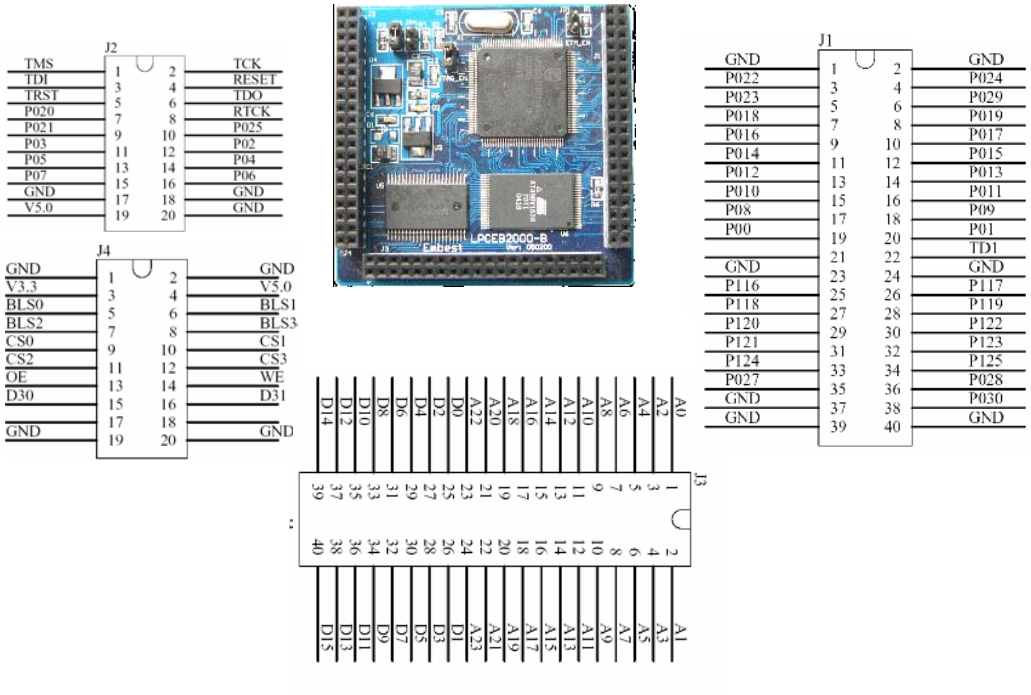
- RBLE: Veient les connexions a la RAM i el datasheet de la RAM, aquest bit ha de ser igual a '1'. Per què?
- Els bits IDCY (idle cycles) permeten especificar el nombre mínim de cicles de rellotge entre accessos de bus consecutius al mateix banc o a altres. Per evitar el problema de '*bus contention'*, col·lisió al bus de dades entre cicles de bus consecutius, s'ha d'especificar el nombre mínim de cicles de rellotge per tal que el problema no es doni.

- Els bits WST1 i WST2 permeten especificar els cicles d'espera en lectura i escriptura respectivament.

A la placa LPCEB2000-B hi ha un xip de memòria flash i un altre de RAM. Les connexions al xip de RAM es poden veure a la següent figura:



Com podem veure és un xip de 512Kbytes de capacitat total, en el format 256Kx16. Els pins de la placa LPCEB2000-B són totalment accessibles i són els següents:



Els exemples típics de cicle de bus que ens dóna el fabricant del LPC2292 són els següents (cal tenir en compte que el comportament de BLS0* i BLS1* en lectura pot ser diferent del que surt als cronogrames, depenent del bit RBLE del registre BCFG1):

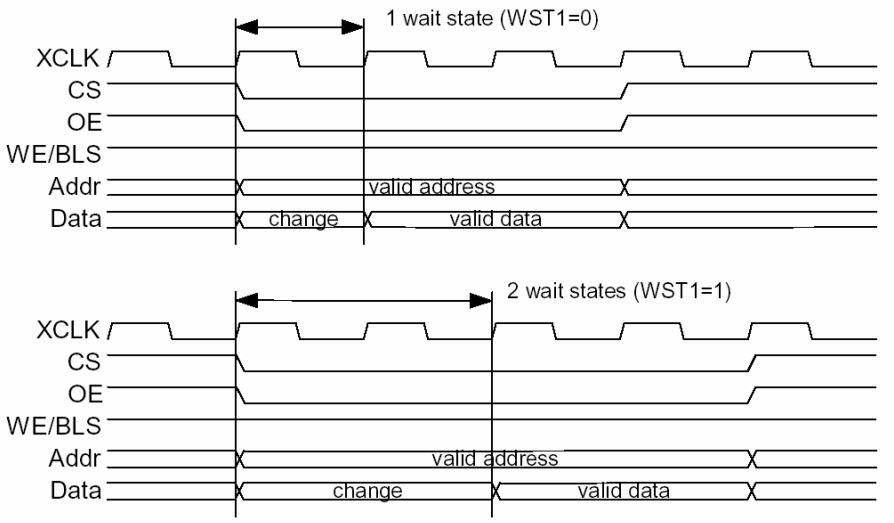


Figure 10: External memory read access (WST1=0 and WST1=1 examples)

