Nom i cognoms: Grup:

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona UNIVERSITAT POLITÈCNICA DE CATALUNYA DEPARTAMENT D'ENGINYERIA ELECTRÒNICA	**CISE IV** 16 de Juny de 2009 Data notes provisionals: 23 de Juny Límit d'al·legacions: 26 de Juny a les 14h. Data notes revisades: 30 de Juny

Professors: Sergi Bermejo, Manel Domínguez, Clemente Pol

Informacions addicionals:

- Durada de l'examen: 2h 40min.
- S'ha de respondre en aquests mateixos fulls d'enunciat al tots els problemes.
- S'han de lliurar els problemes per separat.

PROBLEMA 1 (30%)

El sistema digital programat que es proposa està basat en un µP amb 32 bits de Bus de Dades (**D0 – D31**), un Bus d'Adreces ***extern*** de 30 bits (**A2 – A31**) i un senyal de validació general d'adreces **AS*** (*Address Strobe**, actiu a nivell baix com indica el asterisc). Per tal d'indicar el(s) *byte(s)* que es tindran en compte en cada cicle de bus de lectura/escriptura, aquest µP presenta 4 senyals de sortida específics (amb la mateixa variació temporal que AS* quan s'activen):

BEX* (*Bus Enable X**) que amb X = 0 indica el *byte* D0 – D7; X = 1 indica el *byte* D8 – D15;
X = 2 " " D16 – D23; X = 3 " " D24 – D31.

Es demana:

a) (1 punt) Indicar la capacitat del mapa de memòria d'aquest µP en paraules (*words* de 32 bits) i en *bytes*.

Capacitat mapa en words = 2^{30}words = 1Gword ***Cap. en bytes = 2^{32}bytes = 4Gbyte***

b) (2 punts) Confeccioneu el diagrama d'aquest mapa **organitzat en *bytes***, tot indicant les següents adreces en hexadecimal:
- inicial i final del mapa,
- inicial i final d'una zona de 32M x 32 bits començant a la meitat del mapa,
- inicial d'una altra zona de 32M x 32 bits que acabi a la fi del mapa (adreces més altes).

0000 0000h. . . . . .adreça inicial mapa i de la 1a zona

8000 0000h. . . . .adreça inicial zona que comença a la meitat del mapa

87FF FFFFh. . . . .adreça final de la zona anterior de 32Mwords = 128Mbytes

F800 0000h. . . . .adreça inicial de la 2a zona de 32Mwords = 128Mbytes

FFFF FFFFh. . . .adreça final del mapa i de la 2a zona

c) (2 punts) Indiqueu els bits (línies) del bus d'adreces *extern* del µP (apart dels que es connectin directament al Bus d'Adreces dels xips de memòria) que haurien d'ésser utilitzats en una descodificació **completa** de qualsevol de les zones esmentades (de 32M x 32 bits) amb xips de memòria de 8M x 8 bits. I si els xips memòria fossin de 32M x 8bits?

Cada xip de 8Mbytes té una capacitat de 2^{23}bytes ⇒ Bus d'Adreces (B.A.) de 23 bits (A0 – A22) connectats a A2 – A24 del µP. Per tant, caldrà utilitzar (A25 – A31) del µP (7 bits del B.A.) per assolir una descodificació completa.

Cada xip de 32Mbytes té una capacitat de 2^{25}bytes ⇒ Bus d'Adreces (B.A.) de 25 bits (A0 – A24) connectats a A2 – A26 del µP. Per tant, caldrà utilitzar (A27 – A31) del µP (5 bits del B.A.) per assolir una descodificació completa.

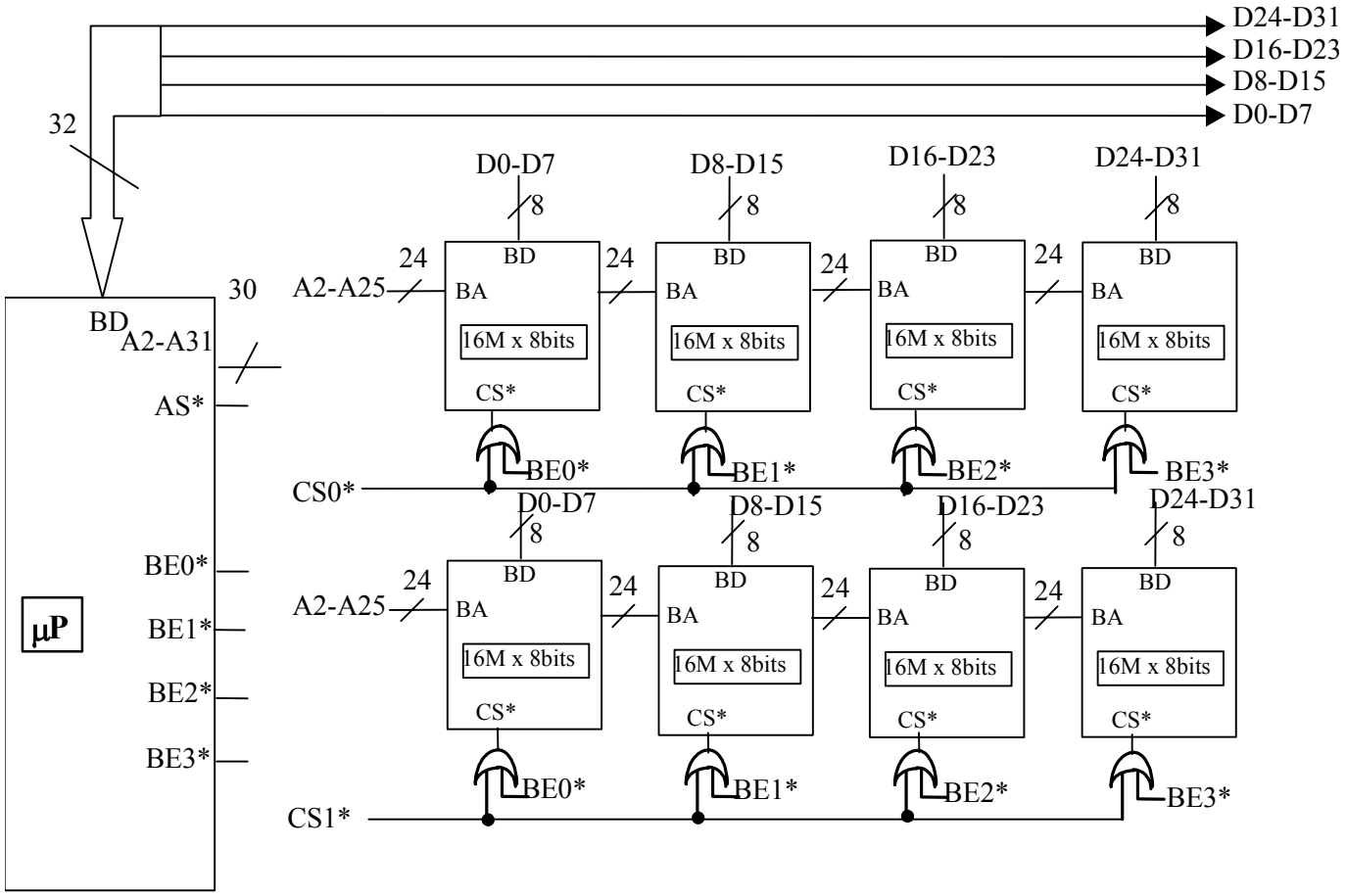
e) (1 punt) Segons el diagrama de la figura final, quants xips de memòria com el que ja està dibuixat (de 16M x 8 bits) caldran per a completar un bloc de memòria de 32M x 32 bits? Quants bits tindria el Bus d'Adreces d'aquests xips?

