Índex

Índex de taules

Índex de figures

Agraïments

Primer de tot agrair al meu tutor Martí Fàbregas les hores dedicades i el seu suport per tirar el projecte endavant.

També agrair a la meva família el suport que m'han donat, en moments puntuals, per enllestir el projecte.

Capítol 1

Descripció i Objectius del projecte

Contents

1.1 Introducció

El bruixisme és una activitat nocturna i no funcional dels músculs de la masticació, que pot ocasionar greus problemes, depenent de la freqüència i de la intensitat en que es produeix. L'evolució d'aquesta malaltia pot provocar alteracions greus en l'engranatge de les dents de la boca, degeneracions de les genives, deformacions mandibulars. Els símptomes d'aquesta malaltia poden aparèixer en forma de mal de cap, mals d'oïda, dolor de coll, dolor al obrir la boca, etc.

La diagnosi del grau de bruxisme es fa després de controlar la pressió i moviment dels músculs que mouen la barra inferior mentre es dorm. La realització d'aquests controls comporta moltes complicacions, des del possible internament del pacient fins als costosos equips de mesura. Coneixem de l'existència d'un sistema molt elemental de prendre aquestes mesures, l'inconvenient més gran que ens han constatat del sistema, a part del seu preu,

és la seva extremada simplicitat que només permet una mesura global i massa elemental de l'activitat muscular nocturna (només la dóna resumida en tres valors possibles, no present, present o intensa).

S'ha creat un sistema pel diagnòstic de la patologia del bruixisme el qual consta de dos subsistemes: el sistema de captura i el sistema de processament de dades.

- El sistema de captura de dades està compost per dos blocs: captura del senyal i pretractament del senyal.
  - La captura del senyal es fa mitjançant sensors específics de la electromiografia [1], que permeten una bona adherència a la pell i són eficients amb tensions baixes.
  - El pretractament del senyal està dividit en els següents elements: amplificador, filtre i postamplificador.
    - L'amplificador és l'encarregat d'amplificar les tensions procedents del sistema de captura del senyal a un guany de 100.
    - El filtre elimina les freqüències inferiors a 50Hz i superiors a 2000Hz del senyal procedent l'amplificador.
    - El postamplificador torna a amplificar el senyal degut a que el senyal continua sent baix per procedir a la seva emissió.
- El sistema de procesament de dades està compost per dos blocs: receptor i processament del senyal.
  - El receptor és una ràdio sintonitzada a la freqüència 432,95MHz, mitjançant la sortida audio s'envia el senyal cap a l'entrada del convertidor A/D del microprocessador.
  - El processament del senyal es fa mitjançant el sistema microprocessat Olorim, el qual s'encarrega d'emmagatzemar les dades a la memòria interna de la placa, per al seu posterior enviament via serie cap a un PC per a la seva visualització en un gràfic.

Aquesta part del projecte preten millorar l'apartat de capacitat per a les dades i oferir major portabilitat mitjançant una targeta SD. Per dur a terme aquesta millora recollirem les dades emmagatzemades a la memòria interna (la qual esta limitada a una capacitat de 62Kb) del sistema microprocessat i les emmagatzemarem en una memòria SD (la qual té una capacitat molt més amplia, actualment 32Gb). Per poder connectar la targeta SD a la placa

[1] Es una prova que evalua la salut dels músculs i dels nervis que controlen els músculs

Olorim, cal un connector per a targetes SD. Aquest connector el tenim al port 0 de l'Olorim.

Les dades s'emmagatzemaran a la targeta SD dins un fitxer creat previament amb l'ordinador, el qual ha de ser el primer fitxer que es crea a la targeta, ja que ha d'estar en sectors consecutius. En aquest fitxer s'aniran emmagatzemant les dades que ens proporcioni el sistema de captura en format RAW.

Per poder visualitzar les dades amb més claredat, s'ha creat un petit programa mitjançant matlab per poder veure les dades obtingudes en una gràfica.

1.2 Estudi previ

1.2.1 El sistema

En un primer moment vam veure que teniem dos possibilitats per dur a terme aquest projecte, un era amb el sistema microprocessat Olorim i l'altre era mitjançant un PIC.

Tenint en compte que aquest projecte era una continuació d'un projecte anterior que estava desenvolupat amb el sistema microprocessat Olorim, finalment ens vem decantar per aquest. Un altre raó per decantar-nos per l'Olorim va ser que es una placa ja coneguda per els estudiants de microprocessadors i el seu joc d'instruccions ens és conegut, en canvi, el els PIC's fan servir un altre joc d'intruccions completament diferent.

També vam pensar de fer la comuniciació amb la placa Olorim a través d'USB, ja que el port serie està en vies d'extinció (els PC's actuals ja no incorporen cap port serie), però va ser descartat ràpidament ja que el protocol USB és propietari. Tampoc esdevenia cap prioritat, ja que amb la velocitat del port serie suposavem que tindriem suficient per al nostre projecte. Tal com es pot veure més endavant a l'apartat 4.3 temps d'execució pàgina 71.

1.2.2 La memòria

El sistema complet està previst ulilitzar-lo com un equip independent que s'utilitzarà amb dues parts, la primera conté el sistema de captura de dades i el sistema de processament de dades ja realitzat al mòdul anterior, la segona part la conforma una placa microprocessada que rep les dades del primer mòdul i les ha de guardar dins un fitxer en una memòria flash per tal que aquesta memòria pugui ser extreta del sistema microprocessat i poder visualitzar les dades en un gràfic mitjançant qualsevol PC.

1.2.3 Tipus de targetes

Al mercat hi ha diferents tipus de targetes de memoria:

	Pins	Mode SPI	Mode 1 bit	Mode 4 bits	Mode 8 bits	Rellotge
MMC	7	Opcional	Si	No	No	0-20 MHz
RS MMC	7	Opcional	Si	No	No	0-20 MHz
MMC Plus	13	Opcional	Si	Si	Si	0-54 MHz
Secure MMC	7	Necesari	Si	Si	Si	0-20 MHz
SD	9	Necesari	Si	Opcional	No	0-25 MHz
SDIO	9	Necesari	Si	Opcional	No	0-25 MHz
MiniSD	11	Necesari	Si	Opcional	No	0-20 MHz
MicroSD	8	Opcional	Si	Opcional	No	0-12 MHz

Taula 1.1: Característiques: Targetes de memòria

Les compacts flash van ser les primeres a aparèixer al mercat, com el seu nom indica fan servir memòria tipus flash, els primers en fabricarla van ser els de SanDisk, també van ser els encarregats de fer l'especificació. Inicialment es van fabricar a partir de memòries flash basades en NOR de Intel, encara que finalment van fer servir NAND. La memòria flash basada en NOR té una densitat menor que la dels sistemes més recents basats en NAND. Les compact flash tenen un petit controlador IDE integrat al mateix dispositiu.

Les targetes MMC són les precursores de les targetes SD que s'expliquen a continuació, la única diferència amb la SD és el seu grossor, ja que es una mica més reduït, i la carència de la pestanya de seguretat que evita sobrescriure informació emmagatzemda.

Les targetes SD igual que les compact flash i totes les targetes que analitzem fan servir memòria flash. La majoria de fabricants de càmeres fotogràfiques digitals fan servir targetes SD, excepte Olympus i Fuji que fan servir targetes XD, i Sony que fa servir les memory stick. Aquesta targeta es diu Secure perque Toshiba va afeigir hardware de cifrat a la ja existent targeta MMC, per treure el temor a la indústia de la música, que es queixaven que les targetes MMC permetien el pirateig fàcilment. En teoria aquest cifrat permetria complir fàcilment els esquemes DRM sobre la música digital. Com a curiositat les targetes MMC es poden fer servir a les ranures per targetes SD, ja que les targetes MMC són més fines.

Les compact flash van quedar descartades per el nombre elevats de pins que fan servir, ja que a la placa Olorim només disposem de 4 ports de 8 bits i les compact flash tenen 50 pins.

Per aquesta raó vam optar per les targetes SD ja que actualment sembla que són les que tenen més quota de mercat. També cal dir que vam optar

per implementar l'standard SPI, ja que l'standard SD es propietari i està subjecte a patents.

Targetes SD

Les targetes SD estaven limitades a 2 GB de capacitat. En un principi estava previst que la capacitat màxima fos de 4 GB, però aleshores el sistema d'arxius hauria de ser FAT32 i l'especificació SD només admet FAT12 i FAT16. Aleshores es va crear un nou estandard SDHC (SD High Capacity) que fa servir el mateix factor de forma físic però fa servir un adreçament de memòria diferent (adreçament per sector enlloc d'adreçament per byte) i permet fer servir targetes amb capacitat de 4 GB fins a 32 GB.

1.2.4 Com es comunica el processador?

Per ports

El processador es comunica directament amb els perifèrics a través dels ports disponibles al processador. A aquest sistema s'hi va incorporar el sistema de chipsets, que consisteix en sistemes especifics que contenen entre altres coses la UART i la USART que fa servir el micro, ja que els diferents dispositius que s'hi conectaven tenien velocitats molt dispars. La velocitat de la memòria era molt més ràpida que la velocitat del port serie, però quan el port serie estava transmeten dades el bus quedava ocupat i cap perifèric més podia accedir, fins al final de la transmissió. Per tant, la memòria tot i ser més ràpida havia d'esperar a que el port serie deixes d'utilitzar el bus, per poder accedir-hi; això comportava una gran pèrdua de temps i d'aqui que es desenvolupes el sistema de chipsets.

Northbridge i southbridge

El sistema de chipsets s'ha desenvolupat posant en un xip específic les unitats d'entrada i sortida, del processador, bàsicament les UARTS i les USARTS. Actualment la majoria de microprocessadors fan servir dos tipus de xips, el northbridge i el southbridge. El processador es comunica amb els chipsets Northbridge (perifèrics ràpids; bus AGP, PCI Express, Gigabit Ethernet) i Southbridge (perifèrics lents; Fast Ethernet, USB, SMBus, targetes de so), i aquests es comuniquen amb els diferents dispositius. Amb aquest sistema el que es va aconseguir va ser descarregar el processador de les tasques d'entrada/sortida. Arribat a aquest punt el coll d'ampolla deixava d'estar al processador per passar al Northbridge.

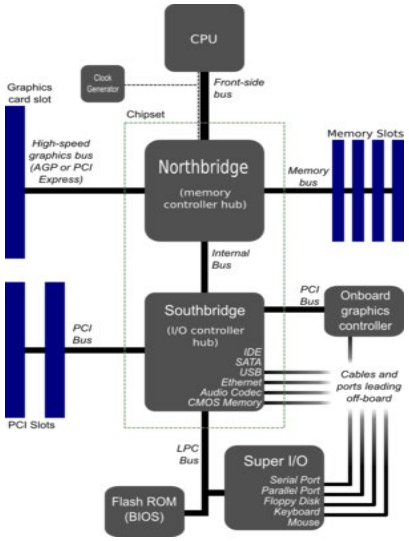


Figura 1.1: Northbridge i Southbridge

Hypertransport

Un altre sistema de comunicació del microprocesador amb els preíferics es fent servir la tecnologia Hypertransport també coneguda com Lightning Data Transport, és una tecnologia de comunicació bidireccional, que funciona tant en serie com en paral·lel y que ofereix un bus de molt baixa latencia i gran ample de banda. Una de les funcions de la tecnlogia hypertransport es com a front side bus, amb això s'elimina el coll d'ampolla de la tecnologia Northbridge; ja que hypertransport és més ràpid i perque el processador pot tenir varis enllaços hypertransport amb l'exterior. També es fa servir per interconectar processadors NUMA.
Es tracta d'un bus que fa servir de 2 a 32 linies de dades d'un bit. Les possibles configuracions són 2, 4, 8, 16 i 32 linies de dades, que es diuen CAD en la nomenclatura d'hyperTransport. Cada linia transmet a una freqüencia entre 200 MHz i 2,6 GHz. El bus hyperTransport opera a 2,5 v i fa servir scrambling [2] amb un patró diferent per a cada linia per evitar la autoinducció entre pistes.

1.2.5 Material necesari

Per dur a terme aquest projecte fem servir una placa microcontroladora Olorim que explicarem amb més profunditat a l'apartat 3.2 de la pàgina 23. Per

[2] Tècnica que permet canviar la fase d'un senyal modificant la distància entre pistes

la comunicació via serie entre el PC i el sistema microprocessat Olorim fem servir un cable serie creuat. Per construir-lo seguirem l'esquema de la figura 1.2, ja que tan el PC com el sistema microprocessat són equips del tipus DTE (Data Terminal Equipment) i per tant cal fer servir aquest tipus de cablejat especial. Originalment l'estandard RS-232 només definia conexions de DTE amb DCE (Data Communication Equipment), amb aquest cable podem fer la comunicació del PC amb el sistema microprocessat Olorim sense cap problema, aixi es pot escriure el programa previament compilat amb el PC a la memòria EPROM del sistema Olorim.

Les eines de compilació que es fan servir són les mateixes que les del laboratori de microcontroladors (a8051.exe, xlink.exe)

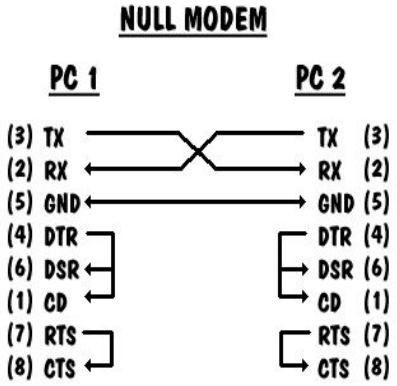


Figura 1.2: Connexions d'un cable Null Modem per RS-232

Capítol 2

Estat de l'art

A la Universitat de Michigan tenen un projecte [2] per muntar una targeta SD a un microcontrolador MSP430 de Texas Instruments. Aquest projecte està fet amb llenguatge C, igual que en el nostre projecte la base d'aquest projecte és llegir i escriure la targeta en format raw.

Hem trobat un altre treball [4] fet amb targetes SD, en aquest cas es tracta d'una placa creada especialment per aquest cas en la qual s'ha soldat un microcontrolador que no s'especifica i s'hi ha soldat un connector SD per poder insertar la targeta i poder procedir a llegir i escriure sobre la targeta.

També hi ha un projecte [5] que es basa en crear un dispositiu basat en un microcontrolador 8051 i afegir-hi també un connector SD per poder fer escriptures i lectures a la targeta.

Finalment cal destacar que en la actualitat aquestes targetes es fan servir en multitud de dispositius tal com càmares fotogràfiques, PDA's, PALM's, GPS's i d'altres dispositius. Actualment la consola de nova generació Wii també disposa d'un lector per a targetes SD, també hi ha impressores que disposen de ranures per targetes SD, CF, MMC, XD i d'altres, així es poden imprimir les fotogràfies directament a la impressora sense necesitat d'utilitzar el PC.

Capítol 3

Fonaments teòrics

Contents

3.1 El 80C552

El microcontrolador 80C552 està fabricat per Philips en un avançat proces CMOS i és un derivat de la familia de microcontroladors 80C51. Pel que fa al joc d'instruccions és el mateix que el del 80C51. Hi ha tres versions d'aquesta familia de microcontroladors:

- 83C552 - Disposa de 8 Kbytes de ROM programable
- 80C552 - Es la versió del 83C552 sense ROM

- 87C552 - Disposa de 8 Kbytes de EPROM

Característique generals:

- CPU de 8 bits
- Processador booleà
- 4 ports de 8 bits
- 256 bytes de memòria interna RAM útil per l'usuari i 384 bytes en total considerant l'àrea dels registres especials (SFR).
- 8 Kbytes de ROM (83C552)
- Espai de memòria de 64K per programes externs
- Espai de memòria de 64K per dades externes
- Conté tres timers
- Comunicació asíncrona full-duplex
- 6 fonts d'interrupcions amb nivell de prioritat
  - 2 Interrupcions externes
  - 3 Interrupcions dels timers
  - 1 Interrupció de la comunicació serie

3.1.1 Organització de la memòria

El microcontrolador 80C552 disposa de 8 Kbytes de memòria ampliables a 64 Kbytes amb memòries externes tal com es veu a la figura: 3.4. Els primers 255 bytes són de memòria RAM (memòria de dades) i estan distribuits de tal manera que de les posicions 0 a la 127 són adresables tan directament com indirectament. De la 127 a la 255 depenent del mode d'adreçament emprat accedim a RAM (indirectament) o al registres SFR (directament). De la 256 fins a la 8191 és memòria ROM (memòria de codi)

La memòria de programa només pot ser llegida i té com a màxim 64KB. La memòria de programes interna es de 8KB. El pin $\overline{EA}$ del microcontrolador quan EA=0 indica que el punter de programes busca direccions desde la posició de memòria 0000H fins la 1FFFH a la memòria interna i de la 2000H fins la FFFFH a la memòria externa. Si EA=1, aleshores sempre busca a la memòria externa.

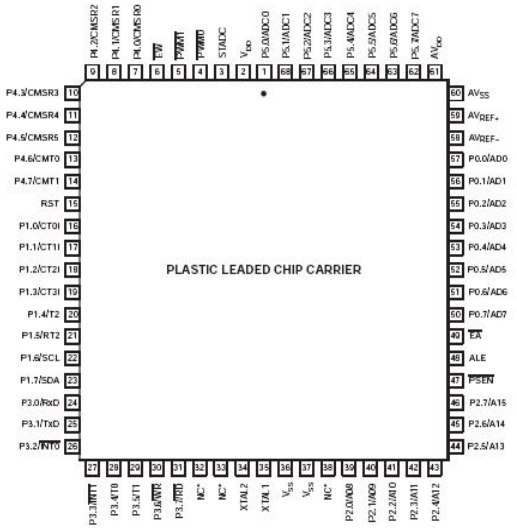


Figura 3.1: 80C552-quadrat

La memòria de dades admet operacions de lectura i escriptura, i com es veu a la figura pot ser interna o externa i pot adreçar fins a 64KB com la memòria de programes. El microprocesador s'encarrega de generar els senyals $\overline{RD}$ (lectura) i $\overline{WR}$ (escriptura) per llegir o escriure a la memòria de dades externa

La memòria de dades i la memòria de programa poden coexistir en el mateix espai d'adreces o en espais separats; en el primer cas es diu que la memòria es combinada i en el segon segregada. La memòria combinada s'obtè aplicant les senyals $\overline{RD}$ i $\overline{PSEN}$ a una porta lògica AND i fent servir la sortida de la porta com un strobe [1] del chip de memòria externa de programes i dades

Memòria de programes

A la part baixa de la memòria de programes es troba la zona dedicada a les rutines d'interrupcions. Com es veu a la figura 3.6 l'espai entre dos interrupcions es de 8 bytes, espai que pot contenir una petita rutina, pero si l'espai no es suficient es pot fer servir una instrucció de salt incondicional, que adreci la interrupció cap a un espai de memòria més ampli.