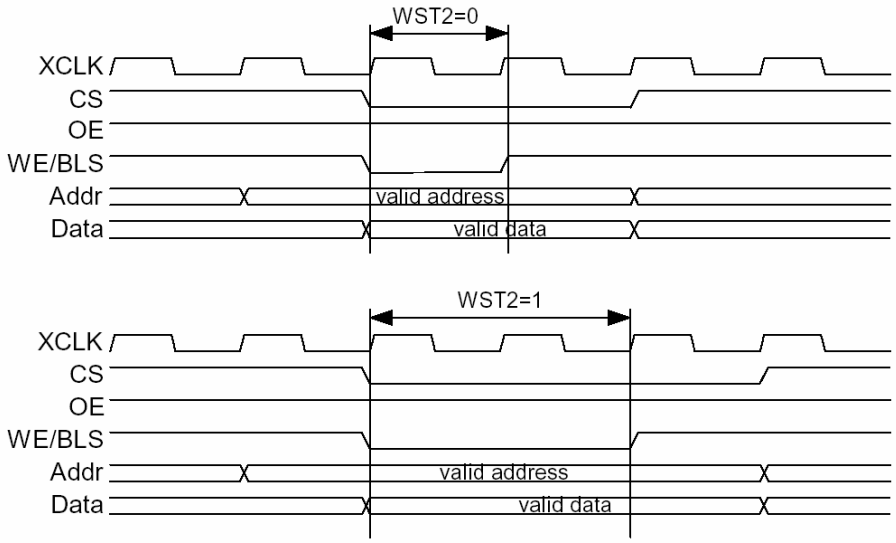
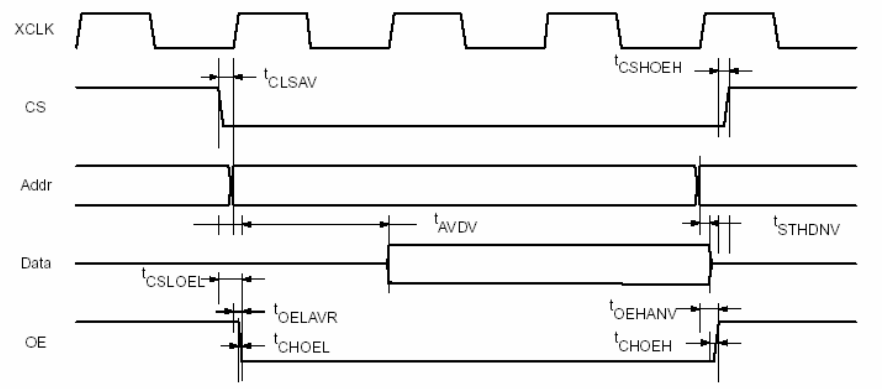
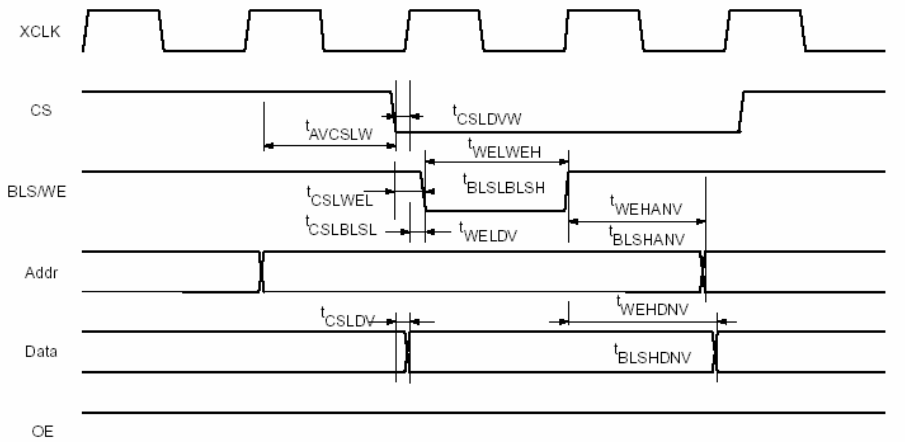


Figure 11: External memory write access (WST2=0 and WST2=1 examples)

La informació més detallada dels temps del microcontrolador es pot trobar al datasheet del LPC2292:



Cicle de lectura



Cicle d'escriptura

Symbol	Description	Min	Max	Unit
Read Cycle Parameters				
t_{CSLAV}	CS LOW to Address Valid	–5[1]	10	ns
t_{OELAVR}	OE LOW to Address Valid	–5[1]	10	ns
t_{CSLOEL}	CS LOW to OE LOW	–5	5	ns
t_{AVDV}	Memory Access Time (latest of Address Valid, CS LOW, OE LOW to Data Valid)	(t_{CYC}*(2 + WST1)) + (–20)	-	ns
t_{AVDV}	Burst-ROM Initial Memory Access Time (latest of Address Valid, CS LOW, OE LOW to Data Valid)	(t_{CYC}*(2 + WST1)) + (–20)	-	ns
t_{AVDV}	Burst-ROM Subsequent Memory Access Time (Address Valid to Data Valid)	t_{CYC} + (–20)	-	ns
t_{STHDNV}	Data Hold Time (earliest of CS HIGH, OE HIGH, Address Change to Data Invalid)	0	-	ns
t_{CSHOEH}	CS HIGH to OE HIGH	–5	5	ns
t_{OEHANV}	OE HIGH to Address Invalid	–5	5	ns
t_{CHOEL}	XCLK HIGH to OE LOW	–5	5	ns
t_{CHOEH}	XCLK HIGH to OE HIGH	–5	5	ns
Symbol	**Description**	**Min**	**Max**	**Unit**
Write Cycle Parameters				
t_{AVCSLW}	Address Valid to CS LOW	t_{CYC} – 10 [1]	-	ns
t_{CSLDVW}	CS LOW to Data Valid	–5	5	ns
t_{CSLWEL}	CS LOW to WE LOW	–5	5	ns
$t_{CSLBLSL}$	CS LOW to BLS LOW	–5	5	ns
t_{WELDV}	WE LOW to Data Valid	–5	5	ns
t_{CSLDV}	CS LOW to Data Valid	–5	5	ns
t_{WELWEH}	WE LOW to WE HIGH	$t_{CYC} \times (1 + WST2) – 5$	t_{CYC}*(1 + WST2) + 5	ns
t_{WELWEH}	BLS LOW to BLS HIGH	$t_{CYC} \times (1 + WST2) – 5$	t_{CYC}*(1 + WST2) + 5	ns
t_{WEHANV}	WE HIGH to Address Invalid	t_{CYC}–5	t_{CYC} + 5	ns
t_{WEHDNV}	WE HIGH to Data Invalid	$(2 \times t_{CYC})$–5	$(2 \times t_{CYC})$ + 5	ns
$t_{BLSHANV}$	BLS HIGH to Address Invalid	t_{CYC}–5	t_{CYC} + 5	ns
$t_{BLSHDNV}$	BLS HIGH to Data Invalid	$(2 \times t_{CYC})$–5	$(2 \times t_{CYC})$ + 5	ns
t_{CHDV}	XCLK HIGH to Data Valid	-	10	ns
t_{CHWEL}	XCLK HIGH to WE LOW	-	10	ns
$t_{CHHBLSL}$	XCLK HIGH to BLS LOW	-	10	ns
t_{AVCSL}	XCLK HIGH to WE HIGH	-	10	ns
t_{AVCSL}	XCLK HIGH to BLS HIGH	-	10	ns
t_{AVCSL}	XCLK HIGH to Data Invalid	-	10	ns

Rellotge del sistema (OPCIONAL)

El LPC2292 disposa d'un rellotge programable, *cclk*, que es pot generar a partir del cristall de quars extern, multiplicant la seva freqüència per un número fins un màxim de 60MHz. La freqüència del cristall de quars a la placa és de 10MHz.

Fins ara no hem tocat *cclk*, que era de 50MHz. Els perifèrics interns (timers, etc.) funcionen amb un altre rellotge, el *pclk*: peripheral clk, que fins ara també tenia la mateixa freqüència (50MHz).