32Mx32bits = 8 x (16Mx8bits) ***16M = 2^{24}***

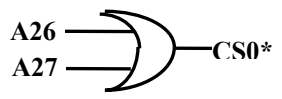
Nombre de xips = 8 ***Nombre d'Adreces per xip = 24 (A0 – A23)***

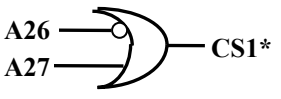
f) (4 punts) Efectueu les connexions que calguin al diagrama circuital de la figura, afegint-hi **el mínim de circuits necessaris** (es disposa d'inversors i de portes OR de 2/3/4 entrades), per tal de situar aquest bloc, que hem fet a l'apartat anterior, en posicions **consecutives** a **partir de la meitat del mapa de memòria del µP** en una descodificació **incompleta**, però **sense ocupar** la zona de 32M x32 bits del final del mapa (adreces més altes). També cal tenir en compte que el circuit ha de permetre cicles de lectura/escriptura de 1, 2, 3 o 4 *bytes*. **Quantes zones imatge en resulten**?

NOTA: No s'indiquen les línies de lectura/escriptura per claredat i facilitat del dibuix i no cal afegir-les. Tampoc cal dibuixar totes las línies de connexió si fan el dibuix molt confús; bastarà amb indicar el nom del senyal del µP que correspongui connectar als xips de memòria.



A26 és el bit següent d'adreces dels que van directament als xips de memòria i, per tant, cal utilitzar-lo per tal de seleccionar cada bloc de memòria en adreces consecutives. Com que la zona prohibida va des de F8000 0000h a FFFF FFFFh, su característica és que A31 = A30 = A29 = A28 = A27 ='1'. Així una manera molt senzilla de evitar aquesta zona seria, per exemple, si assegurem que les zones amb A27 =1 no entrin a la selecció de CS0* i CS1*:





Així, quedarien (A28 – A31) sense utilitzar i tindríem, doncs, 2^4 = 16 zones imatge. No hem emprat el senyal AS* perquè els senyals BEX* ja fan la seva funció de validació d'adreces.

NOTA: ***També s'acceptarà com a solució correcta si s'afegeix una entrada més a les portes OR de CS=* i CS1*: l'entrada A31*(es à dir, A31 negada) per si es va entendre de l'enunciat que només es podia omplir la meitat del mapa d'adreces altes (no l'obliga l'enunciat). Això sí, cal ésser conseqüent amb aquesta solució i només hauria 2^3 = 8 zones imatge.***

Nom i cognoms: Grup:

PROBLEMA 2 (35%)

La memòria StrataFlash (J3) d'Intel permet cicles de lectura asíncrona d'una, quatre o vuit paraules a la vegada.

1. **(4p) Cicle de lectura asíncrona d'una paraula**. A la Fig.1 es mostren les formes d'ona que dóna Intel pel cicle de lectura d'una paraula i a la Taula 1 es defineixen les especificacions temporals associades. En canvi, a la Fig.2 es mostren les formes d'ona mesurades a l'entrada de la memòria en el pitjor dels casos possibles (retards màxims, etc.) en un sistema format per aquesta i un µP configurat pel cicles de lectura amb 3 estats d'espera.

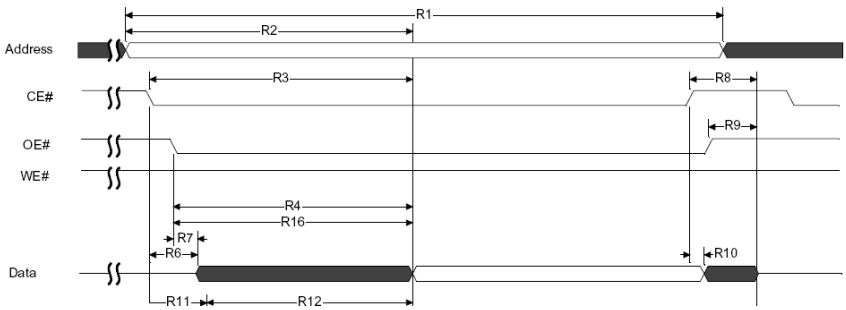


Fig.1. Especificacions que dóna el fabricant del cicle de lectura asíncrona d'una única paraula.