Com es pot veure a la figura 3.7 les 16 linies d'adreçament corresponen al port 0 i al port 2, el port 0 també serveix com a bus de dades, multiplexat en el temps:

[1] És un pols de sincronització, normalment es fa servir com a entrada del pin chip select

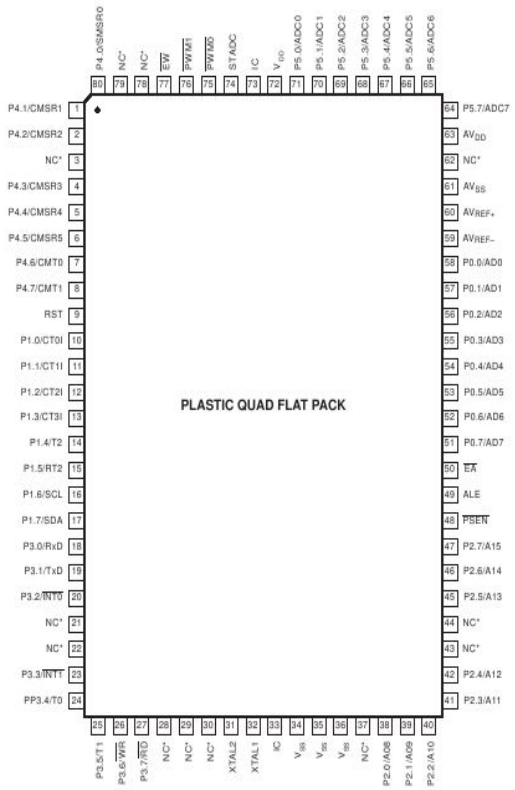


Figura 3.2: 80C552-rectangular

- El port P0 emet el byte baix de l'adreça on es vol accedir. Una vegada establert el senyal sobre P0, el senyal ALE (Address Latch Enable) introdueix aquesta adreça al dispositiu latch, que pasa a apuntar l'adreça de la memòria externa de programes. Al mateix temps que el microcontrolador emet l'adreça baixa per P0, la part alta de l'adreça s'emet per P2. Llavors $\overline{PSEN}$ autoritza la lectura al microcontrolador del codi d'instrucció a través del port 0

Memòria de dades

El 80C552 pot adreçar fins a 64KB de memòria de dades externa. La figura mostra la configuració per accedir als 2KB de memòria de dades externa. En aquest cas la CPU té el programa intern a la memòria ROM. El port 0 multiplexa en el temps adreces i dades de la RAM. La CPU genera els senyals de lectura $\overline{RD}$ i escriptura $\overline{WR}$ que necesita la RAM externa.

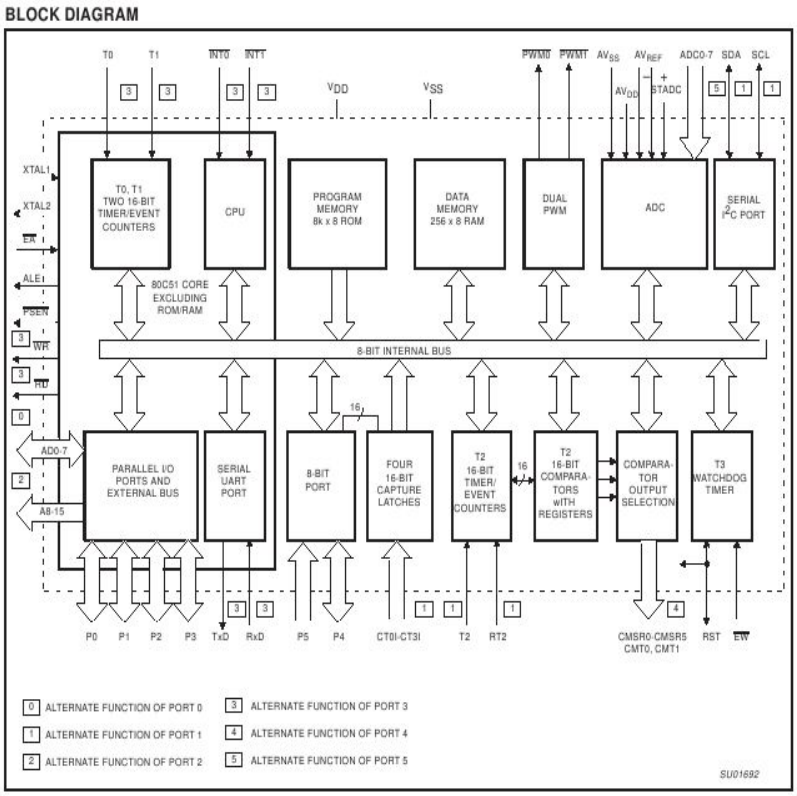


Figura 3.3: Diagrama de blocs

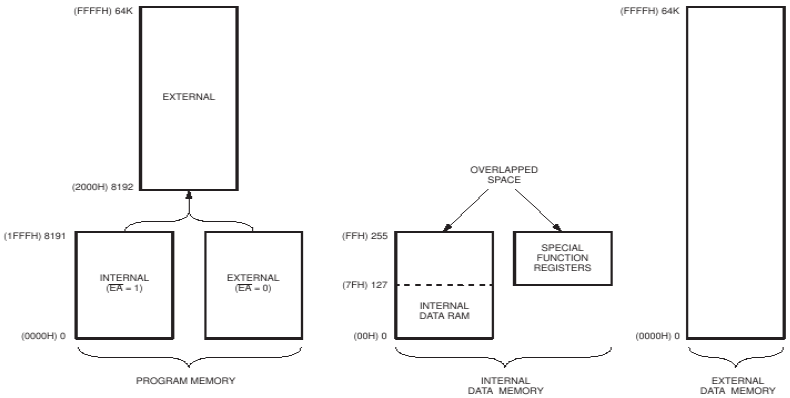


Figura 3.4: Memòria 80C552

A la figura 3.4 es pot veure el mapa de la memòria de dades. La memòria interna es troba dividida en tres blocs, els 128 bytes baixos, els 128 bytes

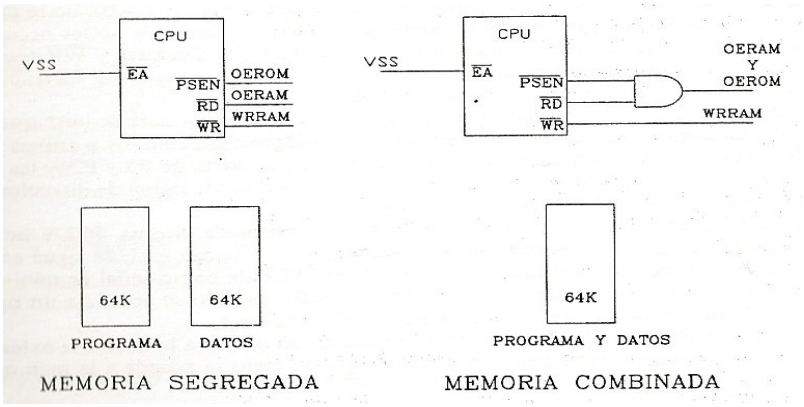


Figura 3.5: Memòria segregada i combinada

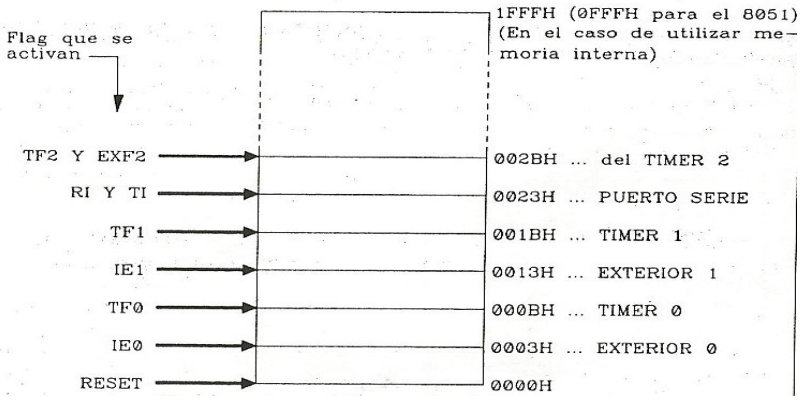


Figura 3.6: Adreça d'interrupcions

alts i l'espai ocupat pels Registres de Funció Especial (SFR). Com es veu a la figura 3.7, la memòria externa es pot adreçar utilitzant les linies d'E/S del port 2, en funció de la quantitat de memòria a adreçar. També es pot utilitzar 2 bytes d'adreces; byte baix al port 0 i byte alt al port 2. Amb aquest sistema es permet adreçar fins a un màxim de 64KB. A la memòria de dades interna es pot accedir a un total de 384 bytes, combinant els blocs de 128 bytes amb els modes d'adreçament.

Àrea de adreçament directe i indirecte

Els 128 bytes (00H a 7FH) als que es pot accedir tan directament com indirectament, poden ser dividits en tres parts (vegeu figura 3.8):

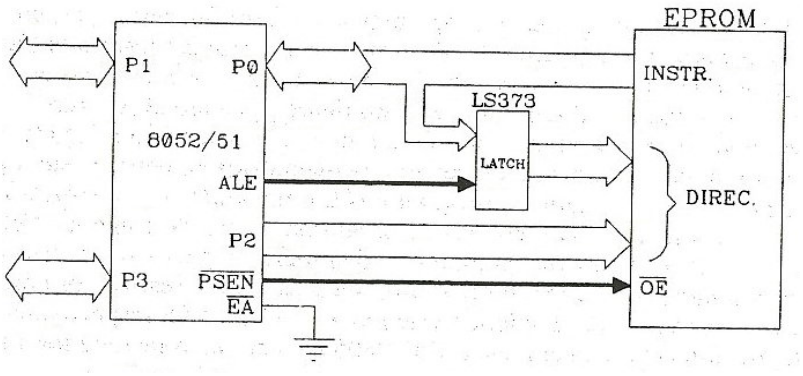


Figura 3.7: Memòria externa

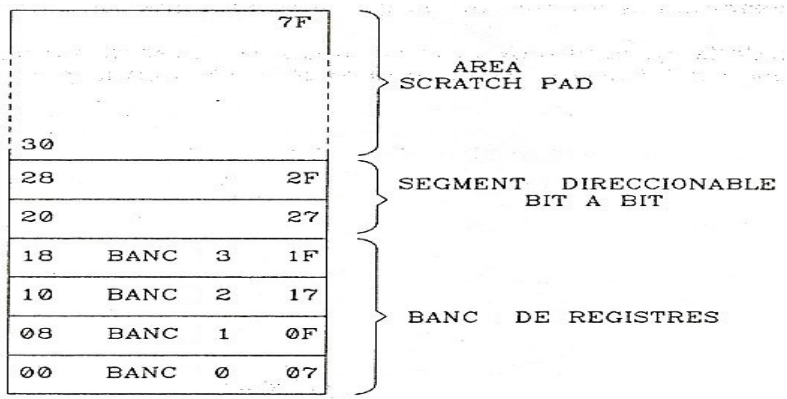


Figura 3.8: Memòria interna

- **Banc de registres**
  Els registres es troben desde la direcció 00H a 1FH (32bytes). Despres de la operació de inicialització del microcontrolador, aplicant un nivell alt al "pin"RESET, el banc operatiu per defecte és el 0. La selecció d'un altre banc de registres es fa per software escribint al registre d'estat PSW.

  El RESET inicialitza l'Stack Pointer (SP) a la posició de memòria 07H i s'incrementa immediatament a la posició de memòria 08H, que es el primer registre R0 del segon banc de registres.

- **Subàrea direccionable bit a bit**
  Aquesta àrea te una longitud de 16 bytes (segment 20H a 2FH). Cadas-

cun dels 128 bits d'aquest segment es poden adreçar directament (00H a 7FH). Els bits es poden referenciar de dos modes diferents, bé per les seves adreces (bits 00H a bits 7FH), o per els bytes que els contenen (20H a 2FH). Els bits del 0 al 7 s'accedeixen aixi: 20.0 a 20.7, i del 8 al F s'hi accedeix aixi: 21.0 a 21.7, etc Cadascun dels 16 bytes d'aquest segment poden ser adreçats com a bytes.

- **Subàrea Scratch Pad**
  La memòria scratch pad s'entèn com una memòria de ràpid accès però d'escasa capacitat. Ocupa les posicions de memòria 30H a 7FH. És la memòria de treball de la (RAM) d'usuari

Àrea d'adreçament només indirecte

A la figura 3.9, s'observa que l'àrea de l'SFR i d'adreçament indirecte (80H a FFH) tenen les mateixes adreces. Però estan separats per camins d'accès diferents, segons l'adreçament de les instruccions. Per exemple, la instrucció:

```
MOV 90H,#ABH
```

escriu AB a la posició de memòria 90H de la memòria RAM de dades i concretament per fer servir l'adreçament directe en el port 1 (90H = P1) de l'àrea SFR. En canvi, les instruccions:

```
MOV R0,#90H
MOV @R0,#ABH
```

escriuen AB a la posició de memòria 90H de la memòria de dades i com utilitzen l'adreçament indirecte escriuen fora de l'SFR, a la figura 3.9 es veu que escriu a la mitja pàgina que es veu darrera l'SFR.

Àrea de registres o SFR

La taula 3.10 mostra els registres especials que utilitza el 80C552 i les seves adreces.

- **ACC Acumulador**: És un registre de pròposit general, i un dels més importants per la seva utilització.
- **B Registre B**: Està especialitzat en les operacions de multiplicació i divisió, igualment es pot fer servir com un registre de pròposit general.
- **PSW Program Status Word**: Conté informació d'estat de la CPU en cada cicle d'instrucció. La taula 3.11 mostra el format de la paraula d'estat.

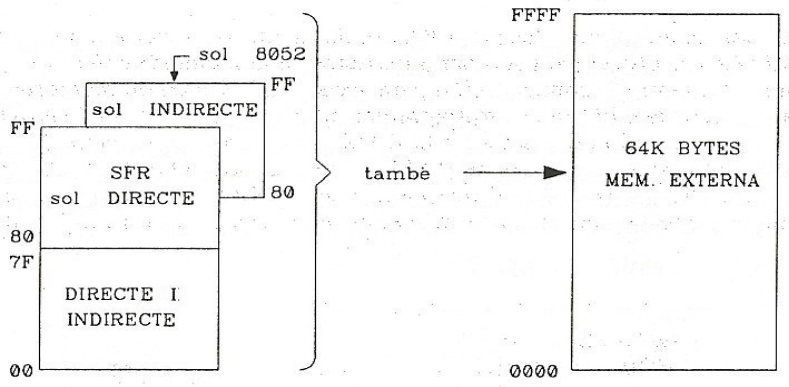


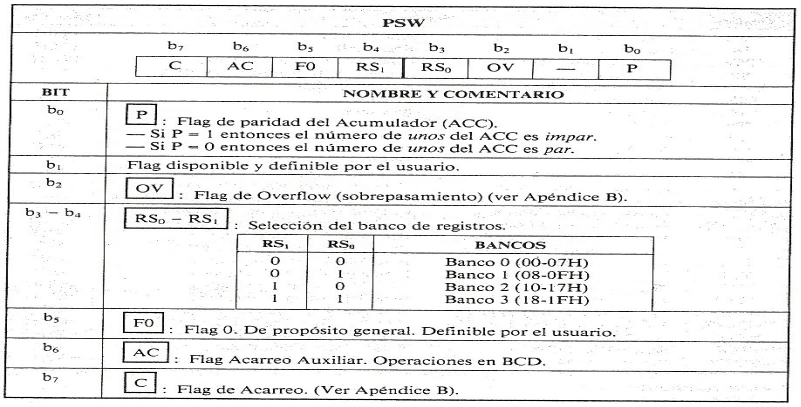
Figura 3.9: Mode d'adreçament

Símbolo	Nombre		Dirección
*ACC	Accumulator	Acumulador	0E0H
*B	B Register	Registro B	0F0H
*PSW	Program Status Word	Palabra de Estado Programa	0D0H
SP	Stack Pointer	Puntero de la memoria de PILA	81H
DPTR	Data Pointer 2 Bytes	Puntero de datos	—
DPL	Low Byte	Byte bajo	82H
DPH	High Byte	Byte alto	83H
*P0	Port 0	*Puerto 0*	80H
*P1	Port 1	*Puerto 1*	90H
*P2	Port 2	*Puerto 2*	0A0H
*P3	Port 3	*Puerto 3*	0B0H
*IP	Interrupt Priority Control	Control Prioridad Interrupciones	0B8H
*IE	Interrupt Enable Control	Control Autorización Interrupciones	0A8H
TMOD	Timer/Counter Mode Control	Control Modo Temporizador/Contador	89H
*TCON	Timer/Counter Control	Control Temporizador/Contador	88H
*+T2CON	Timer/Counter 2 Control	Control Temporizador/Contador 2	0C8H
TH0	Timer/Counter 0 High Byte	Byte Alto Temp./Contador 0	8CH
TL0	Timer/Counter 0 Low Byte	Byte Bajo Temp./Contador 0	8AH
TH1	Timer/Counter 1 High Byte	Byte Alto Temp./Contador 1	8DH
TL1	Timer/Counter 1 Low Byte	Byte Bajo Temp./Contador 1	8BH
+TH2	Timer/Counter 2 High Byte	Byte Alto Temp./Contador 2	0CDH
+TL2	Timer/Counter 2 Low Byte	Byte Bajo Temp./Contador 2	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	Byte Alto Reg. Captura T/C 2	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	Byte Bajo Reg. Captura T/C 2	0CAH
*SCON	Serial Control	Control Comunicaciones Serie	98H
SBUF	Serial Data Buffer	Buffer Datos Comun. Serie	99H
PCON	Power Control	Control Consumo de Potencia	87H

Figura 3.10: Special Function Register

- **SP Stack Pointer**: És un registre de 8 bits, que s'utilitza d'apuntador (apunta a una direcció de memòria). S'inicialitza per defecte amb el valor 07H. Quan s'excuta la primera instrucció PUSH o CALL el SP es carrèga amb el valor 08H

- **DPL y DPH Data Pointer**: El seu propòsit principal es contindre l'adreça del punter de dades. Pot ser manipulat com un registre de 16



PSW

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
C	AC	F0	RS_1	RS_0	OV	—	P

BIT	NOMBRE Y COMENTARIO
b_0	P : Flag de paridad del Acumulador (ACC). — Si P = 1 entonces el número de *unos* del ACC es *impar*. — Si P = 0 entonces el número de *unos* del ACC es *par*.
b_1	Flag disponible y definible por el usuario.
b_2	OV : Flag de Overflow (sobrepasamiento) (ver Apéndice B).
$b_3 - b_4$	$RS_0 - RS_1$: Selección del banco de registros.