El LPC2292 ens ofereix la possibilitat de veure el rellotge intern del microprocessador pel pin A23 (el pin 23 d'adreces) que es troba al Port 3: P3.23. Aquest pin pot ser bit extern d'adreces o treure un rellotge *XCLK*, que s'obté a partir del rellotge *pclk*.

La següent rutina, la podeu afegir al vostre programa per fixar que el rellotge *XCLK*, que es veu pel pin A23, sigui el rellotge del microprocessador, *CCLK*, i el rellotge dels perifèrics, *PCLK*, igual a 50 MHz.

```
void initPLL(void)
{
        // A aquesta rutina fixem que el cclk (processor clock) = 50MHz
        // i que el pclk (peripheral clock) sigui 50MHz
        // i que per A23 surti XCLK=CCLK


    unsigned int M=6;
    unsigned int P=2;

    *PLLCON=0x1;                          // PLL enable
    *PLLCFG=((P-1)<<5)+(M-1);             // Fixem que cclk = 10MHz * M < 60MHz
                                          // Ha de ser 156MHz<fcco=10MHz*2*M*P<320MHz
    *PLLFEED=0xaa;
    *PLLFEED=0x55;

    while (!(*PLLSTAT&(B10)));         // Que fa aquesta instrucció ?

    *PLLCON=0x3;                          // El PLL es el cclk del lpc2292
    *PLLFEED=0xaa;
    *PLLFEED=0x55;


     *VPBDIV=P4+0x01;                     // xclk = pclk te la mateixa freq. que cclk
     *PINSEL2&=~P25;*PINSEL2|=P13;        // Fem que el pin P3.23 no sigui A23 sino XCLK (p. 120)

}
```

Així és molt senzill canviar la freqüència del microprocessador (canviant els valors de P i M, respectant els límits que podeu veure als comentaris de la rutina). Podeu mirar tots els detalls dels diferents rellotges del sistema al Capítol 4 (PLL) del Manual d'usuari del LPC2292 (p. 72-77).

NOTA: El Manual d'usuari del LPC2292 és una mica confús respecte als valors que cal posar de M i P al registre PLLCFG. Pareu atenció dels comentaris que hi ha a la pàgina 77 del manual. És per aquesta raó que es posa P-1 i M-1 al registre PLLCFG.

Els valors de P i M de la rutina anterior es poden canviar, però cal ser curós perquè fàcilment ens equivocarem. Podeu posar una freqüència, p.e., de 30MHz, 40MHz o 60MHz.

Estudi Previ

EP7.1 A partir de la informació del datasheet de la memòria RAM IS61LV25616-10 (IS61LV25616.pdf) i la informació del LPC2292, calculeu el nombre mínim d'estats d'espera en lectura. Freq. de treball del LPC2292: 50MHz. Nombre de zones imatge del xip de memòria RAM al mapa de memòria ?

EP7.2 A partir de la informació del mateix datasheet i del LPC2292 calculeu el nombre mínim amb que cal configurar els cicles d'espera en escriptura.

EP7.3 A partir de la informació d'aquest apartat i la del fitxer olimex_lpce2294.ram del projecte, a on es carrega el nostre programa quan l'executem amb l'Eclipse ?

EP7.3 (OPCIONAL) Calculeu el nombre mínim d'estats IDCY (cicles de rellotge entre cicles de bus per evitar el problema de '*bus contention'*).

Treball a realitzar al laboratori

L7.1- Connectarem a l'analitzador lògic els següents senyals:

- A23 (pin pel que es veu XCLK) (OPCIONAL)
- A1:3
- CS1
- BLS0:1
- OE
- WE
- D11:15 (o altres bits de bus de dades)
- P0.18 o P0.19 (relacionat amb l'actualització dels displays de 7 segments)

L7.2- Quan depurem a les pràctiques, i llevat de que canviem els fitxers de configuració, el programa es carrega a la RAM externa, i les seves dades també s'emmagatzemen al mateix xip. Tenint en compte això:

2.1 A partir de mesures, doneu la màxima informació possible dels temps d'accés en lectura REALS del xip de memòria RAM.

2.2 Compareu aquests resultats amb la informació del corresponent datasheet: IS61LV25616.pdf

2.3 Captureu cicles burst. Quina diferència hi ha entre cicles burst en lectura i escriptura ? Per què ?

2.4 Verifiqueu la freqüència d'interrupció del Timer que controla els displays de 7 segments, mirant al pin P0.19 (o el P0.18).

L7.3- Examineu l'efecte de programar diversos cicles d'espera en lectura (WST1) i escriptura (WST2), així com diferents valors de cicles de rellotge IDCY.

L7.4- (OPCIONAL junt amb EP7.4) Programeu diferents freqüències de rellotge del microcontrolador (modificant la configuració del PLL). Observeu els resultats a l'analitzador lògic.

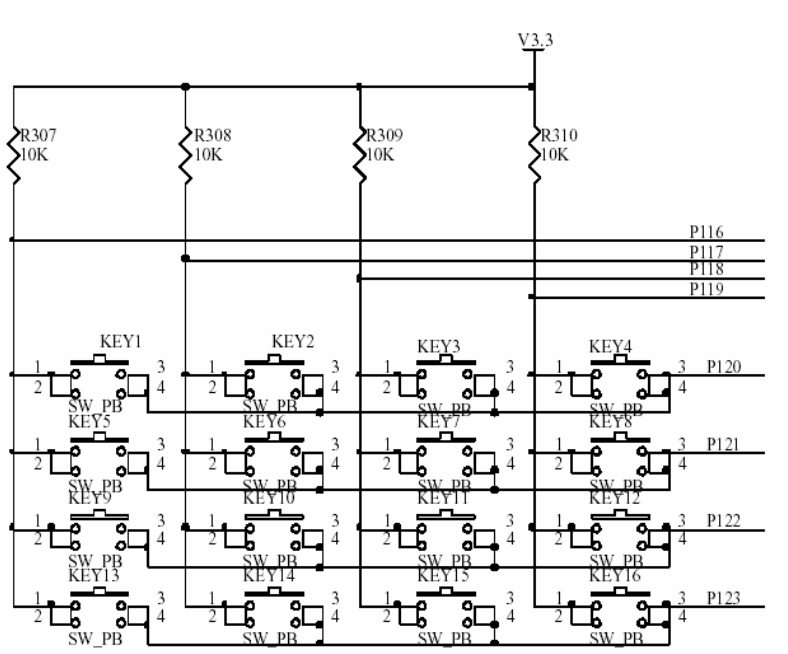
NOTA: Documenteu els resultats amb captures de la pantalla de l'analitzador lògic.