#	Sym	Parameter
R1	t_{AVAV}	Read/Write Cycle Time
R2	t_{AVQV}	Address to Output Delay
R3	t_{ELQV}	CEx to Output Delay
R4	t_{GLQV}	OE# to Non-Array Output Delay
R5	t_{PHQV}	RP# High to Output Delay
R6	t_{ELQX}	CEx to Output in Low Z
R7	t_{GLQX}	OE# to Output in Low Z
R8	t_{EHQZ}	CEx High to Output in High Z
R9	t_{GHQZ}	OE# High to Output in High Z
R10	t_{OH}	Output Hold from Address, CEx, or OE# Change, Whichever Occurs First
R11	t_{ELFL}/ t_{ELFH}	CEx Low to BYTE# High or Low
R12	t_{FLQV}/ t_{FHQV}	BYTE# to Output Delay
R13	t_{FLQZ}	BYTE# to Output in High Z
R14	t_{EHEL}	CEx High to CEx Low
R15	t_{APA}	Page Address Access Time
R16	t_{GLQV}	OE# to Array Output Delay

Taula 1. Especificacions temporals dels cicles de lectura.

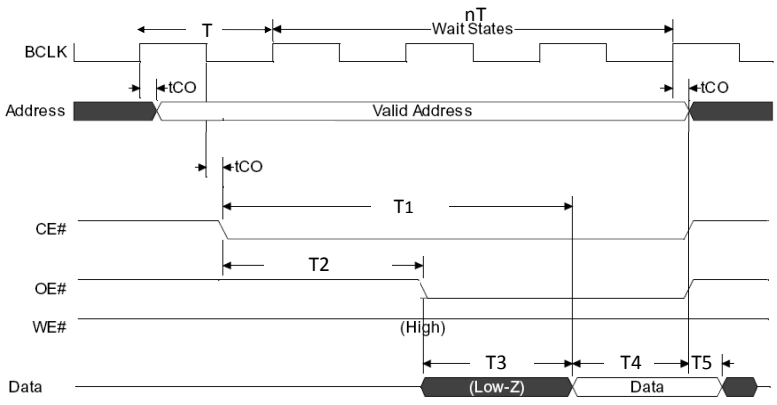


Fig.2. Cicle de lectura asíncrona d'una paraula mesurat entre un µP i la memòria.
(**NOTA**: BCLK és el senyal de rellotge del µP i n=3 a la figura.)

Sabent que si CE# = 1 la memòria roman desactivada i que $t_{ELQV} > t_{GLQV}$, es *demana* **justificadament**:

1.1 (1p) El valor de T1 –en funció dels paràmetres de la Taula 1– suposant que OE# s'activés a la vegada que CE# a la figura 2. (P.ex. $T1=t_{ELQX} - t_{OH}$)

T1= max(t_{AVQV},t_{ELQV}) *atesos que: 1) $t_{ELQV}<t_{GLQV}$ per tant si CE# i OE# s'activen a la vegada es complirà abans el temps d'accés t_{GLQV}; i 2) es diu que mentre CE#=1 la memòria roman desactivada, per tant que el temps d'accés des d'adreces es comença a comptar amb l'activació de CE#.*

1.2 (1p) El valor de T2 i T3 –en funció dels paràmetres de la Taula 1– per tal de endarrerir al màxim l'activació de OE# però sense que el temps d'accés des de OE# afectés al moment de l'aparició de la dada en el bus de dades calculat a l'apartat anterior.

*En el cas límit **$T3=t_{GLQV}$** per tal de que el temps de propagació des de OE# no canviés l'instant d'aparició de la dada. Sabent de la figura que T1=T2+T3, obtenim que **$T2=T1- t_{GLQV}$**.*

1.3 (1p) El valor mínim de T4 i T5 –en funció dels paràmetres de la Taula 1 i certs paràmetres temporals típics d'un µP– per tal de que la lectura fos correcte.

*Com a sistema digital síncron el µP llegirà la dada en l'últim flanc del BCLK amb dada vàlida. En aquest cas és l'últim flanc ascendent. Per una lectura correcta s'hauran de respectar els temps de set-up t_{DS} i hold t_H del µP. De la figura es poden mesurar els temps reals de que disposa el sistema que són: $t'_{DS}=T4-t_{CO}$ i $t'_H=T5+ t_{CO}$. En general, s'haurà de complir que $t'_{DS} \geq t_{DS}$ i $t'_H \geq t_H$. Els mínims valors t'_{DS} i t'_H seran llavors per a $t'_{DS}(mín)=t_{DS}(mín)$ i $t'_H(mín)=t_H(mín)$. D'aquí obtenim finalment que **$T4(mín)=t_{DS}(mín)+t_{CO}$ i $T5(mín)= t_{OH}(mín)=t_H(mín)-t_{CO}$**. Per tant, sense incloure **t_{CO}** podrem dir que: **$T4(mín)>t_{DS}(mín)$ i $T5(mín)=t_{OH}(mín)<t_H(mín)$**.*

1.4 (1p) La inequació pel càlcul òptim dels estats d'espera 'n' a inserir en funció de T, t_{CO}, T1,…,T5 i t_{DS} = temps *set-up* de dades del µP (P.ex. $T(n+1) > T1-T2 - t_{CO}$). (**NOTA**: veure definició de 'n' a la Fig. 2.)

*D'acord amb la finestra temporal observada a la Fig.2 i sabent que l'instant de captura de la dada és l'últim flanc ascendent de BLCK (veure l'apartat anterior) es pot escriure que: **$(0.5 + n)T \geq t_{CO} + T1 + t_{DS}$**.*

2. **(6p) Cicle de lectura asíncrona de 4 paraules**. A la Fig.3 es mostren les formes d'ona que dóna Intel pel cicle de lectura de 4 paraules i a la Taula 1 es defineixen les especificacions temporals associades. En canvi, a la Fig.4 es mostren les formes d'ona mesurades a l'entrada de la memòria en el pitjor dels casos possibles (retards màxims, etc.) en un sistema format per aquesta i un μP configurat amb 3 estats d'espera per a la lectura de la primera paraula i 1 estat d'espera per a la resta de lectures.