RS_1	RS_0	BANCOS
0	0	Banco 0 (00-07H)
0	1	Banco 1 (08-0FH)
1	0	Banco 2 (10-17H)
1	1	Banco 3 (18-1FH)

b_5	F0 : Flag 0. De propósito general. Definible por el usuario.
b_6	AC : Flag Acarreo Auxiliar. Operaciones en BCD.
b_7	C : Flag de Acarreo. (Ver Apéndice B).

Figura 3.11: Program Status Word

bits DPTR o com dos registres de 8 bits

- **P0, P1, P2 i P3**: Són els latchs dels respectius ports.
- **SBUF Serial Data Bufer**: Són dos registres buffer aparentment els mateixos, però fisicament separats, buffer de transmissió i buffer de recepció. Al escriure una dada a l'SBUF, comença a transferirse per la linia corresponent al port serie. En el mode recepció la dada que ingresa per la linea d'entrada del port serie es registra al buffer receptor (SBUF)
- **TH0-TL0, TH1-TL1, TH2-TL2 Timer Registers**: Son parells de registres de 16 bits, que es fan servir com a temporitzadors o contadors.
- **RCAP2H-RCAP2L Capture Registers**: Aquest parell de registres actuen com registres de d'inicialització del timer 2, quan aquest treballa en mode autorecàrrega. En aquest mode, quan es produeix una transició del flanc al pin T2EX, es copien els valors de TH2 i TL2 sobre RCAP2H I RCAP2L, respectivament.
- **PCON Power Control Register**: Per aplicacions on la característica de consum elèctrica sigui crítica, la versió CHMOS ofereix dos modes de treball de baix consum: el mode POWER DOWN i el IDLE. Aquest registre també ofereix la possibilitat de variar la velocitat de comunicació SERIE.
- **TCON Timer/Counter Control Register**: Aquest registre controla el mode d'operació dels Timers 0 i 1 en relació amb les interrupcions i els flancs d'activació de les mateixes.

- **TMOD Timer/Counter Mode Control Register**: Selecciona el Timer 0 o 1, el mode d'operació, si actúa com a temporitzador o com a contador, etc.
- **T2COM Timer/Counter 2 Control Register**: Té la mateixa funció que els registres TCON i TMOD, però per al timer 2
- **IE Interrupt Enable Register**: És un registre per habilitar o deshabilitar les interrupcions. Permet que no s'atengui cap interrupció, o només aquelles que es vulguin activar.
- **IP Interrupt Priority Register**: Cada interrupció pot ser programada individualment en el nivell 1 o 2 de prioritat, posant a 1 o a 0 els bist d'aquest registre. Una interrupció de baix nivell de prioritat pot ser interrompuda per una altre d'un nivell més alt. Una interrupció de nivell alt de prioritat no pot ser interrompuda per una altre interrupció d'un nivell més baix.
- **SCON Serial Control Register**: Aquest registre s'encarrega d'establir els paràmetres per a la transmissió i recepció de dades de la comunicació serie, format de paraula (bit d'start, bits de dades, bit d'stop), velocitat, etc.

A la figura 3.12 es fa refència al mapa de registres SFR. Els registres que estan a la columna esquerra són direccionables bit a bit, són els registres que estan a les posicions 80H, 88H, 90H, ..., F8H, és a dir, els que acaben en 0 o 8.

3.1.2 Ports

Port 0: Multiplexa en el temps en les seves 8 linies la part baixa del bus de direccions durant l'accès a memòria externa de programes i dades, i el bus de dades. El port 0 també rep els bytes de codi durant la programació de la memòria EPROM integrada i surten a través d'ell els bytes de codi durant la verificació de la memòria EPROM o ROM

Port 1: El port 1 també rep la part baixa de direccions, durant la programació i verificació de la memòria EPROM Els bits P1.0 i P1.1 tenen una altra funció també

Port 2: El port 2 emet la part alta del bus de direccions en els accesos a memòria externa quan utilitzen 16 bits de direcció i en els accesos a memòria de dades que fan servir també 16 bits de direcció. Durant l'accès a la memòria de dades externa amb adreçament de 8 bits, els pins del port 2 emeten el

F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8	(T2CON)		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH			PCON	87

Figura 3.12: Mapa de registres

Pins	Funció alternativa
P1.0	T2 (Timer/Comptador 2. Entrada externa)
P1.1	T2EX (Timer/ Comptador 2. Captura i impuls de recarega)

Taula 3.1: Port 1

contingut del registre P2 del SFR El port 2 rep la aprt alta de la direcció, durant la programació i verificació de la memòria EPROM

Port 3: Aquest port té unes funcions especials:

Pins	Funció alternativa
P3.0	RXD (Entrada port serie)
P3.1	TXD (Sortida port serie)
P3.2	$\overline{INT0}$ (Interrupció 0. Externa)
P3.3	$\overline{INT1}$ (Interrupció 1. Externa)
P3.4	T0 (Entrada externa. Timer 0)
P3.5	T1 (Entrada externa. Timer 1)
P3.6	$\overline{WR}$ (Autorització d'escriptura a la memòria de dades externa)
P3.7	$\overline{RD}$ (Autorització de lectura a la memòria de dades externa)

Taula 3.2: Port 3

3.1.3 Modes d'adreçament

Adreçament directe

L'operand s'especifica dins la instrucció per un camp d'adreça de 8 bits. Només la RAM interna de dades (primers 128 bytes) i la zona de SFR es poden adreçar d'aquesta manera.

Exemple:

```
ADD A,3BH
```

Suma el contingut de l'acumulador amb el contingut de la posició de memòria 3BH, deixa el resultat a l'acumulador.

Adreçament indirecte

L'instrucció especifica un registre que conté l'adreça de l'operand. Tant la memòria RAM interna (256 bytes) com l'externa es poden adreçar indirectament. Els registres per adreçar amb 8 bits poden ser R0 i R1 del banc de registres seleccionat, o l'Stack Pointer. El registre per adreçar sobre 16 bits només pot ser el DPTR

Exemple:

```
ADD A,@R0
```

Si R0=3BH, seria exactament la mateixa instrucció que abans, ja que accedeix als primers 128 bytes.

Adreçament immediat

L'operand és directament un valor constant

Exemple:

```
MOV A,#255
```

Escriu a l'acumulador el valor 255

3.2 La targeta microprocessada Olorim

Per dur a terme aquest projecte hem fet servir un sistema de desenvolupament anomenat Olorim el qual va ser desenvolupat per professors de la Universitat de Girona, aquest sistema al llarg de les seves evolucions sempre ha estat basat en microcontroladors de la família '51 de Intel. La placa Olorim disposa de 5 ports, tot i que a nivell d'usuari només es poden fer servir 4, ja que el

port 2 es fa servir per carregar les adreces de les instruccions del programa. Una altra característica interesant són les seves dimensions reduïdes 16cm x 11 cm x 5cm, que fan que sigui fàcil de portar.

Les principals característiques d'aquest sistema són les següents:

CPU:

- 80C552 a 18432MHz

Memòria:

- 32 Kbytes ROM, 32 Kbytes RAM no-volàtil
- Zona combinada que permet la càrrega de programes (veure figura 3.13)

Entrades/Sortides

- Ports 1,3 i 4 del microprocessador totalment accesibles pel usuari
- Port 5 protegit contra sobretensions
- Connexió per dos busos de quatre espais de setze bytes descodificats i protegits/amplificats mitjançant búffers amb adaptació d'impedàncies
- Connexió per un espai extern de vuit Kbytes descodificat

La memòria de l'olorim esta distribuida de la següent manera:

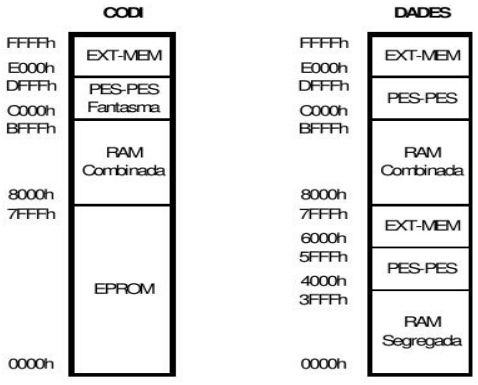


Figura 3.13: Memòria Olorim

Zona EPROM: Espai de 32 Kbytes ubicat des de l'adreça 0000H fins a la 7FFFH de la memòria de Codi. Els codis (programes) acceptats pel carregador han d'estar en format "Hexadecimal Intel Standard". El sistema

accepta dispositius EPROM des del tipus 2764 (8 Kbytes) fins a 27256 (32 Kbytes).

Zona RAM: Espai de 32 Kbytes ubicat en dos espais lògics diferents amb inicis a l'adreça 0000h i 8000h i mida 16 Kbytes. Aquesta configuració ens permet la càrrega de codi. El bloc ïnferior"(0000h fins 3FFFh) és segregat i únicament vist des del espai de dades mentre que el superior (8000h fins BFFFh) es presenta combinat i és visible tant del espai de codi com el de dades. Així qualsevol valor escrit com a dada podrà ser executat. El sistema accepta dispositius RAM des del tipus 2716 (2 Kbytes) fins a 27256.

Zona PES-PES: (en anglès PIO-PIO): Espai ubicat a les zones 4000h fins 5FFFh i C000h fins DFFFh. Aquest espai respon al nom de Entrada Sortida Paral·lel i consta de dos blocs (busos corresponents a J8 i J9) en cadascun dels quals hi ha descodificats quatre "perifèrics"de setze adreces consecutives. Cada vuit blocs de setze adreces són replicats seixanta-quatre cops consecutivament.

Zona EXT-MEM: Espai de 8 Kbytes replicat a dos llocs, amb inicis a les adreces 6000h i E000h. Aquest espai és destinat a la connexió de perifèrics que requereixen "gran finestra de memòria"o bé aplicacions amb "milers de ports"

Per més informació consulteu la bibliografia [12]

3.3 Les memòries SD

Les memòries SD són memòries de tipus flash, és a dir, ens permet escriure i esborrar múltiples posicions de memòria en una sola operació mitjançant polsos elèctrics. En memòries anteriors només es podia escriure o esborrar una única posició cada vegada. Una altra característica important de les memòries flash es que són no volàtils, és a dir, la informació que emmagatzema no es perd quan es desconnecta del corrent.

Les sigles SD venen de Secure Digital, es diu secure ja que Toshiba va afegir un hardware de xifrat per poder complir amb la tecnologia DRM. Actualment com hem comentat anteriorment es fan servir en varis dispositius com camères fotogràfiques digitals, PDA's i Palm's, entre altres.

3.3.1 Característiques de les memòries SD:

- Les targetes SD normals tenen una capacitat màxima de 4GB, venen en formats 128 MB, 256MB, 512MB, 1GB, 2GB i 4GB.
- Les targetes SDHC tenen una capacitat màxima de 32GB, venen en formats de 8GB, 16GB i 32GB

- Compatible amb el protocol SPI (Serial Peripheral Interface) que és un estandard de comunicacions.
- Clock variable 0 - 25 MHZ
- Transferència de fins 100 Mbps, en el mode SD, fent servir les 4 linies de dades.
- Tranferència de fins a 25Mbps en el mode SPI
- Correció d'errors mitjançant CRC (cyclic redundancy check).

3.3.2 Descripció

Les memòries SD tenen dos modes de funcionament: Mode SD i Mode SPI

Mode SD	Mode SPI
Sis canals de comunicació (clock, comanda, 4 línies de dades)	Tres canals de comunciació (clock, dataIn, dataOut)
Protecció d'errors de transferència	Opcional sense protecció de transferència
Simple o multiple blocs de transferència	Simple o multiple blocs de transferència

Taula 3.3: Comparació SPI i SD.

En el mode SD es fan servir quatre línies de dades, una linia de comandes, una de clock, dos de massa i una de suministrament de voltatge.

En el mode SPI es fa servir una línia d'entrada de dades, una línia de sortida de dades, una de CS, una de clock, una de suministrament de voltatge i dos de massa.

[1]S=voltatge;I=Entrada;O=sortida
[2]Les línies DAT1-DAT3 són entrades durant el procès d'engegada
[4]Els pins RSV són d'alta impedància
[5]El registre RCA no està disponible en el mode SPI

n^o Pin	Nom	Tipus[2]	Descripció
1	CD/DAT3[3]	I/O	Card Detect/Data Line [Bit 3]
2	CMD	I/O	Command/Response
3	CMD	S	Supply voltatge ground
4	V_{SS1}	S	Supply voltatge
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltatge ground
7	DAT0	I/O	Data Line [Bit 0]
8	DAT1	I/O	Data Line [Bit 1]
9	DAT 2	I/O	Data Line [Bit 2]

Taula 3.4: Mode SD assignació de Pins

n^o PIN	Nom	Tipus	Descripció
1	CS	I	Chip Select (Active low)
2	DataIn	I	Host to Card Commands and Data
3	VSS1	S	Supply voltatge ground
4	VDD	S	Supply voltatge
5	CLK	I	Clock
6	VSS2	S	Supply voltatge ground
7	DataOut	O	Card to Host Data and Status
8	RSV	I	Reserved
9	RSV[4]	I	Reserved

Taula 3.5: Mode SPI assignació de Pins

Nom	**Bits**	**Descripció**
CID	128	Card Identification Number: número d'identificació diferent a cada tarjeta
RCA[5]	16	Relative Card Address
CSD	128	Card Specific Data: Informació sobre les condicions d'operació de la tarjeta
SCR	64	SD Configuration Register: informació sobre característiques especials de la tarjeta
OCR	32	Operation Condition Register

Taula 3.6: Registres de la tarjeta SD

3.3.3 Com s'escriuen i com es llegeixen?

Hi ha dos formats de lectura/escriptura:

- **Bloc a bloc**
  En aquest mode es pot llegir o escriure un bloc d'una llargada de 512 bytes o inferior. Pot arribar a ser tant petit com un byte. El bloc no pot estar contingut en dos sectors diferents, ha de pertànyer a un sector

- **Multiple bloc**
  En aquest mode es poden llegir varis blocs amb la restricció que han de tenir la mateixa mida i han de ser correlatius, la transferència s'acaba amb la transmissió d'una comanda d'stop

Lectura de la targeta

En el mode SPI la targeta SD suporta els dos formats: bloc a bloc(comanda CMD17) i multiple bloc(comanda CMD18). Després de rebre correctament la comanda de lectura la targeta respon amb un token seguit de les dades de la mida definida previament amb la comanda SET_BLOCK_LENGTH(CMD16). (veieu figura: 3.14)

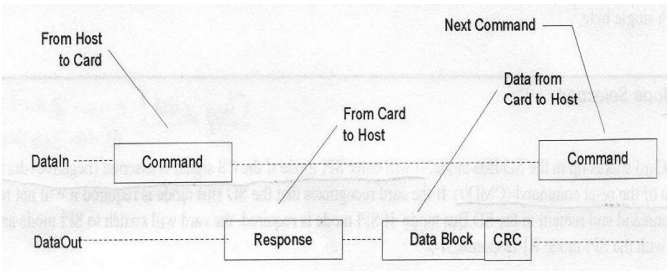


Figura 3.14: Operació de lectura d'un bloc

La mida màxima per un bloc es de 512 bytes tal com està definit al parametre READ_BL_LEN del registre CSD. La mida de bloc pot ser qualsevol valor entre 1 i READ_BL_LEN.

L'adreça inicial pot ser qualssevol valor dintre del rang d'adreces de la targeta. Cada bloc està dintre d'un sector físic de la targeta

En la operació de lectura de multiple bloc, cada bloc enviat conté un sufix de 16 bits amb el CRC. La comanda de parada de transmissió(CMD12), fa el que indica quan es rep aquesta comanda significa que el procès de transmissió s'ha acabat. (veieu figura: 3.15)

Escriptura de la targeta

En el mode SPI la targeta SD suporta els dos formats: bloc a bloc(comanda CMD24) i multiple bloc(comanda CMD25).Despres de rebre correctament la

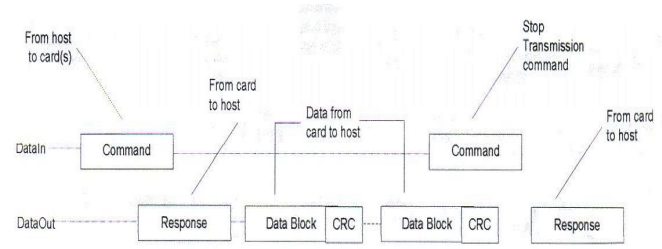


Figura 3.15: Operació de lectura de múltiples blocs

comanda d'escriptura la targeta respon amb un token i queda a l'espera de rebre les dades que s'han d'escriure a la targeta (veieu figura: 3.16). Només es pot fer servir una mida de bloc de 512 bytes. Si es fa servir una mida de bloc més petita es produira un error d'escriptura en la següent comanda d'escriptura.