APARTATS OPCIONALS

7. Teclat (OPCIONAL)

Objectius concrets: controlar el teclat.

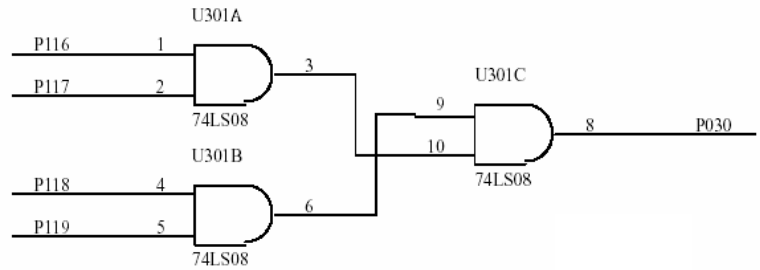
A la següent figura podem veure l'esquema de connexió del teclat a la placa LPCEB2000-I:



Suposem que els pins P1.20- P1.23 els hem configurat com a GPIO-sortides, els pins P1.16-P1.19 son GPIO-entrades, i apliquem els valors P1.20='0', P1.21=P12.2=P1.23='1'. En cas de pitjar una de les tecles a la primera fila (la indexada per P1.20) veurem un '0' al corresponent pin P1.16-P1.19.

Si en comptes d'això últim, activem el pin P1.21 (és a dir, P1.20=P1.22=P1.23=1 i P1.21=0) podrem veure si tenim alguna tecla pitjada a la segona fila (KEY5-KEY8), i així successivament. Normalment el que es fa és escombrar contínuament els pins P120-P123, activant en tot moment com a màxim un d'aquests, i mirar als pins P1.16:P1.19 les tecles que s'han pitjat.

Aquesta estratègia consumeix recursos de computació de manera continuada perquè cal examinar el teclat periòdicament, independentment de que alguna tecla estigui pitjada o no. Per evitar aquest problema podem fer servir interrupcions. Aprofitarem el següent circuit que es troba a la placa LPCEB2000-I:



Com podem veure, si hi ha un '0' a qualsevol dels pins P1.16-P1.19 el resultat és un '0' al pin P0.30. Noteu que els primers quatre pins són al Port 1 i el pin P0.30 és del Port 0. El pin P0.30 es pot configurar per generar interrupcions externes (EINT3).

Per tant, es poden proposar dues estratègies diferents de control del teclat:

1- Escombrat periòdic continu dels pins P1.20-P1.23, llegint els pins P1.16-P1.19. La periodicitat pot venir marcada pels Timers (Timer0 o Timer1).

2- Posar a '0' els pins P1.20-P1.23 i activar interrupcions externes (EINT3) del P0.30. En cas de produir-se una interrupció fer un escombrat complet, a la mateixa RSI, i mirar quina(es) tecla(es) s'han pitjat.

Per simplificar la RSI es pot fer que el pin P0.30 interrompi per flanc de baixada. Un cop estiguem a la RSI canviem a interrupció per flanc de pujada, i així alternadament cada cop que entrem a la RSI. Això implica que si premem una tecla hi haurà dues interrupcions (una quan comencem a prémer la tecla i una altra quan deixem de fer-ho).

EP3.1: Quins avantatges/desavantatges presenten comparativament ambdues estratègies ? (hi ha punts positius i negatius per a totes dues). Podeu proposar una altra ?

La informació per programar les interrupcions externes es troba a: Capítol 4. System Control Block. External Interrupt Inputs. El registre PINSEL2 té la següent forma:

Table 64: Pin Function Select Register 2 for LPC2292/2294 (PINSEL2 - 0xE002C014)

PINSEL2	Description	Reset Value
1:0	Reserved.	00
2	When 0, pins P1.36:26 are used as GPIO pins. When 1, P1.31:26 are used as a Debug port.	$\overline{\text{P1.26}}$/RTCK
3	When 0, pins P1.25:16 are used as GPIO pins. When 1, P1.25:16 are used as a Trace port.	$\overline{\text{P1.20}}$/ TRACESYNC
5:4	Controls the use of the data bus and strobe pins: Pins P2.7:0 — 11 = P2.7:0 — 0x or 10 = D7:0 Pin P1.0 — 11 = P1.0 — 0x or 10 = CS0 Pin P1.1 — 11 = P1.1 — 0x or 10 = OE Pin P3.31 — 11 = P3.31 — 0x or 10 = BLS0 Pins P2.15:8 — 00 or 11 = P2.15:8 — 01 or 10 = D15:8 Pin P3.30 — 00 or 11 = P3.30 — 01 or 10 = BLS1 Pins P2.27:16 — 0x or 11 = P2.27:16 — 10 = D27:16 Pins P2.29:28 — 0x or 11 = P2.29:28 or Reserved — 10 = D29:28 Pins P2.31:30 — 0x or 11 = P2.31:30 or AIN5:4 — 10 = D31:30 Pins P3.29:28 — 0x or 11 = P3.29:28 or AIN6:7 — 10 = BLS2:3	BOOT1:0
6	If bits 5:4 are not 10, controls the use of pin P3.29: 0 enables P3.29, 1 enables AIN6.	1
7	If bits 5:4 are not 10, controls the use of pin P3.28: 0 enables P3.28, 1 enables AIN7.	1
8	Controls the use of pin P3.27: 0 enables P3.27, 1 enables WE.	0
10:9	Reserved.	-
11	Controls the use of pin P3.26: 0 enables P3.26, 1 enables CS1.	0
12	Reserved.	-
13	If bits 25:23 are not 111, controls the use of pin P3.23/A23/XCLK: 0 enables P3.23, 1 enables XCLK.	0
15:14	Controls the use of pin P3.25: 00 enables P3.25, 01 enables CS2, 10 and 11 are reserved values.	00
17:16	Controls the use of pin P3.24: 00 enables P3.24, 01 enables CS3, 10 and 11 are reserved values.	00
19:18	Reserved.	-
20	If bits 5:4 are not 10, controls the use of pin P2.29:28: 0 enables P2.29:28, 1 is . **Warning:** it is possible to select 2 pins for RD6, by setting bits 5:4 to 0x or 11, bits 15;14 to 11 and bit 20 to 1. This situation could be called a programming error. In this case, bit 20 predominatets and RD6 is taken from P2.28.	0
21	If bits 5:4 are not 10, controls the use of pin P2.30: 0 enables P2.30, 1 enables AIN4.	1
22	If bits 5:4 are not 10, controls the use of pin P2.31: 0 enables P2.31, 1 enables AIN5.	1