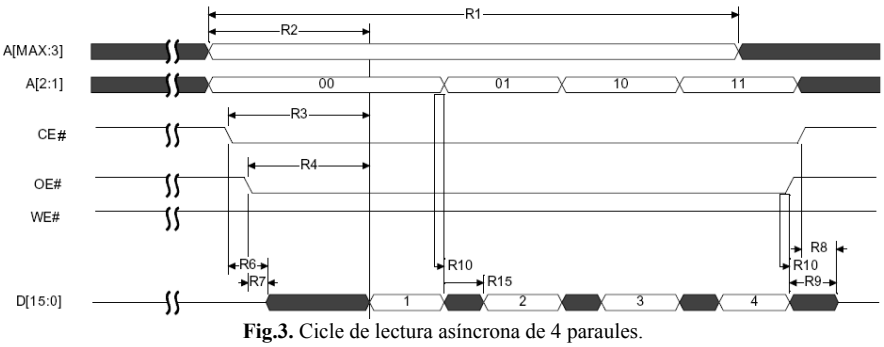


Fig.3. Cicle de lectura asíncrona de 4 paraules.

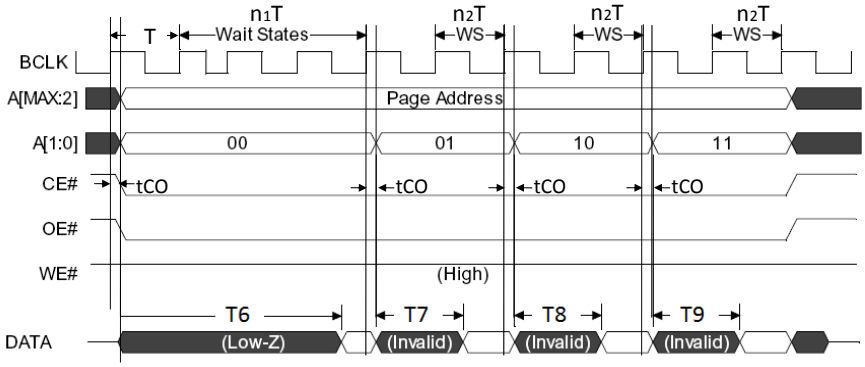


Fig.4. Cicle de lectura asíncrona de 4 paraules mesurat entre un μP i la memòria.
(**NOTA**: BCLK és el senyal de rellotge del μP; n_1=3 i n_2=1 a la figura.)

Es *demana* **justificadament**:

2.1 (1p) El valor dels paràmetres t_{ELQX} i t_{GLQX} de la Taula 1.
A la Fig. 4 es veu com la dada apareix just amb mateix instant que s'activen CE# i OE#. Per tant, $t_{ELQX} = t_{GLQX} = 0$.

2.2 (1p) El valor de T6 en funció dels paràmetres de la Taula 1.
Les adreces, CE# i OE# canvien en el mateix instant, llavors la dada apareixerà quan els major dels tres temps d'accés es compleixi. Per tant, $\boldsymbol{T6 = max(t_{AVQV}, t_{ELQV}, t_{GLQV})}$.

2.3 (1p) El valor de T7, T8 i T9 en funció dels paràmetres de la Taula 1.
De la Fig. 3 es desprèn que $T7=T8=T9=R15=t_{APA}$.

2.4 (1p) La inequació pel càlcul òptim dels estats d'espera 'n_1' a inserir per a la primera paraula en funció de T, t_{CO}, T6,…,T9 i t_{DS}= temps *set-up* de dades del μP. (**NOTA**: veure definició de 'n_1' a la Fig. 4.)
Aplicant el principi de que el μP llegirà la dada en l'últim flanc del BCLK amb dada vàlida, podem escriure que:
$\boldsymbol{(1+n_1)T \geq t_{CO} + T6 + t_{DS}}$.

2.5 (1p) La inequació pel càlcul òptim dels estats d'espera 'n_2' a inserir per a la resta de paraules en funció de T, t_{CO}, T6,…,T9 i t_{DS}= temps *set-up* de dades del μP. (**NOTA**: veure definició de 'n_2' a la Fig. 4.)
Els tres cicles de lectura corresponents a la resta de paraules són idèntics. Així, restringirem l'estudi a un d'ells. Aplicant de nou el principi de que el μP llegirà la dada en l'últim flanc del BCLK amb dada vàlida, podem escriure que:
$\boldsymbol{(1+ n_2)T \geq t_{CO} + T_x + t_{DS}}$ ***amb*** $\boldsymbol{T_x = T7 = T8 = T9}$.

2.6 (1p) Calcular la diferència que comportaria el *temps de lectura promig* d'una paraula fent servir només cicles d'una única paraula (Fig. 2) en comparació a emprar només cicles de quatre en quatre (Fig.4). (**NOTA**: pel càlcul usar els estats d'espera definits a les Fig. 2 i 4, es a dir n=n_1=3 i n_2=1, i expressar el resultat només en funció de T.)
Els temps de lectura mesurats únicament en funció de T implica mesurar entre els instants d'inici entre lectures successives determinats pel canvi del bus d'adreces. Per tant, si t_L*(cicles 1 paraula)=4T i* $<t_L$*(cicles 4 paraules)>=(4T + 3x2T)/4=2.5T, llavors la* ***diferència promig=4T-2.5T=1.5T.***

PROBLEMA 3 (35%)

Volem connectar una memòria *Flash* d'1 Mbyte de capacitat total al microcontrolador LPC2292 de Samsung. Per evitar fer esquemes complexos de connexions, fem servir la memòria 25LF080A de SST, que utilitza el bus SPI (*Serial Peripheral Interface*) per comunicar-se amb el microcontrolador.