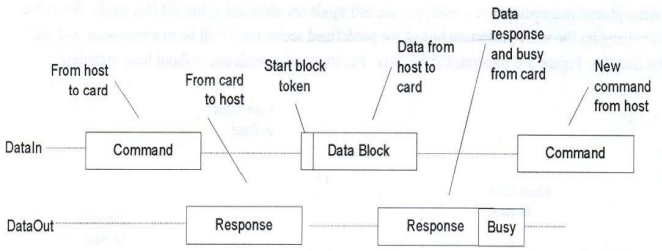


Figura 3.16: Operació d'escriptura d'un bloc

Cada bloc de dades conté un prefix o token d'inici de bloc d'1 byte. Despres de rebre el bloc de dades la targeta respon amb un token de resposta de dades (informa si les dades s'han rebut correctament o no), si el bloc de dades s'ha rebut correctament sense cap error, s'escriu a la targeta el bloc de dades rebut. Mentre la targeta està escrivint les dades, s'envia una trama continua d'ocupat (busy) desde la targeta cap al controlador.

Un cop l'operació d'escriptura finalitza, el controlador revisa el resultat de l'escriptura amb la comanda SEND_STATUS (CMD13). Alguns errors com per exemple adreça fora de rang, protecció contra escriptura, etc. es detecten durant l'escriptura només. L'única validació que és fa al bloc de dades i que es comunica al controlador mitjançant el token de resposta de dades és el CRC.

En l'operació d'escriptura de multiple bloc finalitza la transmissió quan

el controlador envia el token 'Stop tran' en lloc del token 'Start block' al principi del següent bloc. En el cas que el token de resposta de dades indiqui que hi ha hagut un error d'escriptura, el controlador enviara la comanda SEND_NUM_WR_BLOCKS (ACMD22) per saber quants blocs s'han escrit correctament. (veieu figura: 3.17)

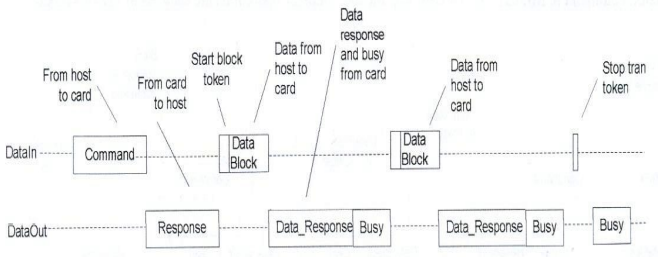


Figura 3.17: Operació d'escriptura de múltiples blocs

Si es reseteja el senyal CS de la targeta mentre està escrivint, això no provoca perdua de dades simplement segueix escrivint. La targeta manté la linia DataOut (tristate) i continua escrivint. Si la targeta es reseleccionada abans que s'hagi acabat d'escriure tota la informació, la linia DataOut es força a l'estat baix i totes les comandes que es rebin són cancelades.

Si es reseteja la targeta mitjançant la comanda (CMD0), si que s'acaba qualssevol feina pendent de la targeta. Això destrueix el format de la targeta. Es responsabilitat del controlador prevenir aquesta situació

Token de resposta de dades

Cada bloc de dades que s'escriu a la targeta és respost amb un token de resposta de dades. Té una mida d'un byte i del bit alt al baix té el següent format:

- 3 bits: Del bit 7 al 5, No importa
- 1 bit: Bit 4, ha de ser 0

- 3 bits: Del bit 3 al 1, bits d'estat
- 1 bit: Bit 0, ha de ser 1

El significat del bits d'estat (status) és el següent:

- '010' - Dada correcte (acceptada)
- '101' - Dada incorrecte degut a un error del CRC
- '110' - Dada refusada degut a un error d'escriptura

En el cas de qualsevol error (error de CRC o d'escriptura) durant l'operació d'escriptura de multiple bloc, el controlador para la transmissió fent servir la comanda CMD12. En el cas d'error d'escriptura (resposta '110') el controlador envia la comanda CMD13 (SEND_STATUS) per saber la causa de l'error d'escriptura.

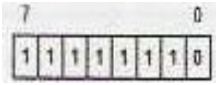
Token de dades

Les comandes de lectura i escriptura fan servir uns tokens per comunicar-se. Les dades s'envien i es reben mitjançant tokens. Tots els bytes de dades s'envien amb el format MSB. Els tokens de dades tenen una mida entre 4 i 515 bytes i tenen el següent format:

Per lectura d'un bloc, escriptura d'un bloc i lectura de multiple bloc:

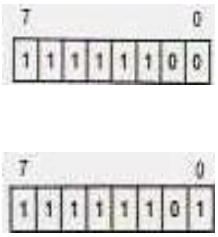
Byte 514	Byte 513 - Byte2	Byte 1 - Byte0
Bloc d'inici	Dades	CRC

- Primer byte: Bloc d'inici (Start bloc)



- Bytes 2-513 (depen de la mida del bloc de dades): Dades
- Últims 2 bytes: 16 bits CRC

Per escriptura de multiple bloc:
Primer byte de cada bloc:
Si s'ha de transmetre una dada aleshores s'envia bloc d'inici:
Si es vol parar la transmissió s'ha d'enviar parada de transmissió (Stop Tran)



Inicialització de la tarjeta

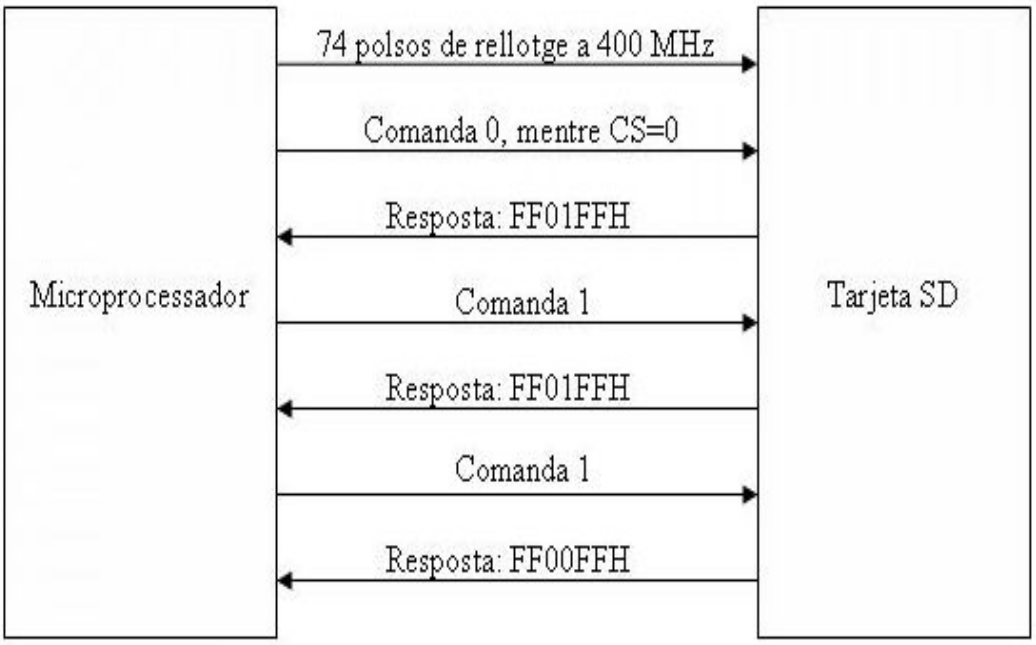


Figura 3.18: Inicialització de la targeta SD

Un cop s'encèn la targeta entra en l'estat idle, per fer-la despertar d'aquest estat s'han d'enviar 74 polsos de rellotge a una freqüència de 400MHz. Això inicialitza els registres interns de la targeta abans de seguir amb el procès d'inicialització. Després per entrar en el mode SPI s'ha d'enviar la comanda CMD0, mentres el bit CS es manté a 0. Tot i que en el mode SPI no hi ha CRC, en el cas d'aquesta primera comanda al no estar encara en mode SPI, cal enviar-ho. El CRC que correspon a aquesta comanda es 95H. Un cop hem despertat la tarjeta es troba en l'estat idle, aleshores cal enviar-li la comanda 1 per inicialitzar la tarjeta. Cal enviar la comanda 1 dos cops, la primera

vegada respon amb FF01FFH, que vol dir comanda rebuda correctament i el segon cop respon amb FF00FF, que vol dir que s'ha fet el reset correctament.

La resposta de la figura 3.19 s'envia desde la targeta després de cada comanda amb l'excepció de les comandes SEND_STATUS, SD_STATUS i READ_OCR. Té una longitud d'un byte, el bit més significatiu sempre és 0 i els altres bits són indicatius d'errors.

- Idle state - La targeta es troba en l'idle state i està en el procès d'iniciatlizació
- Erase reset - La seqüencia d'esborrat s'ha cancelat abans d'executar-se perque s'ha rebut una comanda que que cancelaba l'esborrat
- Illegal command - S'ha detectat un codi de comanda no permés
- Communication CRC error - El CRC de la última comanda no és correcte
- Erase sequence error - S'ha produit un error en la seqüencia d'esborrat
- Address error - L'adreça no és valida, perque no concorda amb el tamany de bloc fet servir a la comanda
- Parameter error - L'argument de la comanda (adreça, mida de bloc,etc) està fora del rang permes per la targeta

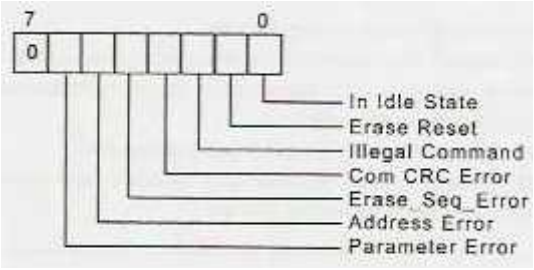


Figura 3.19: Format de resposta R1

La resposta de la figura 3.20 s'envia com a resposta de la comanda SENT_STATUS, té una longitud de 2 bytes. El primer byte és igual al de la resposta R1. El significat del segon byte és el següent:

- Erase param - Selecció incorrecte dels parametres a eliminar

- Write protect violation - La comanda intenta escriure en un bloc protegit contra escriptura
- Card ECC failed - No s'han pogut corretgir les dades
- CC error - Error intern del controlador de targeta
- Error - S'ha produit un error general o desconegut durant l'operació.
- Write protect erase skip - Només s'ha esborrat una part del que es volia degut a que existien blocs protegits contra escriptura
- Card is locked - Suportat per les targetes de SanDisk

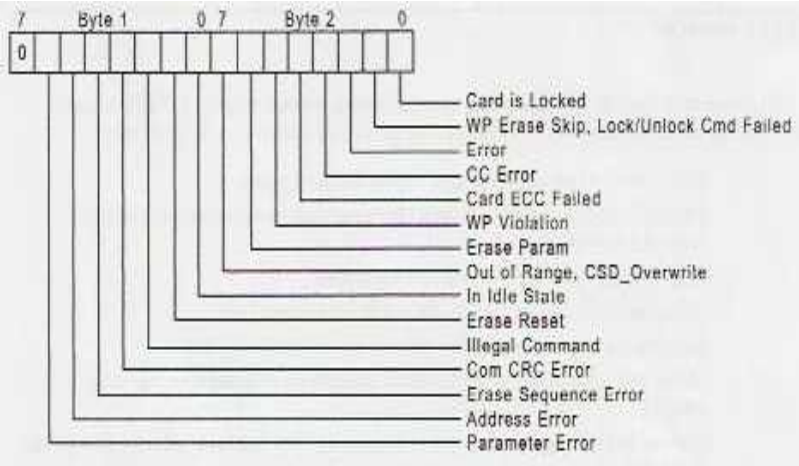


Figura 3.20: Format de resposta R2

La resposta de la figura 3.21 s'envia quan es rep la comanda READ_OCR. La longitud de la resposta és de 5 bytes. L'estructura del primer byte és igual al de la resposta R1. Els altres 4 bytes contenen el registre OCR

Si una operació de lectura falla i la targeta no pot respondre amb les dades, s'envia un token d'error en el seu lloc. Aquest token té una longitud d'un byte i té el format que es mostra a la següent figura 3.22

Totes les comandes de la targeta SD tenen una longitud de 6 bytes i s'envien amb el bit més significatiu primer

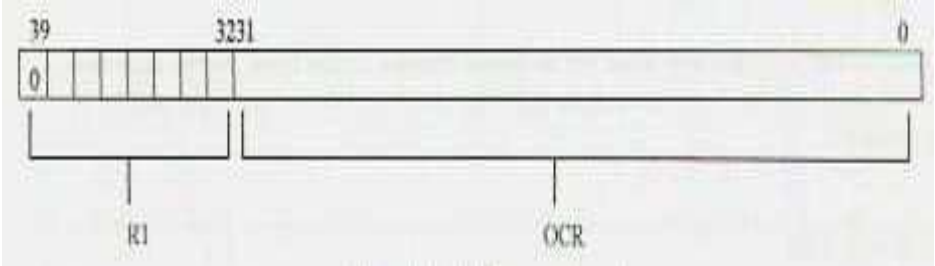


Figura 3.21: Format de resposta R3

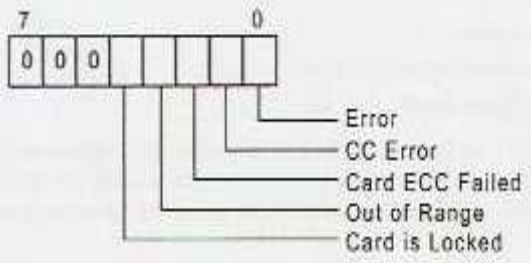


Figura 3.22: Token d'error

Figura 3.23: Format de comanda

3.4 Sistema de fitxers

Els sistemes de fitxers estructuren la informació guardada dins una unitat d'emmagatzematge, que després serà representada de manera textual o gràfica utilitzant un gestor d'arxius. Els sistemes d'arxius defineixen mètodes per crear, moure, renombrar i eliminar arxius i/o directoris. La majoria de sistemes operatius tenen el seu propi sistema de fitxers, per exemple linux fa servir ext2, ext3 i reiser, windows fa servir FAT i NTFS. A part d'quests sistemes de fitxers hi ha també el sistema XFS desenvolupat per Silicon Graphics

i el JFS desenvolupat per IBM i d'altres.

Tot seguit explicarem amb més detall el sistema de fitxers FAT perqué és el que fan servir les targetes SD per defecte.

3.4.1 Sector d'inici i BPB

El sistema de fitxers FAT (File Allocation Table) va neixer a finals dels anys 1970 i principis del 1980, es el sistema de fitxers suportat per el sistema operatiu de Microsoft. Actualment hi ha els següents tipus: FAT12, FAT16 I FAT32. El significat del nombre és per el nombre de bits que fan servir a cada entrada de la FAT.

El sistema de fitxers FAT està format per 4 regions, amb el següent ordre: 0- Regió Reservada 1- Regió FAT 2- Regió de directori arrel (no existeix en FAT32) 3- Regió de fitxers i directoris (dades)

L'estructura de dades més important dins de la FAT és el BPB (BIOS Parameter Block), el qual es troba en el primer sector del volum dins de la regió reservada. Aquest sector es diu "boot sector"o "reserved sector"o "sector 0". Les dades dels 512 bytes de l'estructura del sector 0 representen des de la llargada del sector (hi ha de 512, 1024, 2048 o 4096 bytes) fins al nombre de sectors ocupats per la FAT. Ho detallem a continuació:

Nom	Offset	Tamany	Descripció
BS_jmpBoot	0	3	Instrucció de salt cap al codi d'inici. Pot contenir el següent: jmpBoot[0]=0xEB, jmpBoot[1]=0x??, jmpBoot[2]=0x90 o jmpBoot[0]=0xE9, jmpBoot[1]=0x??, jmpBoot[2]=0x??
BS_OEMName	3	8	"MSWIN4.1"Aquest camp és simplement un string. El sistema operatiu de Microsoft no paren atenció en aquest camp. Tot i que és un camp que no és fa servir si es posa quelcom diferent a "MSWIN4.1", pot donar problemes de compatibilitat
BPB_BytsPerSec	11	2	Contador de bytes per sector. Aquest camp pot pendre els següents valors: 512, 1024, 2048 o 4096. Si es vol tenir la màxima compatibilitat amb antigues implementacions, només es pot fer servir el valor 512.
			Continua a la pàgina següent

Taula 3.7 – continuació de la pàgina anterior

Nom	Offset	Tamany	Descripció
BPB_SecPerClus	13	1	Nombre de sectors per cluster. Aquest valor ha de ser una potencia de 2 més gran que 0. Valors posibles: 1, 2, 4, 8, 16, 32, 64 i 128. Tot i que es possible definir un valor superior a 32K bytes per cluster (BPB_BytsPerSec * BPB_SecPerClus), no és aconsellable ja que es poden trobar problemes de compatibilitat
BPB_RsvdSecCnt	14	2	Nombre de sectors de la regió reservada del volum, els quals començen al primer sector. Per FAT 12 i FAT16 aquest valor no pot ser un altre que 1. Per FAT32 aquest valor normalment es 32
BPB_NumFATs	16	1	És el nombre de regions FAT dins el volum. Aquest camp sempre conté el valor 2 per qualsevol tipus de FAT.La raó perque el valor estàndard d'aquest camp sigui 2 és per poder tenir redundància de dades, ja que si una de les dues FAT's té algun sector defectuos hi ha l'altre FAT per poder recuperar les dades.
BPB_RootEntCnt	17	2	Per volums FAT12 i FAT16, aquest camp conté el nombre d'entrades de directori de 32 bytes cadascuna que hi ha a la regió de directori arrel. Per volums de FAT32, aquest camp s'ha de posar a 0. Per volums FAT12 i FAT16, aquest camp sempre especifica un valor que multiplicat per 32 dona un multiple de BPB_BytsPerSec. Per tenir màxima compatibilitat els volums FAT16 fan servir un valor de 512

Continua a la pàgina següent

Taula 3.7 – continuació de la pàgina anterior

Nom	Offset	Tamany	Descripció
BPB_TotSec16	19	2	Aquest camp és el vell contador de sectors de 16 bits. Aquest camp conté el nombre de sectors del volum. Aquest nombre inclou tots els sectors de les 4 regions del volum. Aquest camp pot ser 0; si es 0, aleshores el camp BPB_TotSec32 no pot ser 0. Per volums FAT32 aquest valor és 0. Per volums FAT12 i FAT16, aquest camp conté el nombre de sectors, i BPB_TotSec32 és 0 si el nombre total de sectors és inferior a 0x10000
BPB_Media	21	1	0xf8 és el valor estandàrd per els medis no extraibles. Per els medis extraibles és 0xF0. Els valors possible per aquest camp són: 0xF0, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE i 0xFF. L'altre cosa important d'aquest camp és que el valor que es posa aqui també ha d'anar al byte baix de la primera entrada FAT[0]
BPB_FATSz16	22	2	Aquest camp conté el nombre de sectors ocupats per una FAT. En volums de FAT32 aquest valor és 0, i BPB_FATSz32 conté el nombre de sectors ocupats
BPB_SecPerTrk	24	2	Aquest camp només té importància per medis amb una determinada geometria (volums partits en pistes per múltiples cilindres i capçals) i són visibles en la interrupció 0x13.
BPB_NumHeads	26	2	Nombre de capçals per la interrupció 0x13. Aquest valor només es important en el mateix cas que el BPB_SecPerTrk. Per exemple, en un disquette de 1,44 MB 3,5-polsades aquest valor es 2.
BPB_HiddSec	28	4	Nombre de sectors ocults que precedeixen la partició que conté la FAT. Aquest camp generalment és important per medis visibles a la interrupció 0x13. Aquest camp ha de ser 0 en medis que no estan particionats.