Table 64: Pin Function Select Register 2 for LPC2292/2294 (PINSEL2 - 0xE002C014)

PINSEL2	Description	Reset Value
23	Controls whether P3.0/A0 is a port pin (0) or an address line (1).	1 if BOOT1:0=00 at $\overline{\text{RESET}}$=0, 0 otherwise
24	Controls whether P3.1/A1 is a port pin (0) or an address line (1).	$\overline{\text{BOOT1}}$ during Reset
27:25	Controls the number of pins among P3.23/A23/XCLK and P3.22:2/A2.22:2 that are address lines: 000 = None 001 = A3:2 are address lines. 010 = A5:2 are address lines. 011 = A7:2 are address lines. 100 = A11:2 are address lines. 101 = A15:2 are address lines. 110 = A19:2 are address lines. 111 = A23:2 are address lines.	000 if BOOT1:0=11 at Reset, 111 otherwise
31:28	Reserved.	-

Només ens queden els registres EXTMODE i EXTPOLAR, per configurar els pins EINT0:3 per generar interrupcions per flancs o nivell (EXTMODE), o per flanc de pujada o baixada (EXTPOLAR).

Table 17: External Interrupt Mode Register (EXTMODE - 0xE01FC148)

EXTMODE	Function	Description	Reset Value
0	EXTMODE0	When 0, level-sensitivity is selected for EINT0. When 1, EINT0 is edge-sensitive.	0
1	EXTMODE1	When 0, level-sensitivity is selected for EINT1. When 1, EINT1 is edge-sensitive.	0
2	EXTMODE2	When 0, level-sensitivity is selected for EINT2. When 1, EINT2 is edge-sensitive.	0
3	EXTMODE3	When 0, level-sensitivity is selected for EINT3. When 1, EINT3 is edge-sensitive.	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

External Interrupt Polarity Register (EXTPOLAR - 0xE01FC14C)

In level-sensitive mode, the bits in this register select whether the corresponding pin is high- or low-active. In edge-sensitive mode, they select whether the pin is rising- or falling-edge sensitive. Only pins that are selected for the EINT function (chapter Pin Connect Block on page 116) and enabled in the VICIntEnable register (chapter Vectored Interrupt Controller (VIC) on page 92) can cause interrupts from the External Interrupt function (though of course pins selected for other functions may cause interrupts from those functions).

Note: Software should only change a bit in this register when its interrupt is disabled in VICIntEnable, and should write the corresponding 1 to EXTINT before re-enabling the interrupt, to clear the EXTINT bit that could be set by changing the polarity.

Table 18: External Interrupt Polarity Register (EXTPOLAR - 0xE01FC14C)

EXTPOLAR	Function	Description	Reset Value
0	EXTPOLAR0	When 0, EINT0 is low-active or falling-edge sensitive (depending on EXTMODE0). When 1, EINT0 is high-active or rising-edge sensitive (depending on EXTMODE0).	0
1	EXTPOLAR1	When 0, EINT1 is low-active or falling-edge sensitive (depending on EXTMODE1). When 1, EINT1 is high-active or rising-edge sensitive (depending on EXTMODE1).	0
2	EXTPOLAR2	When 0, EINT2 is low-active or falling-edge sensitive (depending on EXTMODE2). When 1, EINT2 is high-active or rising-edge sensitive (depending on EXTMODE2).	0
3	EXTPOLAR3	When 0, EINT3 is low-active or falling-edge sensitive (depending on EXTMODE3). When 1, EINT3 is high-active or rising-edge sensitive (depending on EXTMODE3).	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Finalment, només ens queda el registre EXTINT, que és a on podem mirar si hi ha alguna interrupció pendent externa (EINT0:3). També a la RSI, s'ha de posar a '1' el bit que correspongui a la interrupció per tal d'anular la petició prèvia que ha generat que s'executi la RSI.

A la RSI que farem per a EINT3, tenint en compte que la programem per flancs, posarem a '1' el bit 3 de EXTINT, abans de sortir de la rutina.

Table 15: External Interrupt Flag Register (EXTINT - 0xE01FC140)

EXTINT	Function	Description	Reset Value
0	EINT0	In level-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT0 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	0
1	EINT1	In level-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT1 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	0
2	EINT2	In level-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT2 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	0
3	EINT3	In level-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the pin is in its active state. In edge-sensitive mode, this bit is set if the EINT3 function is selected for its pin, and the selected edge occurs on the pin. This bit is cleared by writing a one to it, except in level sensitive mode when the pin is in its active state.	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Exemple detecció tecla pitjada per interrupcions

Aquestes rutines configuren el teclat per tal que es generi una interrupció externa cada cop que es prem una tecla, o es deixa de prèmer. A tal efecte, programem a initeclat() el pin P0.30 per generar interrupcions per flanc de baixada. Un cop a dins la RSI, canviem sempre la polaritat de la interrupció (a flanc de baixada o pujada segons convingui).

```
extern void SPI0_VectISR(void);

unsigned int polar=0;

void P030_EINT3_ISR(void)
{
  *VICIntEnClr|=B17;                    // Deshabilitem EINT3

  if (!polar)
        *EXTPOLAR=polar=B3;
      else *EXTPOLAR=polar=0;           // Canviem polaritat EINT3

  *EXTINT|=P3;                          // Anulem qualsevol int. pendent
  *VICVectAddr = 0x00;                  // End of Interrupt cap el VIC
  *VICIntEnable|=B17;                   // Tornem a habilitar EINT3

}
void initeclat(void)
{

   *PINSEL2&=~(P3);                     // Pins P1.25:16 son GPIO
   *IODIR1=P20+P21+P22+P23;             // Pins P1.20:23 son sortides.
                                        // La resta entrades (dins de P1.16:25)

   *IOCLR1=P20+P21+P22+P23;             // Posem a 0 els bits P1.20-P1.23

   *PINSEL1&=~B28;                      // Pin P0.30 es EINT3
   *PINSEL1|=B29;
   *EXTMODE=B3;                         // EINT3 es edge sensitive.
   *EXTPOLAR=0;                         // EINT3 es falling-edge
   *EXTINT|=P3;                         // Anulem qualsevol int. pendent

   *VICIntSelect&=~B17;                 // EINT3 es IRQ (i no FIQ)
   *VICVectAddr2 = (int)EINT3_VectISR;
   *VICVectCntl2 = B5 + 17;             // Fem servir VIC Channel 2
   *VICIntEnable|=B17;
}
```