El bus SPI és **un standard de comunicació sèrie síncron**. Les característiques bàsiques d'aquest bus són:

- En el sistema només hi ha un *Master* (normalment el microcontrolador). La resta de dispositius connectats al bus SPI són *Slaves* i només realitzen transaccions pel bus quan el Master els ho demana.
- El *Master*, quan vol comunicar-se amb un dispositiu, li activa un senyal d'habilitació (noms típics CS* o CE*).
- El *Master* té els següents senyals:
    - **SCK**: *Serial clock* (generat pel *Master*). Servirà per sincronitzar l'enviament d'informació. Només està actiu quan s'estan transferint dades.
    - **MISO**: *Master Input Slave Output*. Senyal d'entrada en el *Master* per on entra la informació sèrie proporcionada pels *Slaves*.
    - **MOSI**: *Master Output Slave Input*. Senyal de sortida del Master per on surt la informació sèrie del *Master* cap als *Slaves*.
- Els *Slaves* tenen els següents senyals:
    - **SCK**: *Serial clock*. Senyal d'entrada de rellotge als *Slaves*.
    - **SI**: *Slave Input*. Senyal d'entrada al *Slave* per on entra la informació sèrie enviada pel *Master*.
    - **SO**: *Slave Output*. Senyal de sortida del *Slave* per on surt la informació sèrie cap el *Master*
- El bus es síncron. Les dades només canvien en un flanc de rellotge (pujada o baixada) i són llegides en l'altre flanc de rellotge (baixada o pujada).
- Les línies MISO i MOSI (o el que és el mateix SI o SO) són d'un bit d'amplada (la transmissió és sèrie).

Es important tenir en compte que la transmissió és **full duplex**. Es a dir, quan el Master decideix que cal fer una transferència, s'activarà el rellotge SCK i hi haurà:

- una transferència del Master al Slave per la línia MOSI, i
- una transferència del Slave al Master per la línia MISO.

Hem connectat la memòria al microcontrolador d'aquesta manera:

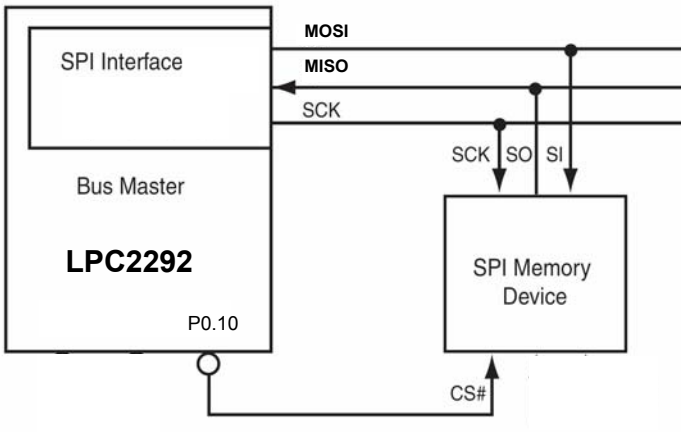


Figura 1: Esquema de connexions.

Com es pot veure hem connectat el pin 10 del Port 0 del LPC2292 al *chip select* (CS*) de la memòria.

El microcontrolador LPC2292 ens permet configurar la comunicació síncrona sèrie, SPI, de vàries maneres diferents. Dues d'elles són CPOL=0, CPHA=0 o CPOL=0, CPHA=1. A l'opció:

- CPOL=0, CPHA=0: el microcontrolador treu dada per línia MOSI en flanc de baixada de SCK, i llegeix dada de línia MISO en flanc de pujada.

- CPOL=0, CPHA=1: el microcontrolador treu dada per línia MOSI en flanc de pujada de SCK, i llegeix dada de línia MISO en flanc de baixada.

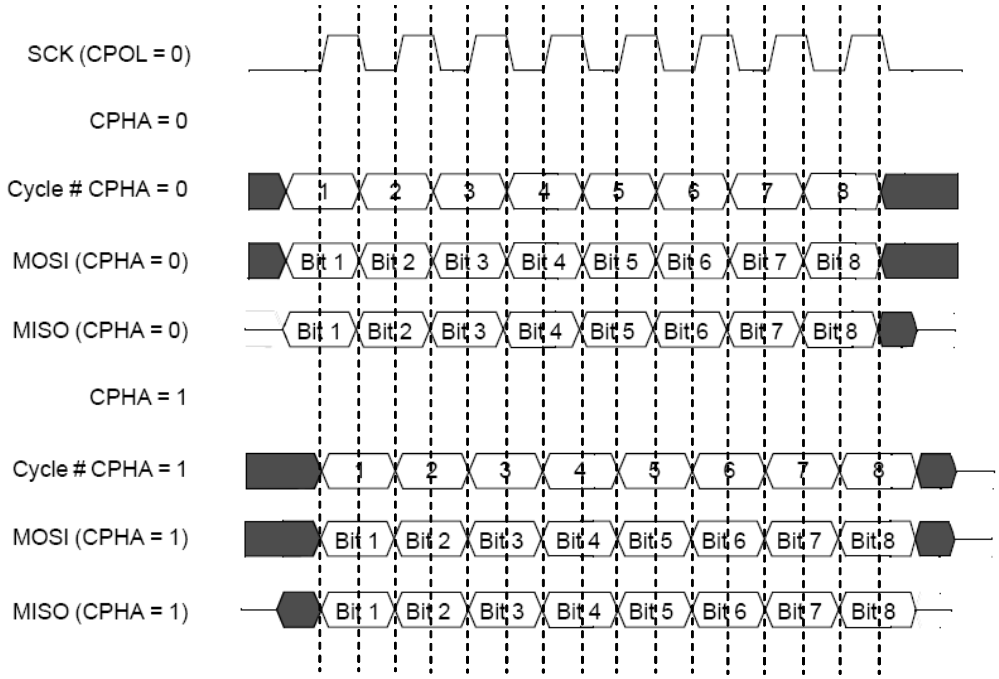


Figura 2: Transferència SPI amb opcions CPOL=0, CPHA=0, o CPOL=0, CPHA=1.