Continua a la pàgina següent

Nom	Offset	Tamany	Desripció
BS_DrvNum	36	1	Aquest camp es soportat per el bootstrap de MS-DOS i es posa a la INT 0x13 "drive number".
BS_Reserved1	37	1	Reservat (per windows NT)
BS_BootSig	38	1	Extensió de signatura d'iniciv (0x29). Aquest byte de signatura indica que els següents tres camps del "boot sector"són presents.
BS_VolID	39	4	Número de serie del volum. Aquest camp, juntament amb el camp BS_VolLab, suporten rastreig de volum. Aquests valors permeten al sistema de fitxers FAT detectar si s'ha insertat un disc/disquette incorrecte.
BS_VolLab	43	11	Etiqueta de volum. Aquest camp coincideix amb els 11 byte de l'etiqueta de volum guardat en el directori arrel.
BS_FilSysType	54	8	Pot tenir un dels següents valors: "FAT12 ", "FAT16 "o "FAT ". Nota: Aquest camp no té res a veure amb el tipus de format que se li dona al dispositiu.

Taula 3.8: FAT12 i FAT16

Taula 3.7 – continuació de la pàgina anterior

Nom	Offset	Tamany	Descripció
BPB_TotSec32	32	4	Aquest camp és el nou contador de sectors de 32 bits. Aquest nombre inclou tots els sectors de les 4 regions del volum. Aquest camp pot ser 0; si es 0, aleshores BPB_TotSec16 no pot ser 0. Per volums FAT32, aquest camp no pot ser 0. Per volums FAT12 i FAT16, aquest camp conté el nombre de sectors si BPB_TotSec16 és 0 (el nombre és més gran o igual que 0x10000).

Taula 3.7: FAT

En aquest punt, el BPB/boot sector de FAT12 i FAT16 es diferent del BPB/boot sector de FAT32.

Nom	Offset	Mida	Descripció
BPB_FATSz32	36	4	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Aquest camp es el nombre de sectors ocupats per una FAT. BPB_FATSz16 ha de ser 0
BPB_ExtFlags	40	2	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Bits 0-3 – Nombre de la FAT activa. Bits 4-6 – Reservat. Bit 7 – 0 significa que la FAT està duplicada en temps d'execució –1 significa que només una FAT està activa; està referenciada en els bits 0-3. Bits 8-15 – Reservats
BPB_FSVer	42	2	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Aquest és el número de versió del volum FAT32.
BPB_RootClus	44	4	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Aquest valor conté el número de cluster del primer cluster del directori arrel, normalment es 2 però no cal que sigui 2.
BPB_FSInfo	48	2	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Número de sector de l'estructura FSINFO a l'àrea reservada del volum FAT32
BPB_BkBootSec	50	2	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Si no es 0, indica el nombre de sector de l'àrea reservada del volum. Normalment és 6. No és recomant un altre valor que no sigui 6
BPB_Reserved	52	12	Aquest camp només es fa servir per volums FAT32 i no existeix en volums FAT12 i FAT16. Reservat per futures extensions.

Continua a la pàgina següent

Taula 3.9 – continuació de la pàgina anterior

Nom	Offset	Tamany	Descripció
BS_DrvNum	64	1	Aquest camp té la mateixa definició que en els volums FAT12 i FAT16. L'única diferencia és que està en un offset diferent dins el sector d'inici.
BS_Reserved1	65	1	Aquest camp té la mateixa definició que en els volums FAT12 i FAT16. L'única diferencia és que està en un offset diferent dins el sector d'inici.
BS_BootSig	66	1	Aquest camp té la mateixa definició que en els volums FAT12 i FAT16. L'única diferencia és que està en un offset diferent dins el sector d'inici.
BS_VolID	67	4	Aquest camp té la mateixa definició que en els volums FAT12 i FAT16. L'única diferencia és que està en un offset diferent dins el sector d'inici.
BS_VolLab	71	11	Aquest camp té la mateixa definició que en els volums FAT12 i FAT16. L'única diferencia és que està en un offset diferent dins el sector d'inici.
BS_FilSysType	82	8	Sempre conté l'string "FAT32 ".

Taula 3.9: FAT32

Una altra cosa important del sector 0 de la FAT és que els dos últims bytes són: sector[510]=0x55 i sector[511]=0xAA.

3.4.2 Estructura de dades FAT

La següent estructura de dades important es la mateixa FAT. La FAT està definida com una llista d'apuntadors cap als fitxers.

El primer sector del cluster 2 (regió de dades) es calcula fent servir els camps BPB del volum. Primer, determinem el nombre de sectors ocupats per el directori arrel:

$$RootDirSectors = \frac{(BPB\_RootEntCnt * 32) + (BPB\_BytsPerSec - 1)}{BPB\_BytsPerSec}$$

S'ha d'observar que en els volums FAT32, el camp BPB_RootEntCnt és

sempre 0, per tant en volums FAT32 RootDirSectors sempre sera 0.
El càlcul de RootDirSectors s'ha d'arrodonir cap a dalt.
Perque aquest càlculs siguin més entenedors a partir d'aquest moment farem servir com a exemple una targeta SD de 32Mb:

$$\frac{(512 * 32) + (512 - 1)}{512} = 33$$

L'inici de la regió de dades, el primer sector del cluster 2, es calcula de la següent manera:

```
If (BPB\_FATSz16 != 0)
FATSz = BPB\_FATSz16;
Else
FATSz = BPB\_FATSz32;
```

$$FirstDataSector = BPB\_ResvdSecCnt + (BPB\_NumFATs * FATSz) + RootDirSectors$$

Exemple:

$$2 + (2 * 243) + 33 = 521$$

Donat qualssevol nombre de cluster N, el nombre de sector del primer sector del cluster es calcula de la següent manera:

$$FirstSectorofCluster = ((N - 2) * BPB\_SecPerClus) + FirstDataSector$$

Per exemple volem saber a quin sector està el cluster 6:

$$((6 - 2) * 1) + 521 = 525$$

3.4.3 Determinar el tipus de FAT

El tipus de FAT: FAT12, FAT16 o FAT32 es determina per el nombre de clusters en el volum i res més. Per determinar el nombre de clusters del volum es fan servir els camps BPB del volum. Primer, es calcula el nombre de sectors ocupats per el directori arrel com hem vist abans

$$RootDirSectors = \frac{(BPB\_RootEntCnt * 32) + (BPB\_BytsPerSec - 1)}{BPB\_BytsPerSec}$$

Despres es calcula el nombre de sectors de la regió de dades del volum:

```
 If (BPB\_FATSz16 != 0)
FATSz = BPB\_FATSz16;
```

```
Else
FATSz = BPB\_FATSz32;
If (BPB\_TotSec16 != 0)
TotSec = BPB\_TotSec16;
Else
TotSec = BPB\_TotSec32;
```

$DataSec = TotSec - (BPB\_ResvdSecCnt + (BPB\_NumFATs * FATSz) + RootDirSectors);$

Exemple:

$$62720 - (2 + (2 * 243) + 33) = 62199$$

Ara ja es pot determinar el nombre de clusters:

$$CountofClusters = DataSec/BPB\_SecPerClus;$$

CountofClusters s'arrodoneix cap avall Exemple:

$$\frac{62199}{1} = 62199$$

Ara es pot determinar el tipus de FAT

```
If (CountofClusters < 4085) {
/* El volum és FAT12 */
} else if (CountofClusters < 65525) {
/* El volum és FAT16 */
} else {
/* El volum és FAT32 */
}
```

Aquesta és l'única via per determinar el tipus de FAT. No existeixen volums FAT12 que tinguin més de 4084 clusters. No existeixen volums FAT16 que tinguin menys de 4085 i més de 65524 clusters. No existeixen volums FAT32 que tinguin menys de 65525 clusters. Si es fa un volum FAT que violi aquestes regles, el sistema operatiu de microsoft no llegiria correctament el dispositiu, perque creuria que el dispositiu té un tipus de FAT diferent al que està fent servir.
S'ha de veure també que el valor CountofClusters es exactament el nombre de "data clusters" començant pel cluster 2. El nombre màxim de clusters del volum es CountofClusters + 1, i el nombre de clusters inclosos els dos clusters reservats és CountofClusters + 2. Per trobar a quina part de la FAT es troba l'entrada per un cluster N, es fa servir el següent algoritme.

```
If (BPB_FATSz16 != 0)
```

```
FATSz = BPB_FATSz16;
Else
FATSz = BPB_FATSz32;

If (FATType == FAT16)
FATOffset = N*2;
Else if (FATType == FAT32)
FATOffset = N*4;

FATSecNum = BPB_ResvdSecCnt + (FATOffset / BPB_BytsPerSec);
FATEntOffset = REM(FATOffset / BPB_BytsPerSec);
```

REM(...) és l'operador del residu.El valor FATSecNum és el nombre de sector de la FAT que conté l'entrada per al cluster N en la primera FAT. Si es vol saber el nombre de sector per la segona FAT només cal sumar el valor FATSz al valor obtingut a FATSecNum.

El codi per la FAT12 és més complicat ja que hi ha 1,5 bytes per cada entrada de la FAT

```
If (FATType == FAT12)
FATOffset = N + (N/2);
/* Multipliquem per 1,5 sense fer servir coma flotant,
la divisió per 2 s'arrodoneix cap avall */
ThisFATSecNum = BPB_ResvdSecCnt + (FATOffset / BPB_BytsPerSec);
ThisFATEntOffset = REM(FATOffset / BPB_BytsPerSec);

If (ThisFATEntOffset == (BPB_BytsPerSec - 1)) {
/* Aquest cluster sobrepasa el limit de la FAT. Hi han varies estrategies per
solucionar això. El més fàcil és carregar els sectors de la FAT per parells
sempre que el volum sigui FAT12 (si es vol carregar el sector N de la FAT,
també s'ha de carregar el sector N + 1 a no ser que el sector N sigui l'últim
sector de la FAT). */
}
```

3.4.4 Inicialització del volum FAT

El sistema operatiu de microsoft formatega els discs menors de 512MB amb FAT16, per defecte, i els discs superiors a 512MB amb FAT32

```
struct DSKSZTOSECPERCLUS {
DWORD DiskSize;
BYTE SecPerClusVal;
```

```
};

/* Aquesta és la taula per el format FAT16 */
DSKSZTOSECPERCLUS DskTableFAT16 [] = {
{8400,0}, /* disc fins a 4.1MB, el valor 0 de SecPerClusVal dona error */
{32680,2}, /* disc fins a 16MB, 1k cluster */
{262144,4}, /* disc fins a 128MB, 2k cluster */
{524288,8}, /* disc fins a 256MB, 4k cluster */
{1048576,16}, /* disc fins a 512MB, 8k cluster */
/* Les següents entrades no es fan servir a menys que es forci el format FAT16 */
{2097152,32}, /* disc fins a 1GB, 16k cluster */
{4194304,64}, /* disc fins a 2GB, 32k cluster */
{0xFFFFFFFF,0}, /* qualssevol disc més gran de 2GB, el valor 0 de SecPerClusVal dona
};
/* Aquesta és la taula per el format FAT32 */
DSKSZTOSECPERCLUS DskTableFAT32 [] = {
{66600,0}, /* disc fins a 32,5MB, el valor 0 de SecPerClusVal dona error */
{532480,1}, /* disc fins a 260MB, 0,5 cluster */
{16777216,8}, /* disc fins a 8GB, 4k cluster */
{33554432,16}, /* disc fins a 16GB, 8k cluster */
{67108864,32}, /* disc fina a 32GB, 16k cluster */
{0xFFFFFFFF,64} /* disc més gran de 32GB, 32k cluster */
};
```

Donada una mida de disc i un tipus de FAT FAT16 o FAT32 podem trobar el valor per BPB_SecPerClus.

L'última cosa que falta és calcular quants sectors ocupara la FAT i per tant quin és el valor de BPB_FATSz16 o BPB_FATSz32.

```
 RootDirSectors = ((BPB_RootEntCnt * 32) + (BPB_BytsPerSec - 1)) / BPB_BytsPerSec;
TmpVal1 = DskSize - (BPB_ResvSecCnt + RootDirSectors);
TmpVal2 = (256 * BPB_SecPerClus) + BPB_NumFATs;
If (FATType == FAT32)
TmpVal2 = TmpVal2 /2;
FATSz = (TMPVal1 + (TmpVal2 - 1)) / TmpVal2;
If (FATType == FAT32) {
BPB_FATSz16 = 0;
BPB_FATSz32 = FATSz;
} else {
BPB_FATSz16 = LOWORD (FATSz);
/* no hi ha BPB_FATSz32 en un volum FAT */
}
```

3.4.5 Estructura de directori FAT

El directori FAT es com un arxiu composat d'una llista lineal de 32 bytes cada component. L'únic directori especial, que sempre ha de ser present es el directori root. Per FAT12 i FAT16, el directori root esta ubicat en una posició fixa del disc inmediatament despres de l'última FAT. Per FAT12 i FAT16, el primer sector del directori root es un nombre de sector relatiu al primer sector del volum FAT:

$$FirstRootDirSecNum = BPB\_ResvdSecCnt + (BPB\_NumFATs * BPB\_FATSz16);$$

Per FAT32, el directori root pot ser de tamany variable i es una cadena de clusters. El primer cluster del directori root en un volum FAT32 es guarda a BPB_RootClus.

Nom	Offset	Tamany	Descripció
DIR_Name	0	11	Nom curt
DIR_Attr	11	1	Atributs de fitxer: ATTR_READ_ONLY 0x01 ATTR_HIDDEN 0x02 ATTR_SYSTEM 0x04 ATTR_VOLUME_ID 0x08 ATTR_DIRECTORY 0x10 ATTR_ARCHIVE 0x20 ATTR_LONG_NAME ATTR_READ_ONLY | ATTR_HIDDEN | ATTR_SYSTEM | ATTR_VOLUME_ID Els dos bits alts del byte d'atributs estan reservats i han d'estar sempre a 0.
DIR_NTRes	12	1	Reservat per l'ús de Windows NT. Posar el valor a 0 quan es crea el fitxer i no modificar.
DIR_CrtTimeTenth	13	1	Aquest camp conté les dècimes de segon. La precisió del camp DIR_CrtTime és de 2 segons, per tant aquest camp representa les decimes d'un segon i el seu rang es de 0-199 inclosos
DIR_CrtTime	14	2	Hora de creació del fitxer
DIR_CrtDate	16	2	Data de creació del fitxer
DIR_LstAccDate	18	2	Data de l'últim accès. Aqui no s'indica a quina hora va ser accedit el fitxer sino només el dia
DIR_FstClusHI	20	2	Part alta del primer cluster d'aquest fitxer (sempre és 0 per FAT12 i FAT16).
DIR_WrtTime	22	2	Hora de l'última escriptura
DIR_WrtDate	24	2	Data de l'última escriptura
DIR_FstClusLO	26	2	Part baixa del primer cluster d'aquest fitxer
DIR_FileSize	28	4	Conté el tamany de fitxer en bytes

El primer byte de cada directori té un significat especial:
DIR_Name[0] == 0xE5, aleshores l'entrada de directori és buida.
DIR_Name[0] == 0x00, aleshores l'entrada de directori es buida i no hi ha cap entrada de directori assignada despres (totes les següents entrades DIR_Name[0] s'han de posar a 0) El valor especial 0, enlloc de 0xE5, indica al driver del sistema de fitxers FAT que la resta d'entrades en aquest directori no necesiten ser mirades ja que són totes buides.
DIR_Name[0] == 0x05, vol dir que l'actual caràcter d'aquest byte es 0xE5. 0xE5 es actualment un KANJI de l'abecedari japonès. El valor 0x05 es fa

servir per aquest caràcter japonès enlloc del 0xE5, així no hi ha problemes d'interpretació per part del driver de la FAT i que cregui que aquesta entrada està lliure.

Els següents caràcters no són vàlids per al valor DIR_Name:

1. Valors més petits que 0x20, excepte en el cas especial descrit abans del 0x05

2. 0x22, 0x2A, 0x2B, 0x2C, 0x2E, 0x2F, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x5B, 0x5C, 0x5D i 0x7C.