afegint a Interrupt.s:

```
IRQHandle P030_EINT3_ISR,EINT3_VectISR
```

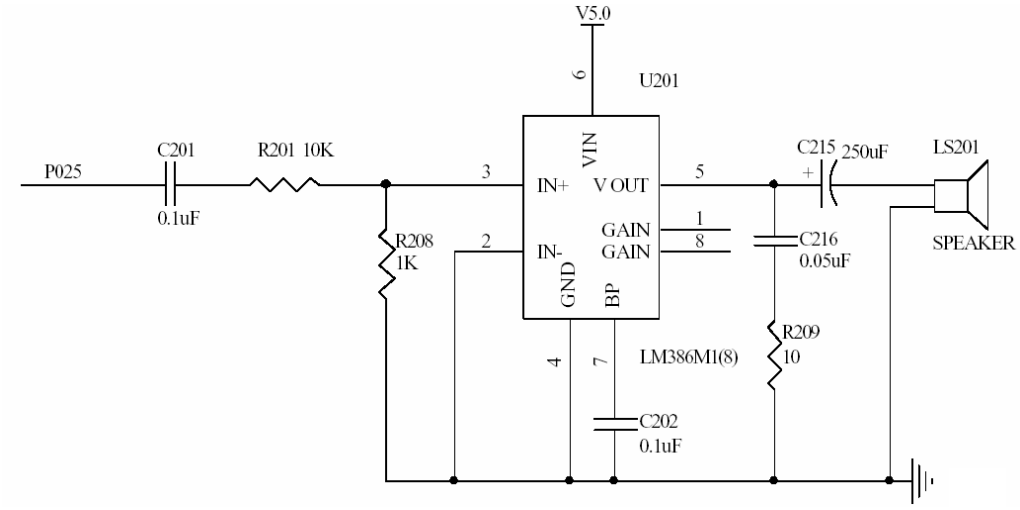
Treball de laboratori:

L3.1 Trieu una de les estratègies per a controlar el teclat que hem explicat. Feu el codi corresponent.

L3.2 Modifiqueu el vostre programa per tal que si es prem una tecla determinada surti un missatge diferent al display.

8. Altaveu (OPCIONAL)

A la següent figura podem veure les connexions de l'altaveu que hi ha a la placa LPCEB2000-I.



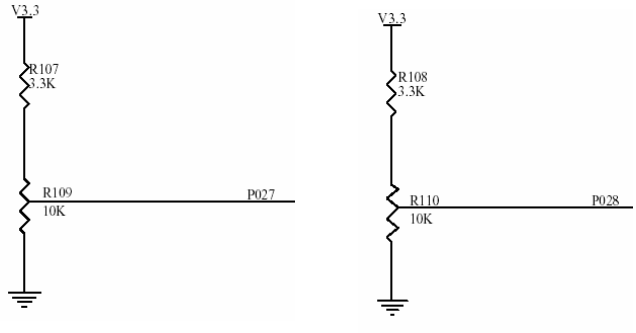
Com podem observar l'altaveu està controlat pel pin P0.25 el qual està connectat a l'entrada d'un petit amplificador d'àudio. Per tant, podem generar ones quadrades a aquest pin i sentirem el to corresponent a l'altaveu.

L4.1 Modifiqueu el vostre programa per tal que, en prémer una tecla determinada, es senti un to a l'altaveu. Podeu utilitzar la RSI del Timer1 (es pot fer una 'sirena' canviant la freqüència d'interrupció del timer a dins la seva RSI). També podeu cercar a Internet com fer una escala cromàtica (do, re, mi, etc.). Eventualment, podeu fixar la durada de les notes amb el valor d'un dels potenciòmetres, si feu el següent apartat.

9. Convertidors A/D (OPCIONAL)

Objetius concrets: controlar per interrupcions els convertidors A/D.

El LPC2292 disposa de un convertidor A/D de 10 bits que permet multiplexar fins a 8 pins. Podem fer una conversió A/D d'un qualsevol d'aquests pins en un moment donat. A la placa LPCEB2000-I s'han col·locat 2 potenciòmetres de la següent manera:



Els pins P0.27 i P0.28 poden ser configurats com a entrades analògiques AIN0 i AIN1, respectivament (amb en registre *PINSEL1). El convertidor té dos registres de control (Capítol 17 del Manual d'usuari del LPC2292): *ADCR (A/D Control Register) i *ADDR (A/D Data Register). Farem servir el *hardware scan mode*.

A/D Control Register (ADCR - 0xE0034000)

Table 174: A/D Control Register (ADCR - 0xE0034000)

ADCR	Name	Description	Reset Value
7:0	SEL	Selects which of the Ain3:0 (LPC2119/2129/2194) or Ain7:0 (LPC2292/2294) pins is (are) to be sampled and converted. Only bits 3:0 should be set to 1 in the 48 or 64 pin package. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones (1 to 4 ones in the 48 or 64 pin package) can be used. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV	The VPB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST	If this bit is 0, conversions are software controlled and require 11 clocks. If this bit is 1, the AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed..	0
19:17	CLKS	This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the LS bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits): 000=11 clocks/10 bits, 001=10 clocks/9 bits,..., 111=4 clocks/3 bits	000
21	PDN	1: the A/D converter is operational 0: the A/D converter is in power down mode	0
23:22	TEST1:0	These bits are used in device testing. 00=normal operation, 01=digital test mode, 10=DAC test mode, and 11=simple conversion test mode.	0
26:24	START	When the BURST bit is 0, these bits control whether and when an A/D conversion is started: 000: no start (this value should be used when clearing PDN to 0) 001: start conversion now 010: start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/ CAP0.2 011: start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/MAT0.0 *Note: for choices 100-111 the MAT signal need not be pinned out:* 100: start conversion when the edge selected by bit 27 occurs on MAT0.1 101: start conversion when the edge selected by bit 27 occurs on MAT0.3 110: start conversion when the edge selected by bit 27 occurs on MAT1.0 111: start conversion when the edge selected by bit 27 occurs on MAT1.1	000
27	EDGE	This bit is significant only when the START field contains 010-111. In these cases: 0: start conversion on a falling edge on the selected CAP/MAT signal 1: start conversion on a rising edge on the selected CAP/MAT signal	0

L'objectiu que tenim és convertir de manera continuada les dues tensions proporcionades pels potenciòmetres. Per tal de no estar convertint en tot moment les tensions analògiques, el que farem és utilitzar el Timer0 i el seu registre *Match Register 1* per disparar les conversions. Això es pot fer automàticament configurant el mode *START=100*.