La temporització de la memòria pel que fa a les transferències pel bus SPI ve donada per les següents gràfiques i taula.

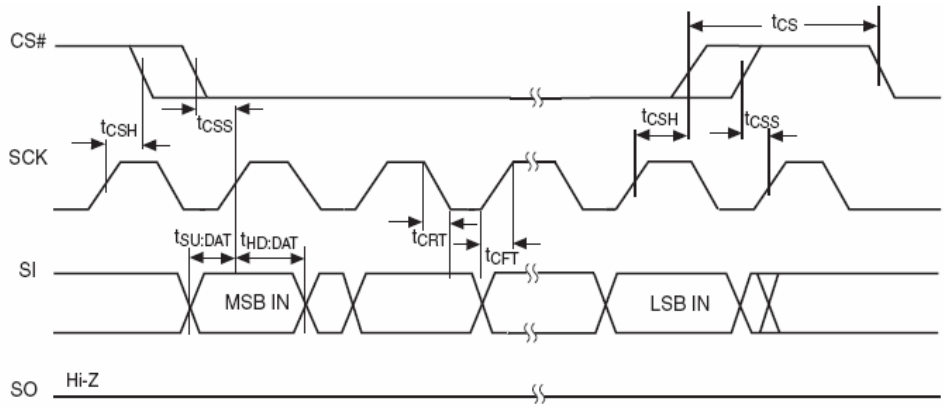


Figura 3: Temporització **de la memòria** en transaccions del Master (microcontrolador) cap al Slave (memòria).

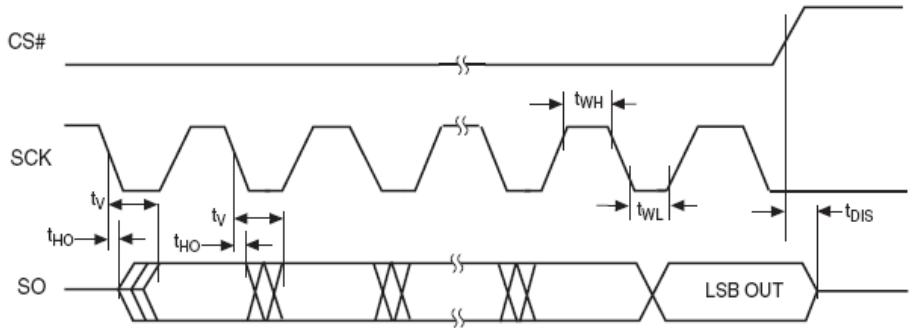


Figura 4: Temporització **de la memòria** en transaccions del *Slave* (memòria) cap al *Master* (microcontrolador).

Symbol (Notes)	Parameter	Min	Typ (Notes)	Max (Notes)	Unit
t_V	Output Valid			10	ns
t_{HO}	Output Hold Time	0			ns
$t_{HD:DAT}$	Data in Hold Time	5			ns
$t_{SU:DAT}$	Data in Setup Time	5			ns

Taula 3: Taula de temps **de la memòria** per a transaccions pel bus SPI.

a) (1 punt) A partir de les Figures 2, 3 i 4, i de la Taula 3, com configurarem el microcontrolador (CPOL=0, CPHA=0, o CPOL=0, CPHA=1) ? Justifiqueu breument la resposta.

Ha de ser CPOL=CPHA=0 perquè la memòria treu les dades a línia MISO en flanc de baixada i llegeix MOSI en flanc de pujada.

Les característiques dinàmiques del microcontrolador per les transferències pel bus SPI són les següents:

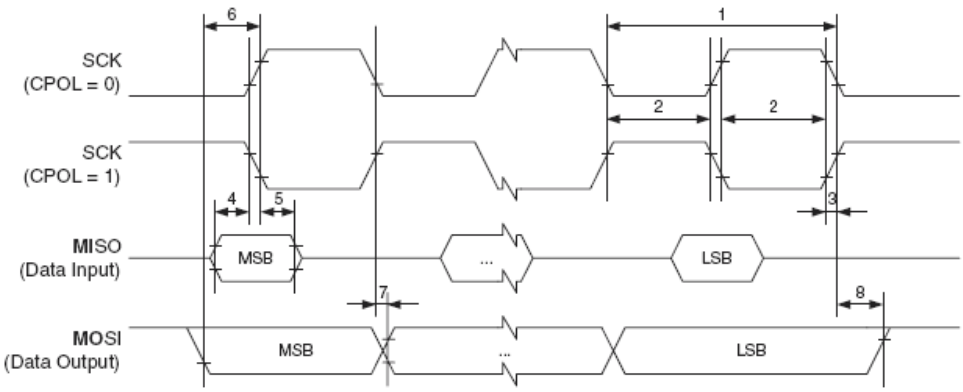


Figura 5: Temporització **del microcontrolador** en transaccions pel bus SPI.

Table 120. SPI Timing Parameters

	Description	Mode	Min	Typ	Max	
1	SCK period	Master				
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		0.5 • t_{SCK}		ns
7	SCK to out	Master		10		
8	SCK to out high	Master		10		

Taula 3: Taula de temps **del microcontrolador** per a transaccions pel bus SPI.

b) (4 punts) Si hem programat la freqüència de rellotge SCK a 5 MHz, es faran adequadament les transaccions entre el microcontrolador i la memòria (en ambdós sentits) pel bus SPI ? Justifiqueu la resposta.