El camp DIR_Attr de la taula anterior especifica atributs del fitxer:

ATTR_READ_ONLY	Indica que no es pot escriure.
ATTR_HIDDEN	Indica que quan es llisten els directoris aquest no apareix.
ATTR_SYSTEM	Indica que es un fitxer de sistema.
ATTR_VOLUME_ID	Només hi ha un fitxer en el volum que tingui aquest atribut i aquest fitxer ha d'estar al directori root. El nom d'aquest fitxer és l'etiqueta del volum.
ATTR_DIRECTORY	Indica que aquest fitxer és un contenidor d'altres fitxers.
ATTR_ARCHIVE	Aquest atribut dona suport a les eines de backup. Aquest bit es activat pel driver de sistema quan un fitxer es creat, modificat o renombrat. Les eines de backup utilitzen aquest atribut per saber si el fitxer ha estat modificat des de l'última vegada que és va fer el backup.

Quan es crea un directori, fitxer que té l'atribut ATTR_DIRECTORY activat, s'ha de posar el camp DIR_FileSize a 0. DIR_FileSize no es fa servir i és sempre 0 en un directori (la mida de directori es calcula seguint la cadena de cluster fins trobar la marca EOC). Ara cal posar als camps DIR_FstClusLO i DIR_FstClusHI el cluster de la FAT que conté el directori, i posar-hi la marca EOC al cluster de la FAT. Si el directori és el directori arrel no cal fer res més. Però si el directori no és el directori arrel, cal crear dues entrades especials, l'entrada punt i la puntpunt.

El camp DIR_FileSize d'ambdues entrades s'ha de posar a 0, i en tots els camps de temps i dates de les dues entrades s'ha de posar els mateixos valors que el directori que acabem de crear. Ara cal posar als camps DIR_FstClusLO i DIR_FstClusHi de l'entrada punt els mateixos valors que té el directori que acabem de crear. Finalment cal posar els valors

DIR_FstClusLO i DIR_FstClusHI de l'entrada puntpunt, aquest valor és el primer cluster del directori que conté el directori que acabem de crear.

Format de data i temps

Format de data
La data és un camp de 16 bits, la data és relativa a la data 01/01/1980.
Bits 0-4: Dia, valors vàlids 1-31 inclusius.
Bits 5-8: Mes, 1= Gener, valors vàlids 1-12 inclusius.
Bits 9-15: Anys desde 1980, valors vàlids 0-127 inclusius (1980-2107).

Format de temps
El temps és un camp de 16 bits i té una precisió de 2 segons.
Bits 0-4: segons, valors vàlids 0-29 inclusius (0-58 segons).
Bits 5-10: minuts, valors vàlids 0-59 inclusius.
Bits 11-15: hores, valors vàlids 0-23 inclusius.

3.4.6 Entrades de directoris llargs

Les entrades llargues de directori són definides com entrades curtes de directori amb un atribut especial:

```
ATTR_LONG_NAME   ATTR_READ_ONLY |
                 ATTR_HIDDEN |
                 ATTR_SYSTEM |
                 ATTR_VOLUME_ID
```

Es fa servir una màscara per saber si una entrada curta de directori té sub-components d'entrades llargues de directori:

```
ATTR_LONG_NAME_MASK   ATTR_READ_ONLY |
                      ATTR_HIDDEN |
                      ATTR_SYSTEM |
                      ATTR_VOLUME_ID |
                      ATTR_DIRECTOY |
                      ATTR_ARCHIVE
```

Nom	Offset	Tamany	Descripció
LDIR_Ord	0	1	Aquest nombre marca l'ordre a seguir per composar el nom llarg. Si està enmascarada amb 0x40, indica que és l'última entrada de directori llarg dins la seqüencia d'entrades de directoris llargs.
LDIR_Name1	1	10	Caràcters de l'1 al 5 del nom de fitxer
LDIR_Attr	11	1	Atributs, ATTR_LONG_NAME
LDIR_Type	12	1	Si és 0, indica una entrada de directori que és un subcomponent d'un nom llarg
LDIR_Chksum	13	1	checksum del nom de l'entrada de directori curta.
LDIR_Name2	14	12	Caràcters del 6 al 11 del nom de fitxer
LDIR_FstClusLO	26	2	Ha de ser 0.
LDIR_Name3	28	4	Caràcters 12 i 13 del nom de fitxer.

Associació de noms de directori curts i llargs

Un conjunt d'entrades de directori llargues sempre precedeixen a una entrada de directori curta associada. Les entrades llargues estan associades a les entrades curtes perque les versions previes de MS-DOS/Windows només veuen les entrades curtes. Sense una entrada curta associada, una entrada de directori llarga és invisible per una versió previa de MS-DOS/Windows. Les entrades de directori llargues sempre precedeixen i són contigues a l'entrada curta a la qual estan associades.

Cada membre del conjunt d'entrades de directori llargues té un nombre únic. El camp LDIR_Ord és el que s'encarrega de numerar-les, l'última entrada del conjunt té el nombre que li correspon dins la seqüència, més una màscara 0x40 indicant que és l'última entrada. Els 8 bits del checksum es calculen mitjançant el nom de fitxer de l'entrada curta. Es fan servir els 11 caràcters del nom de fitxer dins de l'entrada curta. El checksum es posa dins de cada entrada llarga de directori. Si algun checksum de la cadena d'entrades de directori llargues no coincideix amb el checksum de l'entrada curta, aleshores les entrades llargues són tractades com a orfes. Aixo pasa quan un disc amb entrades curtes i llargues es tractat per una versió anterior de MS-DOS/Windows i només es modifica el nom de l'entrada curta.

Emmagatzemament de noms llargs

Un nom llarg consisteix en un nom que conté més caràcters dels que hi caben en una entrada llarga. La següent figura 3.24 mostra com es guarda un

nom llarg en varies entrades de directori llargues. Els noms acaben amb un caràcter NUL seguit de 0xFFFF; sempre i quan no siguin múltiples de 13, en aquest cas els caràcters del nom ocuparien totes les posicions dedicades al nom i no caldria posar-hi el caràcter NUL ni cap 0xFFFF.

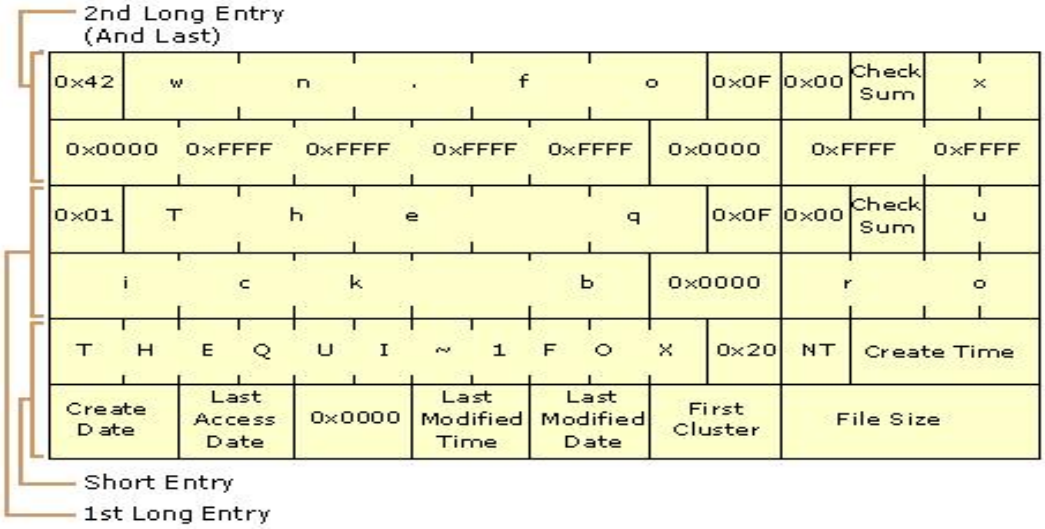


Figura 3.24: Emmagatzemar noms llargs

Capítol 4

Part Pràctica

Contents

4.1 Hardware

Per muntar la targeta SD a la placa Olorim, cal un conector SD, en el nostre cas al no trobar en el mercat cap connector SD, vam desmontar un adaptador USB d'aquests que suporten varis formats (MMC,CF,SD,MSPro,MiniSD). Un cop tenim l'slot SD, cal agafar un cable bus i soldar els pins de la targeta que es faran servir 4.1. Entre l'alimentació positiva que surt de la placa Olorim i el pin 4 (+) de la targeta, cal posar 3 diodes per baixar el voltatge de 5v que suministra la placa, per poder adequar-lo al voltatge d'entrada que necesita la targeta SD (2,7-3,6v). Al mercat hi ha targetes SD de 7 pins i de 9 pins, en el cas que sigui de 9 pins, cal que els 2 pins adicionals estiguin alimentats a 1, ja que no es poden trobar en alta impedancia.

Nº PIN	Nom	Port
1	CS	P1.1
2	DataIn	P1.2
3	VSS	-
4	VDD	+
5	CLK	P1.0
6	VSS	-
7	DAtaOut	P1.3

Taula 4.1: Connexió targeta SD

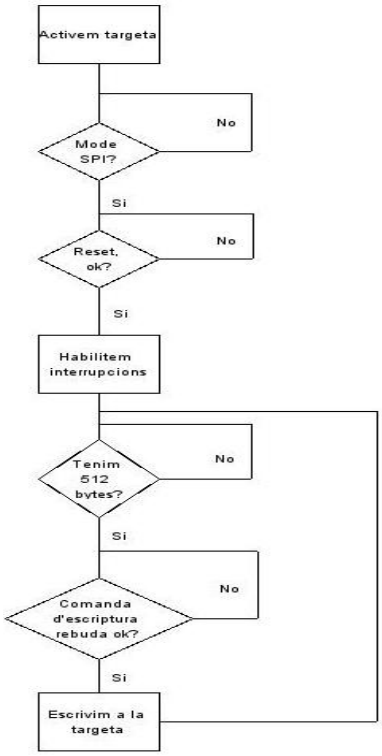


Figura 4.1: Diagrama de flux

4.2 Software

L'estructura bàsica del programa que ens vam plantejar desde un principi estaba composta per primer de tot inicialitzar la targeta SD, un cop aquesta

part estigues enllestida caldria guardar a memòria les dades que ens arribesin del sistema de captura. Com que aquest últim punt pertanya a un altre projecte es va optar per fer una simulació d'aquest, la simulació consisteix a entrar les dades directament desde el PC via interrupció serie i anar-les emmagatzemant a memòria a la posició 1000H fins a la posició 1800H. Un cop ja tinguessim les dades a memòria calia implementar la funció que s'encarregaria d'escriure les dades contingudes a memòria cap a la targeta SD.

El funcionament del programa es el següent, un cop s'ha inicialitzat la targeta SD, s'inicialitzen i activen les interrupcions i s'inicialitzen les dades necesaries per fer els punters a memòria. S'ha implementat una cua circular mitjançant dos punters a memòria ADDRL i ADDRE, en aquest cas hem pogut prescindir de bits de control, ja que després dels càlculs efectuats, tal com es pot veure a l'apartat 4.3 temps d'execució de la pàgina 71, hem vist que l'escriptura a la targeta SD era 4 vegades més ràpida que la recepció de dades per part del sistema de captura.

Per implementar aquesta cua circular hem generat un buffer de 2 Kbytes, tenint en compte que amb 2 Kbytes tenim 4 blocs de 512 bytes, que és més que suficient tal com hem pogut constatar amb els temps d'execució que hem obtingut.

Un cop ja hem definit la base del programa, veurem amb més deteniment com hem implementat la cua circular. Primer de tot tenim la interrupció serie que és l'encarregada d'emmagatzemar dins el buffer de 2 Kbytes les dades que ens van arribant pel port serie, com veurem més endavant el buffer està compres entre la posició 1000H a la 17FFH, aquesta mateixa interrupció va incrementant l'adreça ADDRL que és el punter que indica a quina posició s'ha d'emmagatzemar la dada, un cop aquest punter pren el valor 1800H immediatament es sobreescriu amb el valor 1000H.

La interrupció serie disposa d'un comptador (el registre R1) que s'encarrega d'anar comptant les dades que es guarden a memòria, un cop aquest comptador ha comptat dues vegades fins a 256 [1] s'inicialitza a 0 i es crida a la funció WRITEBLK, que s'encarregarà d'escriure a la targeta l'últim bloc de 512 bytes disponible.

La funció WRITEBLK, que més endavant s'explicara més a fons, escriu el que hi ha a la memòria del microcontrolador cap a la targeta SD. Aquesta funció disposara d'un punter sobre l'estructura circular aquest punter és el ADDRE. La funció WRITEBLK escriu a la targeta SD en blocs de 512 bytes. El punter, un cop s'ha escrit un bloc de 512 bytes, s'haurà d'incrementar en 200H. Tal com passa amb el punter ADDRL, quan ADDRE prengui el valor 1800H immediatament es sobreescriura amb el valor 1000H, per tal d'apuntar

[1] El registre que fem servir es de 8 bits per tant no podem comptar fins a 512

de nou al principi de l'estructura circular.

Abans de tot cal definir, les variables que farem servir. La variable ENVIA1 tal com indica es fa servir per enviar un 1 cap a la targeta, la diferencia entre A i B, es que A envia amb CLK=1 i B amb CLK=0. D'aquesta manera generem el clock, es fa d'aquesta manera per estalviar el timer, per si cal fer-ho servir per futures ampliacions. Una altra raó per fer-ho sense timer es per estalviar codi i maldecaps amb les interrupcions, d'aquesta manera saps que el programa té un flux d'execució ininterromput. Per rebre fem el mateix creem dues variables REP_A amb CLK=1 i l'altre REP_B amb CLK=0

```
ENVIA1_A EQU 0DH; CLK,DI i DO=1   CS=0
ENVIA1_B EQU 0CH; CLK,CS=0   DI,DO=1
ENVIA0_A EQU 01H; CLK=1  CS,DI,DO=0
ENVIA0_B EQU 00H; CLK,CS,DI,DO=0

REP_A EQU 0DH; CLK,DI i DO=1   CS=0
REP_B EQU 0CH; CLK,CS=0   DI,DO=1
```

Ara cal definir les variables que farem servir com a punters a memòria. L'adreça ADDRL és l'adreça on es va emmagatzemant el que arriba pel port serie, aquesta adreça es va incrementant cada vegada que arriba un caracter pel port serie (el control d'aquesta adreça correspon a la interrupció sèrie), un cop l'adreça pren el valor 1800H, és a dir, quan hem llegit 2Kb de dades, la inicialitzem de nou a la posició 1000H. L'adreça ADDRE és l'adreça que conté el que s'ha d'escriure a la targeta SD, en aquest cas l'adreça s'incrementa mitjançant la funció WRITEBLK que és l'encarregada d'escriure un bloc a la targeta SD, tal com pasa amb l'adreça ADDRL, quan pren el valor 1800H, s'inicialitza de nou a 1000H. L'última adreça TARG conté la direcció de memòria de la targeta SD on s'escriuran les dades, aquesta adreça cal incrementar-la cada cop que s'escriu en 200H, ja que els blocs són de 512 bytes. Per últim cal comentar que tal com s'ha explicat amb les adreces ADDRL i ADDRE, hi ha un buffer de 2Kb, aquest buffer té el seu inici a la posició 1000H de memòria i el seu final a la posició 1800H

```
ADDRL DW 1000H ;ADREÇA DE LECTURA 30H,31H
ADDRE DW 1000H ;ADREÇA D'ESCRIPTURA 32H,33H
TARG  DW 0410H ;ADREÇA DE LA TARGETA SD ON ESCRIVIM 34H,35H
ORG 1000H ;ADREÇA ON ES GUARDA EL QUE ARRIBA PEL PORT SERIE
ADDR  DB 54H
```

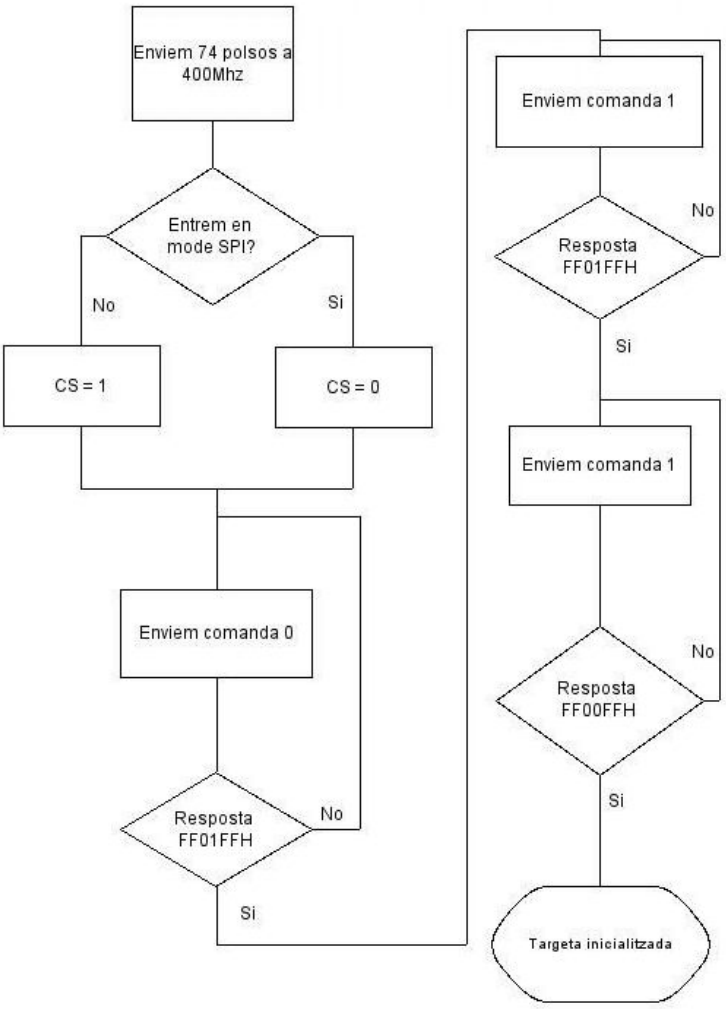


Figura 4.2: Diagrama d'inicialització

4.2.1 Inicialització de la targeta

Per despertar la targeta cal enviar 74 polsos de rellotge a una frequencia de 400Khz, fem servir 2 rutines diferents (WAIT, WAIT2) per fer el clock, ja que quan el clock es a 1 es fa un DJNZ que triga 2 unitats de temps. Això es fa perque el clock quedi perfectament sincronitzat i no hi hagi un desfasament entre l'1 i el 0. Cal notar que la senyal CS es posa a 1 mentre s'envien els polsos de rellotge, cal fer-ho aixi per despertar la targeta.