Consulteu els detalls al Manual d'usuari del LPC2292

El funcionament que volem és:

> Cada cop que el comptador del Timer0 arribi a un cert valor que definirem al registre TIMER0_MR1 (que haurà de ser més petit que el valor definit al registre TIMER0_MR0), començarà una conversió del pin P0.27 o P0.28. Les conversions aniran alternant-se i al final de cadascuna es generarà una interrupció.

EP5.1: Per què ha de ser més petit el valor que posem a TIMER0_MR1 que el de TIMER0_MR0 ? Cal tenir en compte el programa que hem fet a la secció 5.

El registre de dades, *ADDR, té el format següent:

A/D Data Register (ADDR - 0xE0034004)

ADDR	Name	Description	Reset Value
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0
30	OVERUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the LS bits. In non-FIFO operation, this bit is cleared by reading this register.	0
29:27		These bits always read as zeroes. They could be used for expansion of the CHN field in future compatible A/D converters that can convert more channels.	0
26:24	CHN	These bits contain the channel from which the LS bits were converted.	X
23:16		These bits always read as zeroes. They allow accumulation of successive A/D values without AND-masking, for at least 256 values without overflow into the CHN field.	0
15:6	V/VddA	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the VddA pin. Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on VssA, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on VddA. For testing, data written to this field is captured in a shift register that is clocked by the A/D converter clock. The MS bit of this register sources the DINSERI input of the A/D converter, which is used only when TEST1:0 are 10.	X
5:0		These bits always read as zeroes. They provide compatible expansion room for future, higher-resolution A/D converters.	0

Table 175: A/D Data Register (ADDR - 0xE0034004)

Els bits importants d'aquest registre són els CHN, que indiquen quin canal ha estat convertit, i el 15:6 V/VddA que ens donen la conversió digital de la tensió. Aquest registre el llegirem a la RSI del convertidor.

NOTES:

- El rellotge del convertidor s'obté dividint la freqüència de PCLK (15 MHz) per CLKDIV+1 (registre *ADCR). Aquesta freqüència sempre ha de ser menor que 4.5 MHz.
- NO seleccioneu el mode BURST, perquè està relacionat amb un dels 'bugs' del LPC2292 i dóna problemes.
- Feu START=100 i no oblideu fer el EOI del VIC (escriure 0 a VICVectAddr), a la RSI, al final.

Treball al laboratori:

> **L5.1 Modifiqueu el vostre programa per tal que hi hagi conversions periòdiques dels pins P0.27 i P0.28. Caldrà modificar la rutina initimer0, per incoporar el registre de Match 1: TIMER0_MR1.**

La visualització del valor convertit la farem a la següent secció. En cas de no fer el següent apartat representeu les conversions A/D en hexadecimal als displays.

10. Bus SPI: Serial Peripheral Interface (OPCIONAL)

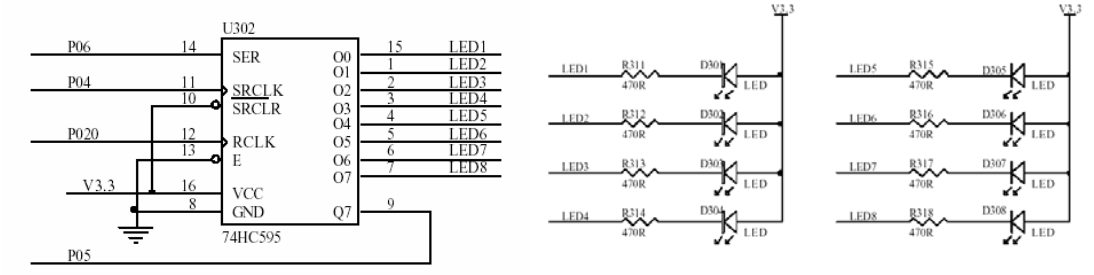
Objetius concrets:

- **Conèixer el bus SPI**
- **Programar-lo per interrupcions per a transmetre dades de manera contínua als leds.**

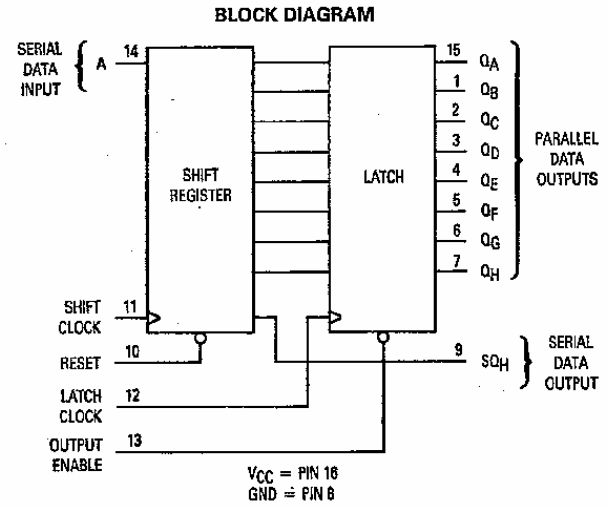
El LPC2292 disposa de dos controladors independents de bus SPI. Aquest bus està dissenyat per enviar i rebre dades de perifèrics en format sèrie síncron. A cada bus hi ha un Master i un o més Slaves. El bus està format per 4 senyals:

- SCK: Serial clock
- SSEL: Selecció de Esclau. Es una entrada dels esclaus que el Master utilitza per dir amb quin esclau vol establir comunicació
- MISO: Master In Slave Out: senyal unidireccional sèrie pel que l'esclau transmet la informació al Master
- MOSI: Master Out Slave In: senyal unidireccional sèrie pel que el Master envia informació a l'esclau.

Les connexions a la placa LPCEB2000-I relacionades amb el bus SPI es poden veure a la següent figura:



En el nostre cas, l'esclau és un simple registre de desplaçament que té les sortides connectades a leds. L'esquema intern del 74HC595 es pot veure a la següent figura:



Aquest component:

- A cada flanc de pujada del clock sèrie (SRCLK o SHIFT CLOCK) desplaça els bits del registre de desplaçament (shift register), i el valor a l'entrada (SER o SERIAL DATA INPUT) és introduit al registre.
- Quan hi ha un flanc de pujada al LATCH CLOCK (o RCLK) el contingut del registre de desplaçament passa al LATCH, i per tant té efectes sobre els leds.