Transaccions micro -> memòria (línia MOSI):

- *Setup de memòria tSU:DAT (min) = 5ns.*
  Les dades estan abans del flanc de rellotge un temps mínim de: T/2-Temps (7) = 100ns-10ns =90ns. Ok.
- *Hold de memòria tHD:DAT (min)=5 ns. Les dades es mantenen un temps mínim de T/2=100ns. Ok*

Transacccions memòria -> micro (línia MISO):

- *Setup micro (4)=10 ns*
  Les dades estan abans del flanc de relloge un temps mínim de: T/2-tV=100ns – 10 ns= 90ns. Ok
- *Hold micro(5)=10ns*
  Les dades es mantenen un temps mínim de 100ns=T/2. Ok.

Per llegir una dada de la memòria és necessari primer indicar a la memòria que volem fer una lectura, després li passarem l'adreça i per últim la memòria enviarà la dada al microcontrolador. Això es tradueix en:

- Primer el microcontrolador envia una 'ordre' de lectura a la memòria. En el nostre cas aquesta comanda pren el valor 0x03.
- Després el microcontrolador envia l'adreça que vol llegir. Necessitaríem 20 bits d'adreces (la memòria és d'1 Mbyte), però com les transferències són sempre múltiples de byte, s'envien 3 bytes (24 bits). Els bits de major pes (MSB: *Most significant bi*t) A23-A20 no seran tinguts en compte per la memòria.
- Finalment el microcontrolador espera la dada per la línia MISO (*Master Input Slave Output*)=SO (*Slave Output*).

Un exemple de lectura d'una dada de la memòria es pot veure a la figura següent:

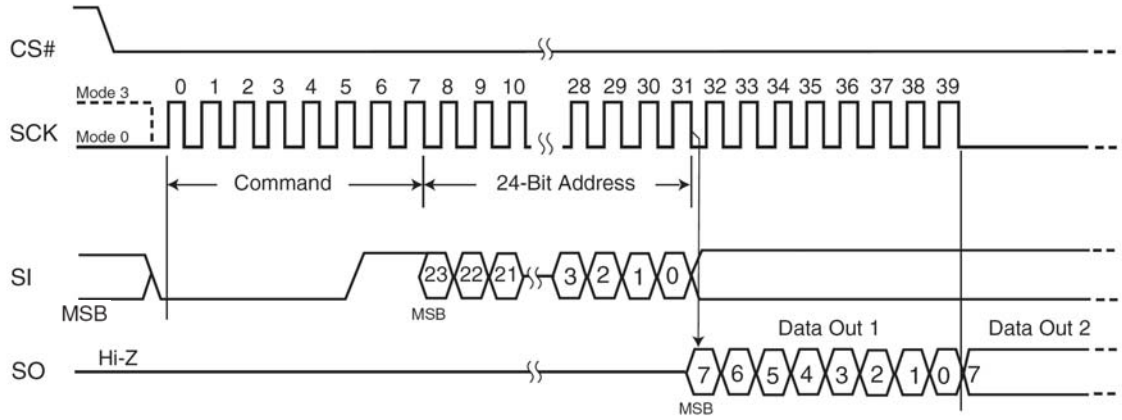


Figura 6: Exemple de lectura d'un byte de la memòria pel bus SPI.

La memòria, un cop rebuda la comanda i l'adreça, en el flanc de baixada del període 31, envia la dada corresponent a l'adreça demanada. En cas de que hi hagi més cicles de rellotge envia la dada de la següent adreça i així indefinidament, fins que es desactivi el senyal de CS* o no hi hagi més cicles de rellotge. La lectura de vàries **adreces consecutives** de memòria es podria fer així:

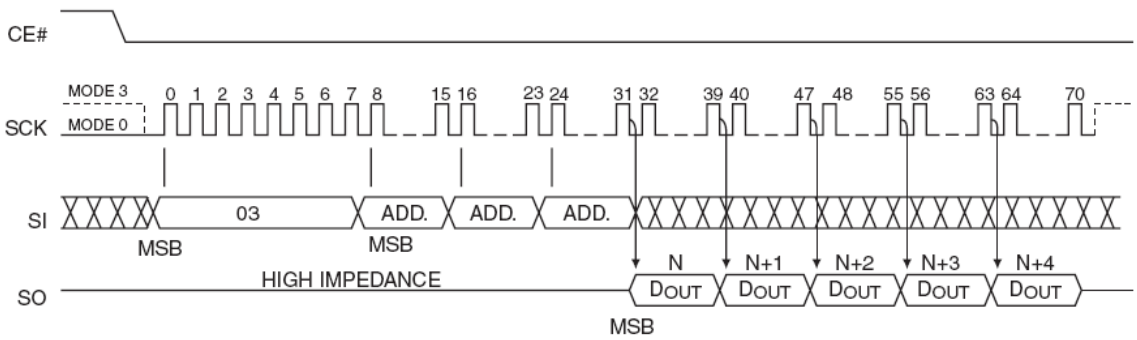


Figura 7: Exemple de lectura de bytes de la memòria amb adreces consecutives pel bus SPI.

c) (1.5 punts) Quants cicles de rellotge SCK trigarem en llegir 10 bytes consecutius de la memòria a partir d'una adreça qualsevol de la memòria ? Justifiqueu breument la resposta.

*8T+24T+8*10=112T (comanda+adreces primer byte + 8*10 bytes).*

d) (1.5 punts) En les mateixes condicions que a l'apartat c), quants cicles de rellotge SCK trigarem en llegir 10 bytes de la memòria en adreces no consecutives ? Justifiqueu breument la resposta.

*(8T+24T+8T)*10 =400T=(comanda + adreces primer byte + 1 byte)*10*

De cara a comunicar-nos amb la memòria farem servir el perifèric intern SPI0, del LPC2292 dedicat al bus SPI. De cara a generar adequadament el senyal CE* de la memòria, **hem connectat el pin 10 del port d'E/S P0 del microcontrolador al CE* de la memòria (Figura 1)**.

