Descodificador MP3 VS1011e

Descripció i aplicació

Documents Treball de Recerca
Gerard Marull Paretas
Maig 2007

Índex

1. Introducció

Una altra part fonamental del nostre dispositiu, és la reproducció de fitxers de música codificats en format MP3. Aquesta feina, és un quant complexa i requereix l'ús d'una quantitat elevada de recursos. El nostre microcontrolador (en aquest cas Atmega8) no és prou potent per dur a terme aquesta tasca. Per tant, ens veurem obligats a utilitzar un component extern que s'encarregui exclusivament de fer aquest procés. El fabricant VLSI ofereix la sèrie de descodificadors d'àudio VS10XX, que són de fàcil implementació i que fins i tot alguns models no només descodifiquen MP3, sinó que poden fer-ho en altres formats com WMA, WAV, MIDI, AAC, OGG... com també són capaços de transformar so analògic (ex. d'un micròfon) a MP3, entre d'altres funcions.

El model que utilitzarem en aquest cas, és el VS1011e, que ofereix la descodificació de fitxers MP3, WAV i algunes altres funcions. S'utilitza aquest model perquè es troba disponible en el tipus d'encapsulat SOIC-28, relativament fàcil de soldar i per tant còmode per fer proves. Altres versions només es troben en encapsulat TQFP-48, així que probablement en la versió final es canviï a aquest segon format per reduir espai, i també aconseguir noves funcionalitats. En tot cas, la nova implementació implicaria una variació mínima

El material necessari és el mateix descrit en l'anterior capítol, afegint el següent:

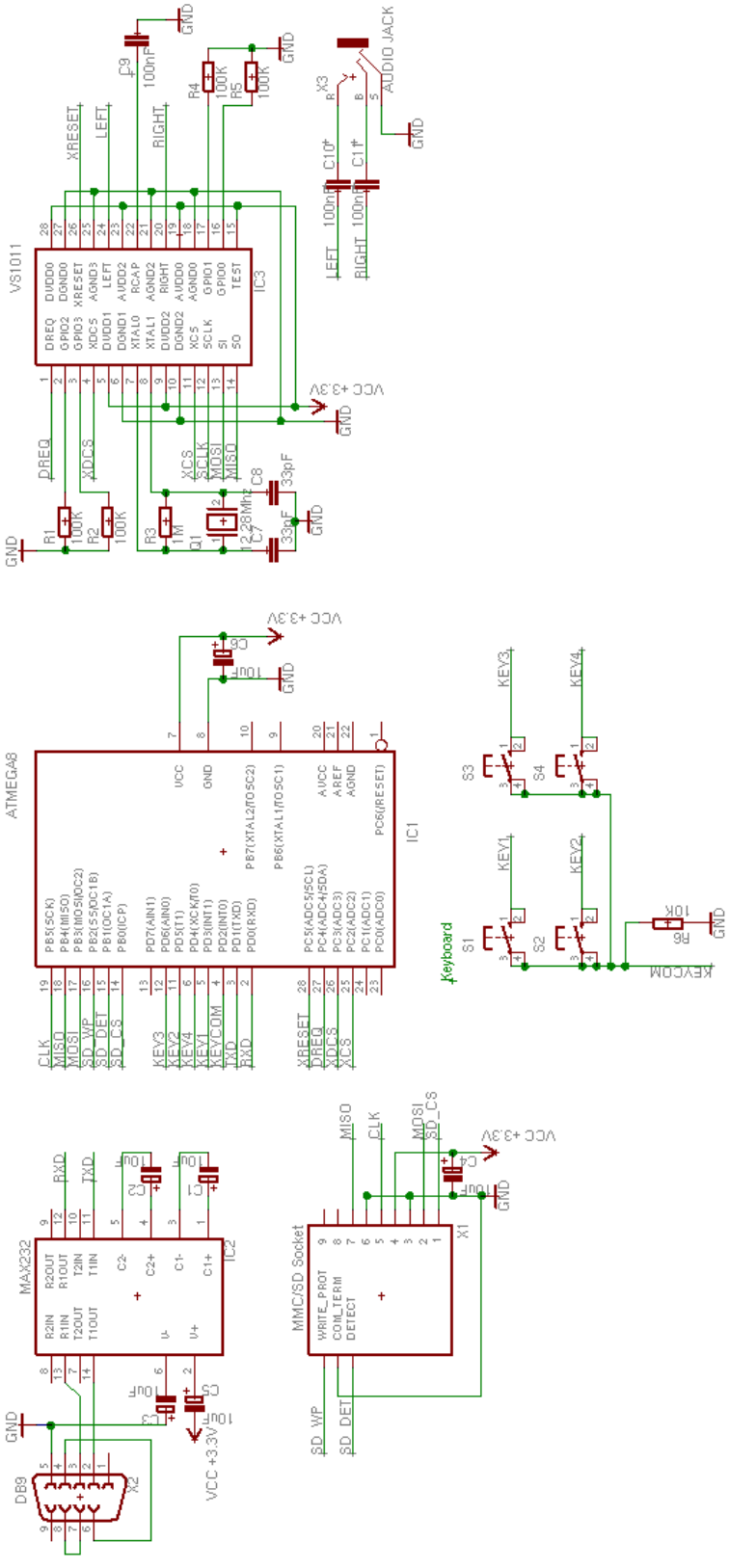
- Components diversos descrits en l'esquema de l'apartat 2.
- Auriculars