`OPEN:`

```
MOV B,#74D
CLK74: MOV PORTSD,#02H; CLK=0 CS=1
LCALL WAIT
MOV PORTSD,#03H; CLK=1 CS=1
LCALL WAIT2
DJNZ B,CLK74

WAIT: MOV R3,#9D
DJNZ R3,$
RET

WAIT2: MOV R3,8D
DJNZ R3,$
RET
```

Quan es desperta la targeta es troba en el mode SD (per defecte). Per fer-la entrar en el mode SPI cal enviar la comanda 0 (reset), mantenint el senyal CS a 0, aquest senyal es actiu a nivell baix. Un cop enviat cal esperar la resposta a la comanda per veure si s'ha entrat en el mode SPI correctament la resposta que esperem es FF01FFH

```
SEND0:
MOV DPTR,#CMD0
LCALL SNDCMD

MOV DPTR,#RESP; Direcció on es guarda la resposta a la comanda
MOV NBYTES,#3D
ACALL GTRESP
MOV DPTR,#RESP

MOVX A,@DPTR
CJNE A,#0FFH,SEND0; Cal verificar que la resposta sigui correcte
INC DPTR

MOVX A,@DPTR
CJNE A,#01H,SEND0 ;
INC DPTR

MOVX A,@DPTR
CJNE A,#0FFH,SEND0
```

Tot seguit cal enviar la comanda 1, que s'encarrega d'activar el proces d'inicialització de la targeta. Cal enviar-la dues vegades per asegurar-nos que s'inicialitza correctament. Enviem la comanda i esperem la resposta en aquest cas FF01FFH, tornem a enviar la comanda i esperem de nou la resposta aquest cop ha de ser FF00FFH, si tot es correcte la targeta queda totalment inicialitzada i a punt per poder llegir i escriure.

```
SEND11:

MOV DPTR,#CMD1
LCALL SNDCMD
MOV DPTR,#RESP
MOV NBYTES,#3D
LCALL GTRESP
MOV DPTR,#RESP
MOVX A,@DPTR
CJNE A,#0FFH,SEND11
INC DPTR

MOVX A,@DPTR
CJNE A,#01H,SEND11
INC DPTR

MOVX A,@DPTR
CJNE A,#0FFH,SEND11
LCALL WAIT

SEND12:
MOV DPTR,#CMD1
ACALL SNDCMD
MOV DPTR,#RESP
MOV NBYTES,#3D
ACALL GTRESP
MOV DPTR,#RESP
MOVX A,@DPTR
CJNE A,#0FFH,SEND12
INC DPTR

MOVX A,@DPTR
CJNE A,#00H,SEND12
```

```
INC DPTR

MOVX A,@DPTR
CJNE A,#0FFH,SEND12
LCALL WAIT
```

4.2.2 Interrupció sèrie

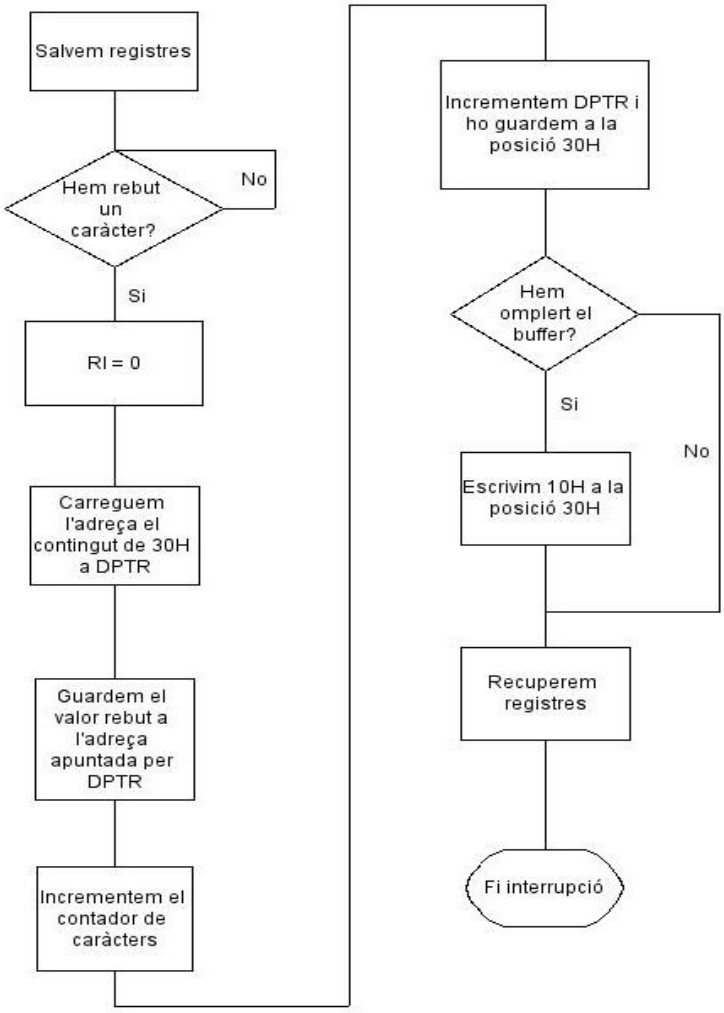


Figura 4.3: Diagrama Interrupció sèrie

Un cop ja hem activat la interrupció sèrie, procedim a crear el codi de la

interrupció. En aquest codi podem veure, tal com fan totes les interrupcions, com es guarden a pila tots els registres que es modifiquen dins la interrrupció. Un cop s'han salvat els registres a utilitzar per la interrupció, ja podem modificar els registres que calgui sense que això afecti al correcte funcionament del programa. Tal com hem explicat abans l'interrupció sèrie fa servir el punter a memòria ADDRL i aquest punter està en les posicions de memòria 30H i 31H, doncs el primer que fem es càrregar aquestes dues posicions de memòria a DPH i DPL, respectivament. Un cop ja tenim l'adreça (indirecte) on s'ha d'escriure al registre DPTR, ja podem moure el que ens ha arribat pel port sèrie al buffer de 2 Kbytes que habiem creat. Després, cal incrementar el registre R1, que tal com hem comentat s'encarrega de portar el control de quants bytes hi ha al buffer, i el registre DPTR, que un cop incrementat es torna a guardar a memòria, així quan es torni a produir una interrupció sèrie la següent dada serà copiada a la següent posició de memòria. Per últim, caldrà comprovar que el registre DPTR no ha arribat al valor 1800H, última posició del buffer, si hi ha arribat cal sobreescriure la posició de memòria 30H amb el valor 10H, per tornar al principi del buffer

```
SERIE:
PUSH PSW
PUSH ACC
PUSH DPH
PUSH DPL

JNB RI,FORA

CLR RI
MOV A,SBUF
MOV DPH,30H
MOV DPL,31H
MOVX @DPTR,A
CALL PUTCHAR
INC R1

INC DPTR
MOV 30H,DPH
MOV 31H,DPL

MOV A,DPH
CJNE A,#18H,FORA
MOV 30H,#10H
```

```
FORA: POP DPL
POP DPH
POP ACC
POP PSW
RETI
```

4.2.3 Lectura de blocs

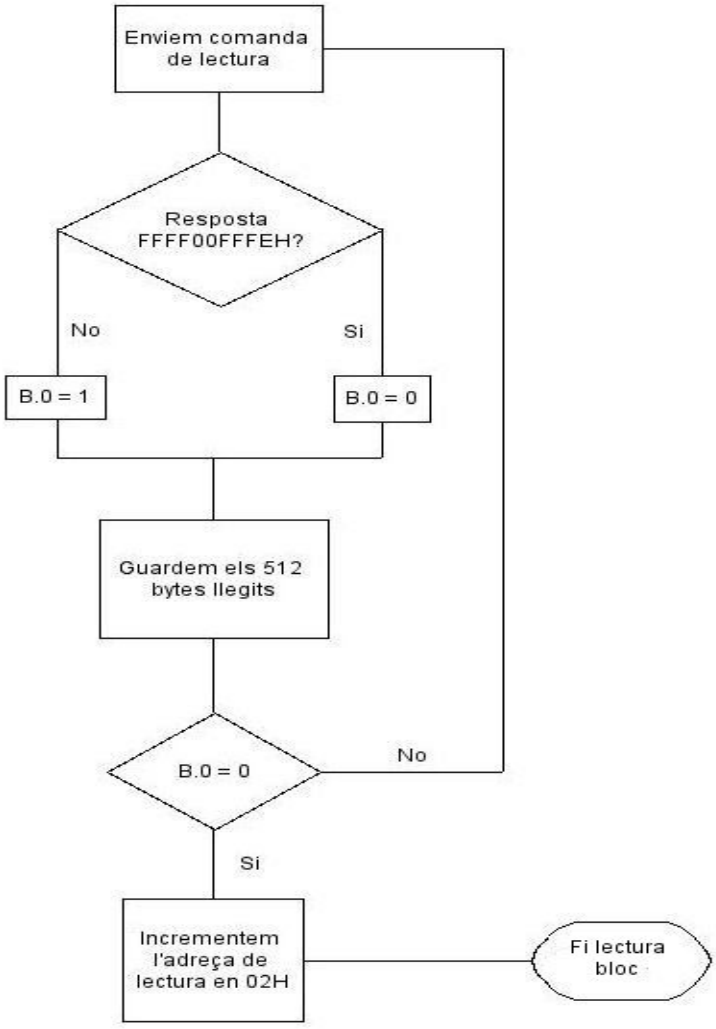


Figura 4.4: Diagrama de lectura

La següent funció es l'encarregada de llegir un bloc de 512 bytes de la targeta SD, el primer byte esta compost per dos bits inicials que són 0 i 1 respectivament i els següents 5 bits pertanyen al codi de comanda. Els següents 4 bytes pertanyen a la direcció de la targeta SD que volem llegir, l'últim byte es deixa a 0 ja que llegim de 512 bytes en 512 bytes, i el bit 0 del penultim byte s'ha d'anar incrementant de 2 en 2 per la mateixa raó. El sise byte conté els 7 bits de CRC i un últim bit amb valor 1. Un cop enviada la comanda de 6 bytes, esperem la resposta de la targeta SD la cual es de 5 bytes que han de ser FFH, FFH, 00H, FFH i FEH, respectivament. L'últim byte correspont a l'start bloc, es a dir, que la targeta esta preparada per ser llegida. Si tot va bé es llegeix la targeta i continuem amb l'execució del programa. En cas contrari es torna a enviar la comanda i a repetir el procediment de comprobació.

```
READBLK:
CLR B.0 ;1cicle
MOV A,#51H ;1cicle
ACALL SNDBYTE ;2cicles
MOV A,#00H ;1cicle
ACALL SNDBYTE ;2cicles
MOV A,#01H ;1cicle MOV DPTR,#ADDR
;MOVX A,@DPTR
ACALL SNDBYTE ;2cicles
MOV A,#00H ;1cicle ;INC DPTR
;MOVX A,@DPTR
ACALL SNDBYTE ;2cicles
MOV A,#00H ;1cicle
ACALL SNDBYTE ;2cicles
MOV A,#01H ;1cicle
ACALL SNDBYTE ;2cicles
ACALL WAIT ;2cicles
ACALL SUPERTOKEN ;2cicles
NADA: MOV DPTR,#RESP ;2cicles
MOV NBYTES,#255D ;2cicles
ACALL GTRESP ;2cicles
;MOV NBYTES,#255D
;MOV DPTR,#RESP
;ACALL IMPR
MOV NBYTES,#255D ;2cicles
ACALL GTRESP ;2cicles
```

```
MOV NBYTES,#2 ;2cicles
ACALL GTRESP ;2cicles

ACALL WAIT ;2cicles
JB B.0,READBLK ;2cicles
RET ;2cicles

SUPERTOKEN:
MOV DPTR,#RESP ;2cicles
MOV NBYTES,#5D ;2cicles
ACALL GTRESP ;2cicles
MOV DPTR,#RESP ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FFH,NOES ;2cicles
INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FFH,NOES ;2cicles
INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#00H,NOES ;2cicles
INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FFH,NOES ;2cicles
INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FEH,NOES ;2cicles
CLR B.0 ;1cicle
SJMP FIN ;2cicles
NOES: MOV DPTR,#NOES2
ACALL IMPR2
SETB B.0 ;1cicle
FIN:
MOV DPTR,#LLEGIT
ACALL IMPR2

RET ;2cicles
```

4.2.4 Escriptura de blocs

La funció WRITEBLK s'encarrega d'escriure un bloc de 512 bytes a la targeta SD. El primer byte esta compost pels bits 0 i 1 seguits dels 5 bits de

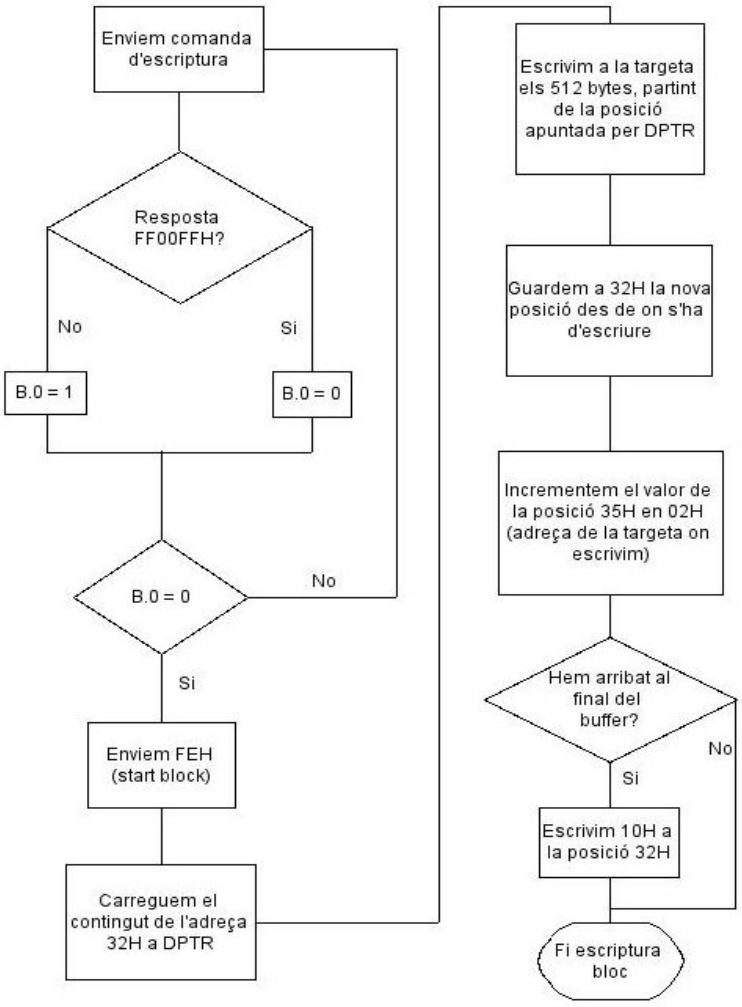


Figura 4.5: Diagrama d'escriptura

comanda. Els següents 4 bytes pertanyen a la direcció de la targeta SD on volem escriure. Tal com pasa amb la funció READBLK, el primer i l'últim byte de la direcció es deixen a 0, per construir les adreces on s'ha d'escriure fem servir dues posicions de memoria la 34H i la 35H, la 34H te un valor de 01H i la posició 35H té un valor inicial de 00H el qual es va incrementan en 02H, després de cada escriptura per anar apuntant als següents blocs. Un cop enviem els 6 bytes corresponents de la comanda, es fa la comprovació per veure si tot s'ha rebut correctament; en aquesta comprovació esperem rebre per part de la targeta els següents bytes FFH, 00H i FFH. Un cop