Suposarem que hem configurat bé el perifèric intern, SPI0, del microcontrolador. Aquest perifèric només fa transferències *byte* a *byte*. Els registres que és necessari manipular per fer transaccions amb la memòria són:

- **SPI0_SPDR (bits 7:0)**: Registre de dades del bus. Si escrivim una dada es farà una transacció del microcontrolador cap a la memòria (per la línia MOSI) i de la memòria al microcontrolador (per la línia MISO). El perifèric SPI0 enviarà primer el bit més significatiu d'aquest registre (bit 7) i acabarà amb el menys significatiu.

- **SPI0_SPSR (bit 7):** El bit 7 d'aquest registre (SPIF) indica que la transacció d'un byte ha acabat. Quan la dada hagi estat transferida del microcontrolador a la memòria, el bit SPIF es posarà a '1'. La dada transferida de la memòria al microcontrolador (tingui o no sentit) es trobarà al registre SPI0_SPDR. **El bit 7 (SPIF) es tornarà a posar a '0' quan haguem llegit el registre SPI0_SPSR, i haguem també llegit o escrit el registre de dades SPI0_SPDR.**

Per llegir 10 bytes consecutius de la memòria, a partir de l'adreça 0x12345, hem fet el següent codi:

```
int i,aux;
char punter[10];

*IOCLR0=P10                          // Posem a '0' el bit 10 del Port 0 (activem el CE* de la memòria)

*SPI0_SPDR=0x03;                     // Enviem la comanda 'read' a la memòria (0x03)
while (*SPI0_SPSR & 0x80 == 0);      // Esperem a que la transacció hagi acabat (bit SPIF)

*SPI0_SPDR=0x01;                     // Comencem a enviar l'adreça
while (*SPI0_SPSR & 0x80  == 0);

*SPI0_SPDR=0x23;
while (*SPI0_SPSR & 0x80  == 0);

*SPI0_SPDR=0x45;                     .
while (*SPI0_SPSR & 0x80  == 0);     // Acabem d'enviar l'adreça

for (i=0;i<10;i++) {

        *SPI0_SPDR=0x00;             // Escrivim una dada (amb un valor que no importa) per tal que surti
                                     //        per la linia MOSI. Així també rebrem la  dada en lectura
                                     //        per la línia MISO.

        while (*SPI0_SPSR & 0x80  == 0);  // Esperem a rebre la dada de la memòria
        punter[i]=*SPI0_SPDR & 0xFF;      // Llegim el valor de la posició 0x12345+i de la memòria.
}

*IOSET0=P10;                         // Desactivem el CS* de la memòria. (posem a '1' el bit 10 del Port0)
```

e) (1 punt) Si per esborrar completament el xip tenim **la comanda '0x60'** (sense enviar res més), escriviu les instruccions que esborrin el xip sencer.

```
*IOCLR0=P10;
*SPI0_SPDR=0x60;                    // enviem comanda
while (*SPI0_SPSR & 0x80==0);
*IOSET0=P10;
```

Per escriure a la memòria flash primer esborrarem la memòria i després escriurem els bytes que vulguem amb el mètode de AAI (***Auto Address Increment*, comanda 0xAF**). A la primera escriptura s'envia l'adreça. A les següents no cal, només hem de posar la comanda (0xAF) i la dada a escriure.

Entre escriptures successives cal desactivar el senyal CE* durant un temps mínim de 20µs. Podem fer servir una funció 'retard()' per generar aquest temps d'espera. Suposarem que la rutina 'retard' ha estat escrita i es pot fer servir al nostre codi.

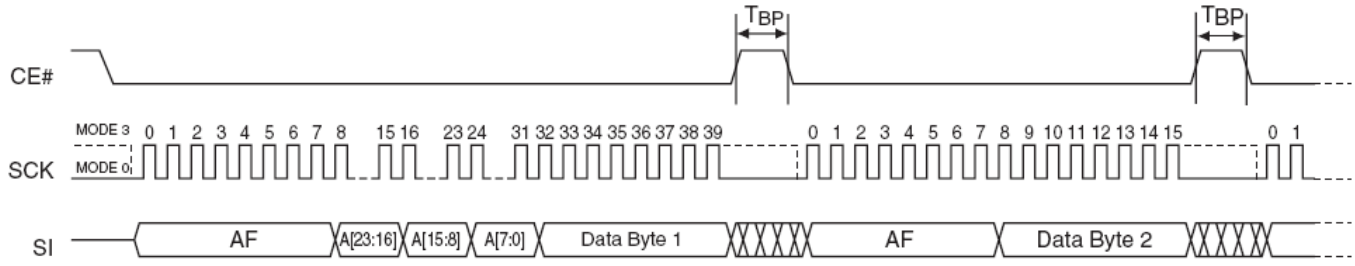


Figura 8: Exemple d'escriptura de diversos *bytes* en adreces consecutives de la memòria, fent servir el mètode AAI (*Auto Address Increment*).

f) (1 punt) Suposant que hem esborrat prèviament la memòria, feu un programa que escrigui 100 *bytes*, amb contingut '0xFF' a la memòria *Flash*, a partir de l'adreça 0x54321.

```
*IOCLR0=P10;

*SPI0_SPDR=0xAF;                    //comanda
while (*SPI0_SPSR & 0x80==0);

*SPI0_SPDR=0x05;                    //adreça
while (*SPI0_SPSR & 0x80==0);
*SPI0_SPDR=0x43;                    //adreça
while (*SPI0_SPSR & 0x80==0);
*SPI0_SPDR=0x21;                    //adreça
while (*SPI0_SPSR & 0x80==0);

*SPI0_SPDR=0xFF;                    //enviem primera dada
while (*SPI0_SPSR & 0x80==0);

for (i=1;i<100;i++) {               // enviem 99 dades després
        *IOSET0=P10; retard(); *IOCLR0=P10;

        *SPI0_SPDR=0xAF;
        while (*SPI0_SPSR & 0x80==0);
        *SPI0_SPDR=0xFF;
        while (*SPI0_SPSR & 0x80==0);
}
*IOSET0=P10;
```