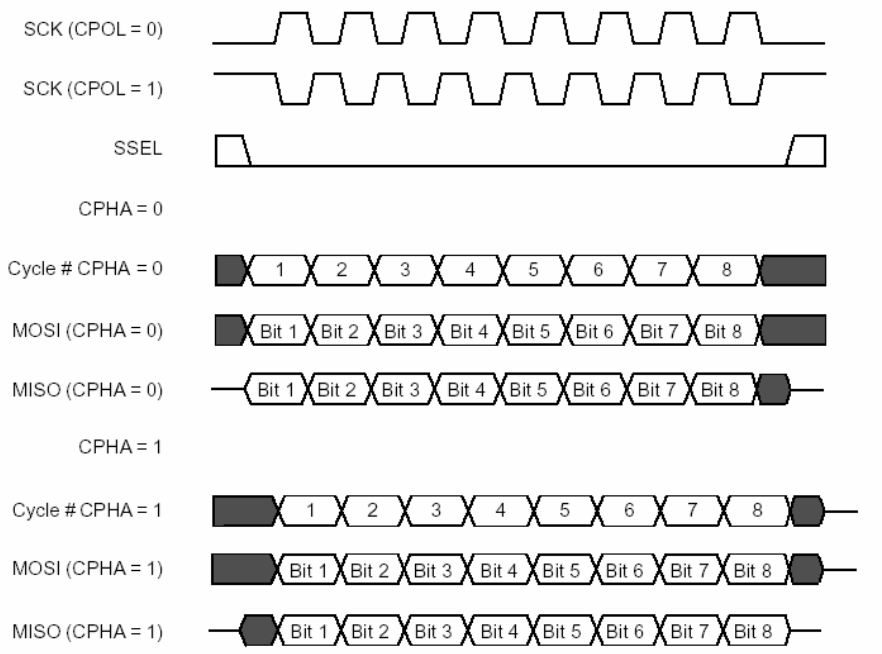
El controlador SPI0 del LPC2292 es controla mitjançant els següents registres: SPI0_SPCR i SPI0_SPCCR.

Table 116: SPI Control Register (S0SPCR - 0xE0020000, S1SPCR - 0xE0030000)

SPCR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	CPHA	Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending. When 1, data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active. When 0, data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	0
4	CPOL	Clock polarity control. When 1, SCK is active low. When 0, SCK is active high.	0
5	MSTR	Master mode select. When 1, the SPI operates in Master mode. When 0, the SPI operates in Slave mode.	0
6	LSBF	LSB First controls which direction each byte is shifted when transferred. When 1, SPI data is transferred LSB (bit 0) first. When 0, SPI data is transferred MSB (bit 7) first.	0
7	SPIE	Serial peripheral interrupt enable. When 1, a hardware interrupt is generated each time the SPIF or MODF bits are activated. When 0, SPI interrupts are inhibited.	0

NOTA: l'etiqueta definida a LPC_Base.h és `SPI0_SPCR`.

La relació entre el rellotge i la línia de dades és la següent:



Un resum es pot veure a la següent taula:

CPOL And CPHA Settings	First Data Driven	Other Data Driven
CPOL = 0, CPHA = 0	Prior to first SCK rising edge	SCK falling edge
CPOL = 0, CPHA = 1	First SCK rising edge	SCK rising edge
CPOL = 1, CPHA = 0	Prior to first SCK falling edge	SCK rising edge
CPOL = 1, CPHA = 1	First SCK falling edge	SCK falling edge

NOTA: Al nostre registre de desplaçament reb dades per flanc de pujada del rellotge sèrie. Per tant ens interessen les combinacions CPOL=CPHA=0 ó CPOL=CPHA=1.

SPI Clock Counter Register (S0SPCCR - 0xE002000C, S1SPCCR - 0xE003000C)

This register controls the frequency of a master's SCK. The register indicates the number of pclk cycles that make up an SPI clock. The value of this register must always be an even number. As a result, bit 0 must always be 0. The value of the register must also always be greater than or equal to 8. Violations of this can result in unpredictable behavior.

Table 119: SPI Clock Counter Register (S0SPCCR - 0xE002000C, S1SPCCR - 0xE003000C)

SPCCR	Function	Description	Reset Value
7:0	Counter	SPI Clock counter setting	0

NOTA: l'etiqueta definida a LPC_Base.h és `SPI0_SPCCR`.

SPI Data Register (S0SPDR - 0xE0020008, S1SPDR - 0xE0030008)

This bi-directional data register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register. When a master, a write to this register will start a SPI data transfer. Writes to this register will be blocked from when a data transfer starts to when the SPIF status bit is set, and the status register has not been read.

Table 118: SPI Data Register (S0SPDR - 0xE0020008, S1SPDR - 0xE0030008)

SPDR	Function	Description	Reset Value
7:0	Data	SPI Bi-directional data port	0

NOTA: l'etiqueta definida a LPC_Base.h és `SPI0_SPDR`. Aquest registre només el farem servir per escriure la dada que volem enviar. El nostre registre de desplaçament esclau no enviarà cap dada al LPC2292.

NOTES:

- El programa haurà de configurar adequadament els registres de control del SPI0: SPI0_SPCR i SPI0_SPCCR.
- També haurà de configurar els pins P0.4, P0.5, P0.6 i P0.7 per a que siguin pins SPI (registre *PINSEL0).
- Cada cop que escrivim una dada al SPI0_SPDR, el registre de dades, es transmetrà una dada cap el registre de desplaçament, però NO s'activarà el rellotge del Latch. Per tant no tindrà efecte la transferència fins que activem, amb un flanc de pujada, el Latch. Això es farà amb el pin P0.20. Aquest pin haurà de ser configurat com a GPIO-sortida.
- Finalment, doncs, quan haguem escrit una dada al registre de dades del SPI caldrà fer el flanc de pujada a la línia P0.20. El problema és que això ho hem de fer un cop la transferència sèrie de la dada ha acabat. Per això es proposa que configureu el SPI0 per que en acabar de fer una transferència es generi una interrupció (que podeu dirigir a un canal del VIC que quedi lliure). A la RSI d'aquesta interrupció caldrà fer el flanc de rellotge del pin P0.20 (connectat al latch del registre de desplaçament).
- A la RSI del SPI0 aprofitarem per donar la pròxima dada a transmetre. D'aquesta manera la transmissió SPI és contínua.
- Les dades a transmetre seran el codi termòmetre de les conversions d'un dels potenciòmetres de la placa.
- A la RSI, al final, cal llegir el registre SPI0_SPSR i escriure un '1' al bit '0' del SPI0_SPINT (més el EOI habitual del VIC, escriure al registre VICVectAddr).

L6.1 Modifiqueu el vostre programa per posar en codi termòmetre les conversions A/D d'un dels potenciòmetres.

L6.2 Visualitzeu els senyals implicats a l'analitzador lògic (l'esquema de connexions el trobareu a la següent secció)