tot es correcte enviem el byte FEH, corresponent a l'inici de trama (aquest byte indica que el que ve a continuació són les dades a guardar a la targeta). Un cop fet això carreguem les adreces 32H i 33H als registres DPH i DPL, respectivament. Ara ja tenim el registre DPTR apuntan a la posició on són les dades a escriure. Aquestes dades són en un buffer de 2KB, un cop s'emplena el buffer es torna a reescriure sobre la posició inicial. Ara només cal anar enviant byte a byte, a traves de l'acumulador, les dades a escriure, un cop s'envia una dada s'incrementa el DPTR per apuntar al següent byte a escriure. Finalment, un cop s'han transmes els 512 bytes, es guarda el que hi ha als registres DPH i DPL a les posicions de memoria 32H i 33H, respectivament. Ara cal comprobar que no sobrepasem el limit de 2KB que té el nostre buffer, seguidament sumem 02H a la posició de memoria 35H, aquesta posició com hem comentat abans te el byte baix de l'adreça de la targeta SD on volem escriure.

```
WRITEBLK: ;67287 cicles 43806,636 microsegons
MOV A,#58H ;1cicle ;NOMBRE DE COMANDA
ACALL SNDBYTE ;2cicles

MOV A,#00H ;1cicle
ACALL SNDBYTE ;2cicles
MOV A,34H ; ;POSICIO DE MEMORIA DE LA TARJETA ON S'ESCRIU
ACALL SNDBYTE ;2cicles
MOV A,35H ;1cicle
ACALL SNDBYTE ;2cicles
MOV A,#00H ;1cicle
ACALL SNDBYTE ;2cicles

MOV A,#01H ;1cicle ;CRC
ACALL SNDBYTE ;2cicles

ACALL WAIT2 ;2cicles   735cicles
MOV DPTR,#RESP ;2cicles
MOV NBYTES,#6D ;2cicles
ACALL GTRESP ;2cicles

MOV DPTR,#RESP ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FFH,OTRA ;2cicles

INC DPTR ;2cicles
```

```
MOVX A,@DPTR ;2cicles
;CJNE A,#00H,OTRA ;2cicles

INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
CJNE A,#0FFH,OTRA ;2cicles

CLR B.0 ;1cicle
SJMP OK ;2cicles
OTRA: SETB B.0 ;1cicle
OK: JB B.0,WRITEBLK ;2cicles

MOV A,#0FEH ;1cicle                    ;Enviem l'start bloc, despres ja s'envien les
ACALL SNDBYTE ;2cicles

;MOV DPTR,#FRASE
MOV DPH,32H;
MOV DPL,33H; ;MOV DPTR,#FRASE ;2cicles;s'ha d'anar incrementant l'adreça!!!!

MOV B,#255D ;1cicle
WRITE1: MOVX A,@DPTR ;2cicles
;LCALL PUTCHAR
ACALL SNDBYTE ;2cicles
INC DPTR ;2cicles

DJNZ B,WRITE1 ;2cicles

;MOV DPTR,#ADDR ;2cicles
MOV B,#255D ;1cicle
WRITE2: MOVX A,@DPTR ;2cicles
;LCALL PUTCHAR
ACALL SNDBYTE ;2cicles
INC DPTR ;2cicles

DJNZ B,WRITE2 ;2cicles

;MOV DPTR,#ADDR ;2cicles
MOV B,#2D ;1cicle
WRITE3: MOVX A,@DPTR ;2cicles ENS HO PODRIEM ESTALVIAR POSANT B=0 ENLLOC DE B=255
;LCALL PUTCHAR
ACALL SNDBYTE ;2cicles
```

```
INC DPTR ;2cicles

DJNZ B,WRITE3 ;2cicles

MOV 32H,DPH
MOV 33H,DPL ;GUARDEM LA NOVA POSICIO DESDE ON S'HA D'ESCRIURE

MOV A,DPH ;Hem escrit 2k dades i tornem el punter a l'inici
CJNE A,#18H,FORA
MOV 32H,#10H

MOV A,#0FEH ;1cicle ;Enviem el CRC, pero com que no fem comprovació es pot en
ACALL SNDBYTE ;2cicles
MOV A,#0FEH ;1cicle
ACALL SNDBYTE

MOV A,#'R'
LCALL PUTCHAR
MOV A,#'S'
LCALL PUTCHAR
MOV A,#'P'
LCALL PUTCHAR

MOV DPTR,#RESP2 ;2cicles
MOV NBYTES,#3D ;2cicles
ACALL GTRESP ;2cicles

MOV DPTR,#RESP ;2cicles
MOVX A,@DPTR ;2cicles

ACALL BITS

INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles


ACALL BITS

INC DPTR ;2cicles
MOVX A,@DPTR ;2cicles
```

```
ACALL BITS

;MOV DPTR,#RESP ;2cicles
;MOV NBYTES,#3D ;2cicles
;ACALL IMPR ;2cicles


MOV A,35H
ADD A,#02H
MOV 35H,A

RET ;2cicles
```

4.2.5 Funcions auxiliars

La funció IMPR es una funció auxiliar amb la qual podem escriure per pantalla el que li pasem a traves del registre DPTR

```
IMPR: MOVX A,@DPTR
LCALL PUTCHAR
INC DPTR
DJNZ NBYTES,IMPR
RET
```

La funció PUTCHAR es l'encarregada d'enviar pel port serie el que hi hagi a l'acumulador

```
PUTCHAR: JNB TI,$
CLR TI
MOV S0BUF,A
RET
```

La funció SNDCMD envia una comanda de 6 bytes que es troba a una posició de memòria on apunta DPTR

```
SNDCMD:
MOV R7,#6D
BYTE: CLR A
MOVC A,@A+DPTR
```

```
LCALL SNDBYTE
INC DPTR; Apuntem al següent byte
DJNZ R7,BYTE
RET
```

La funció SNDBYTE envia un byte (bit a bit), aquest byte es pasa com a parametre a l'acumulador.

```
SNDBYTE:
MOV R6,#8D
CLR C
ACAS: RLC A
JNC ZERO
ONE: MOV PORTSD,#ENVIA1_B
LCALL WAIT
MOV PORTSD,#ENVIA1_A
LCALL WAIT2

SJMP SEW
ZERO: MOV PORTSD,#ENVIA0_B
LCALL WAIT
MOV PORTSD,#ENVIA0_A
LCALL WAIT

SEW: DJNZ R6,ACAS
RET
```

La funció GTRESP, s'encarrega de llegir tant bytes com s'indiquin a la variable NBYTES, deixa els bytes llegits a la posició de memòria on apunta DPTR

```
GTRESP:
LCALL RD_BYTE
MOVX @DPTR,A
INC DPTR
DJNZ NBYTES,GTRESP
RET
```

La funció RD_BYTE s'encarrega de llegir de la targeta bit a bit. Deixa el byte llegit a l'acumulador per a posteriors operacions

```
RD_BYTE:
MOV R7,#8D
```

```
BBIT: MOV PORTSD,#RECIBE_B
LCALL WAIT
MOV C,DO
RLC A
MOV PORTSD,#RECIBE_A
LCALL WAIT
DJNZ R7,BBIT
RET
```

La funció WAIT40 fa un retard d'uns 40 microsegons

```
WAIT40:
MOV R3,#8D; Fem un retard de 40microsegons
NOP
ESP40: DJNZ R3,ESP40
RET
```

4.3 Temps d'execució

Hem fet un estudi sobre el temps que tarda el programa en inicialitzar la targeta SD i el temps que tarda en fer una escriptura d'un bloc de 512 bytes, ja que entre una recepció d'un byte de dades del sensor analogic i un altre tenim un temps de 0,042 segons. Tenint en compte que per poder escriure a la targeta SD necesitem un bloc de 512 bytes (la targeta funciona a nivell de blocs) el temps total es de 21,5 segons.

Tenint en compte que l'oscil·lador que fem servir es de 18,432 MHz i que cada cicle de màquina del 80C552 està compost per 12 cicles de rellotge, tenim que:

Que representa un periode de:

$$\frac{18,432 * 10^6}{12} = 1536000 \text{ cicles de màquina per segon}$$

Que representa un temps de:

$$\frac{1}{1536000} = 651,0416 \text{ nanosegons per cicle de màquina}$$

Per tant cada cicle d'instrucció es fa en 651,0416 nanosegons. Un cop fets aquests càlculs només cal anar mirant a la documentació del fabricant, instrucció per instrucció quants cicles consumeix en executarse.

El temps que tarda en inicialitzar la targeta i deixar-la operativa per poder accedir-hi a fer lectures i/o escriptures es de 6945 cicles de màquina, és a dir, 4,521 milisegons.

El temps que tarda a fer una escriptura es de 67287 cicles d'instrucció, és a dir, 43,806 milisegons. Un cop veiem el que triga en escriure un bloc de 512, ja podem fer el càlcul fer saber si l'escriptura es pot produir dins el rang de temps que tenim disponible. Saben que l'escriptura tarda 43,806 milisegons i hem d'escriure 512 bytes, tenim que:

$$\frac{43,806}{512} = 85.55 \text{ microsegons}$$

Tal com hem vist abans entre la recepció d'una dada i un altre tenim un temps de 0,042 segons. Aquest 0,042 segons s'han de dividir en dos una part per la interrupció serie i l'altre per l'escriptura de la dada a la targeta. Tenint en compte això veiem que l'escriptura a la targeta ens consumeix 86 microsegons dels 0,042 segons disponibles, per tant, tenim 41914 microsegons per la interrupció serie, temps més que suficient per executar un codi que s'encarregui de guardar una dada a memòria.

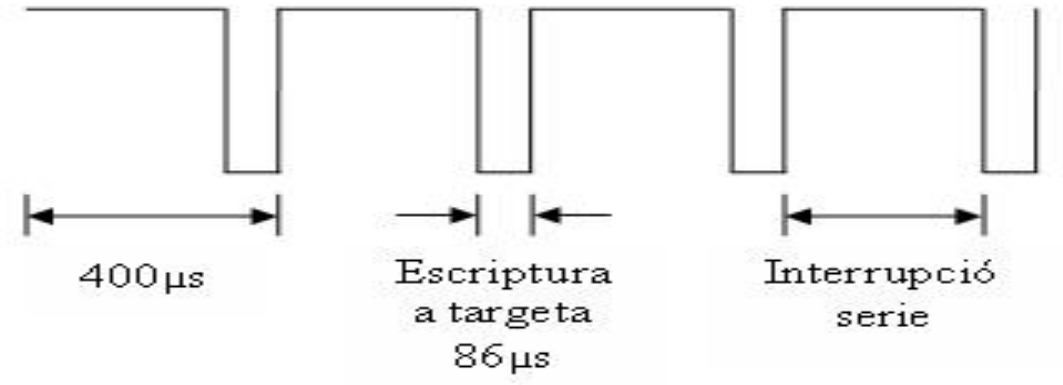


Figura 4.6: Diagrama de flux

També em dut a terme l'estudi del temps d'execució en cas que el sistema fos ampliat a mostres de 500Hz, degut a que el rang de freqüències del bruixisme es de 20Hz a 500Hz, això ens donaria un byte cada 2 milisegons, enlloc dels 42 milisegons dels càlculs anteriors. Si als 2 milisegons li traiem el temps que tarda en escriure la targeta SD un bloc de 512, és a dir, 86 microsegons ens queden 1914 microsegons per la interrupció sèrie, que segueix sent un temps més que suficient per a qualsevol interrupció serie.

A part de fer l'estudi del temps d'execució també cal fer un estudi de capacitat, per poder veure si podem capturar dades durant 24 hores seguides.

En el projecte anterior es guardaven les dades a la memòria interna de l'Olorim, 16 Kbytes, aquesta memòria s'emplenava rapidament a l'ordre d'uns 680 segons (11 minuts amb 20 segons). En el cas de fer el mostreig a 500Hz, la memòria s'emplena en més o menys uns 30 segons. Com es pot veure amb una memòria de 60 Kbytes no podem enregistrar més d'11 minuts seguits, temps insuficient per enregistrar dades mentre la persona dorm.

Amb la targeta SD es preten millorar aquest apartat del projecte anterior. En aquest cas farem l'estudi sobre una freqüència de mostreig de 500Hz. Primer de tot suposem que una persona dorm durant 12 hores seguides, tenim que en 1 segon rebem 500 bytes:

$$\frac{500bytes}{1segon} x \frac{60segons}{1minut} x \frac{60minuts}{1hora} = 1800000 \text{ bytes/hora}$$

$$\frac{1800000bytes}{1hora} x 12hores = 21600000 \text{ bytes}$$

Ara veurem quan és el màxim de temps que podem capturar de forma continuada amb una targeta SD de 4GB (4294967296 bytes):

Fem una regla de 3:
Si 1s → 500 bytes, aleshores
x segons → 4294967296 bytes

$$x = \frac{4294967296bytes * s}{500bytes} = 8589934{,}592 \text{ segons}$$

Amb 4GB es pot capturar de forma continuada durant 99 dies 10 hores 5 minuts 34 segons a una freqüència de mostreig de 500Hz.

4.4 Representació gràfica

Un cop hem recollit les dades, que ens envia el sistema de captura, dins la targeta SD. Ens hem trobat que teniem molta informació emmagatzemada del pacient, però no teniem una manera de mostrar-la clarament a l'usuari final. Per aquest motiu hem decidit fer un programa per veure aquestes dades d'una manera més representativa mitjançant una gràfica. Aquesta representació gràfica l'hem dut a terme amb el software Matlab, que ens permet fer representacions gràfiques amb un bon resultat final. Además Matlab ens permet fer ampliacions de parts de la gràfica, d'aquesta manera podem veure amb més precisió les parts més significatives de la gràfica.

Per fer la representació gràfica primer de tot cal obrir el fitxer amb la funció:

```
f = fopen(fitxer,'r')
```

després s'ha de llegir el fitxer en format unsigned de 8 bits (d'aquesta manera evitem que matlab interpreti els valors com caràcters ASCII) i guardarlo en un vector:

```
x = fread(f,inf,'uint8')
```

Finalment després de fer la transposta del vector anterior (cal fer la transponsta ja que la funció read guarda els valors a la matriu modifican l'index de les y, per tant, ens genera y vectors d'un sol element i nosaltres volem un vector amb y elements), dibuixem la gràfica:

```
plot(x)
```

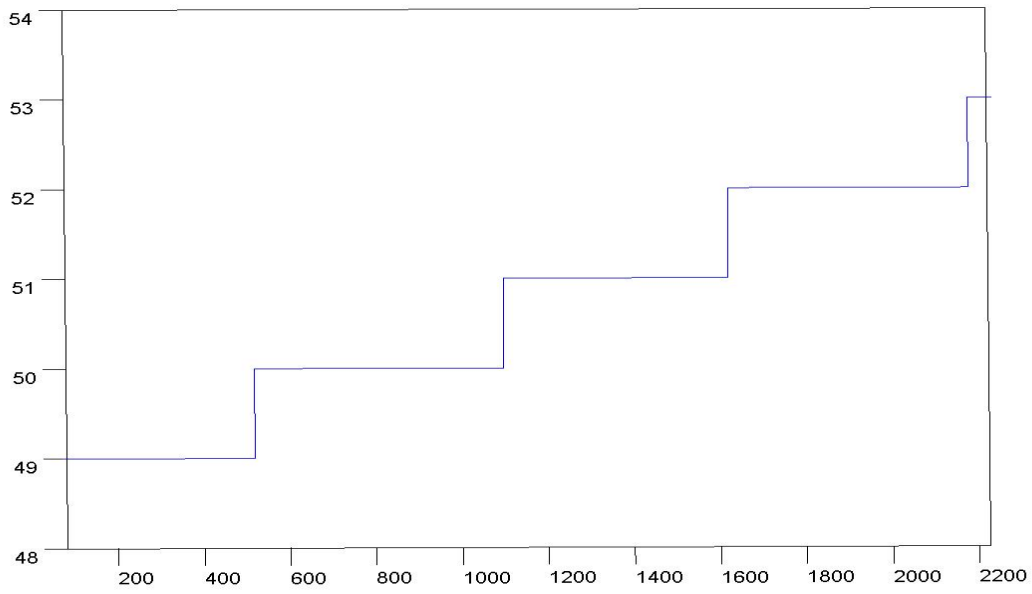


Figura 4.7: Grafica Matlab

Capítol 5

Conclusions

Tot i que al principi aquest projecte semblava fàcil, ja que bàsicament era fer un "driver"d'una targeta SD per poder llegir i escriure-hi, no ha estat tant fàcil com semblava perque abans de començar aquest projecte no sabia com funcionava internament una targeta SD, com la majoria només la havia vist funcionar en camères fotogràfiques digitals o altres dispositius.

Un cop acabat el projecte hem après:

- Funcionament intern d'una targeta SD
- Arquitectura de la targeta SD
- Inicialitzar la targeta
- Comunicació amb la targeta:
  - Enviament de comandes
  - Enviament i recepció de dades

Amb aquest projecte també he tingut l'oportunitat d'aplicar els coneixements obtinguts en les diferents assignatures de la carrera, en especial de l'assignatura computadors de 2º, i he pogut comprovar que amb aquests coneixements i buscant la informació necesària que ens pugui faltar es possible resoldre qualsevol problema sobre el qual no en teniem cap conexiament previ.

A part dels coneixements tècnics adquirits amb el desenvolupament del projecte, també he obtingut coneixements o eines que ens poden servir per a la nostra vida profesional, com pot ser escriure la documentació del projecte, que tot i que es una feina molt farragosa per una persona que ha escollit estudis tècnics, ens valdra per a la nostra vida profesional ja que quan una

empresa disposa d'un nou software cal que el tècnic fagi la documentació perque els usuaris puguin fer servir aquella aplicació.

Finalment hem après a com s'ha de fer per redactar una documentació tècnica, que es el que cal explicar perque una persona amb uns mínims de coneixements sobre la materia pugui interactuar amb el software/hardware que hem desenvolupat.

Bibliografia

[1] "Sd card association.," 2000. http://www.sdcard.org/home.

[2] "Secure digital card interface for the msp430," 2000. http://www.cs.ucr.edu/amitra/sdcard/Additional/sdcard_appnote_foust.pdf.

[3] "Embedded sd card/mmc card," 2002. http://www.freelabs.com/ whitis/sd_card/.

[4] "Mmc/sd card reader example application," 2006. http://www.roland-riegel.de/sd-reader/index.html.

[5] "Targeta sd sobre microcontrolador 8051," 2006. http://homepages.mty.itesm.mx/al778081.

[6] "Sd specifications part 1 physical layer simplified specification," Setembre 2006. http://www.sdcard.org/about/memory_card/pls/Simplified_Physical_Layer_Spec.pdf.

[7] "Los chipsets de la placa madre," 2006.

[8] "Compact flash, sd and sdhc memory cards," 2007. http://www.bobatkins.com/photography/digital/compact_flash_memory_cards.html.

[9] "Secure digital," 2008. http://es.wikipedia.org/wiki/Secure_Digital.

[10] "Northbridge," 2008. http://en.wikipedia.org/wiki/Northbridge_

[11] *SanDisk Secure Digital Card Product Manual - Revision 1.9*, Desembre 2003.

[12] *olorim 2.3 Manual de l'usuari*, març 2003.

[13] "Sistema de fitxers," 2008. http://es.wikipedia.org/wiki/Sistema_de_archivos.

[14] "Microsoft extensible firmware initiative fat32 file system specification," Desembre.

[15] "Ext2," 2008. http://es.wikipedia.org/wiki/EXT2.

[16] "Ext3," 2008. http://es.wikipedia.org/wiki/Ext3.

[17] "Jfs," 2008. http://es.wikipedia.org/wiki/JFS.

[18] "Xfs," 2008. http://es.wikipedia.org/wiki/XFS.

[19] "Compact flash," 2008. http://es.wikipedia.org/wiki/Compactflash.

[20] "Rs-232," 2008. http://es.wikipedia.org/wiki/COM1.

[21] "Null modem," 2008. http://en.wikipedia.org/wiki/Null_modem.