Notes: Tot i que en l'esquema es mostra el teclat, no es tractarà la implementació en aquest capítol, ja que no és definitiu.

Agraïments en especial a Qibo Zhang per ajudar-me a solucionar alguns problemes sorgits durant la implementació del descodificador.

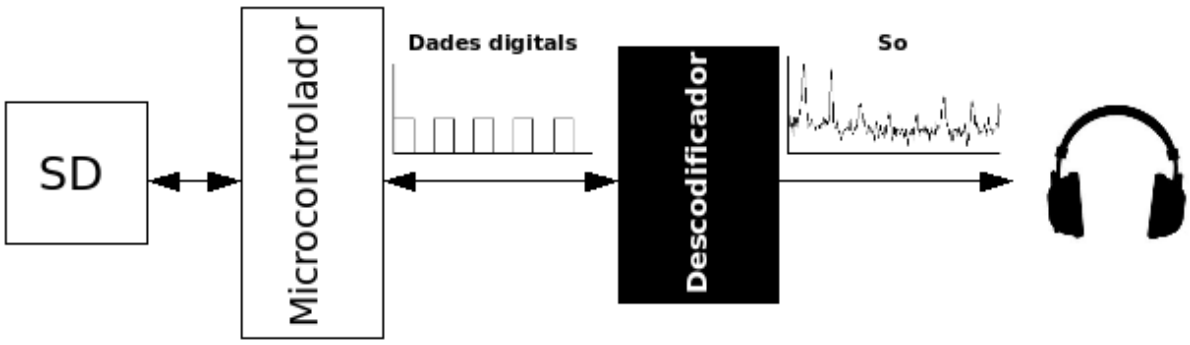
2. Esquema de connexions

L'esquema que següent mostra tots els components necessaris i laconnexió que hi ha entre ells per tal de poder executar i provar el programa d'aquest apartat. *Per obtenir un esquema en tamany real, referiu-vos a l'annex 3.A del CD-ROM.*



3. Funcionament del descodificador

El descodificador, no fa res més que transformar les dades que rep per part del microcontrolador del fitxer de música (senyals digitals) en un tipus de senyals analògics que connectats a un altaveu, generen les ones sonores corresponents. L'esquema següent ens mostra d'una manera simplificada aquest procés:



Com dèiem, hem de proveir el contingut d'un fitxer MP3 al descodificador, i per fer-ho, utilitzarem el sistema SPI (*Serial Peripheral Interface*) que el microcontrolador porta incorporat. Això ens podria semblar un problema: La targeta SD també fa ús de les línies del sistema SPI (SCK, MISO i MOSI) i aparentment les dades es podrien creuar. Però existeix un mètode molt simple de solucionar-ho. Consisteix en és l'ús d'un altre port en cada dispositiu que utilitzi el sistema SPI, de manera que cada dispositiu tindria els 3 ports comuns SPI, i un 4t, que l'anomenarem CS (*Chip Selec*t). El microcontrolador és capaç de controlar el nivell de cada un d'aquests ports (0 ó 1). Així doncs, mantindrem sempre tots aquests ports CS a nivell 1, i quan necessitem enviar dades a un dispositiu determinat, posarem a nivell 0 el port CS d'aquest dispositiu i enviarem les dades.

Solucionat aquest problema, procedirem a explicar el funcionament en detall.

3.1. Operacions amb el descodificador

Abans de poder fer cap pas, hem de conèixer algunes operacions bàsiques, com enviar-li dades o comandes, llegir registres, etc. D'aquesta manera podrem portar a terme futures tasques com graduar el volum, reproduir cançons, entre d'altres.

3.1.1. Registres

El descodificador, disposa d'una sèrie de registres dins la seva memòria que són accessibles (lectura / escriptura) per tal de configurar certs valors com el volum o obtenir informació del fitxer reproduït. Cadascun té un tamany de 16bits, i en formen un total de 16:

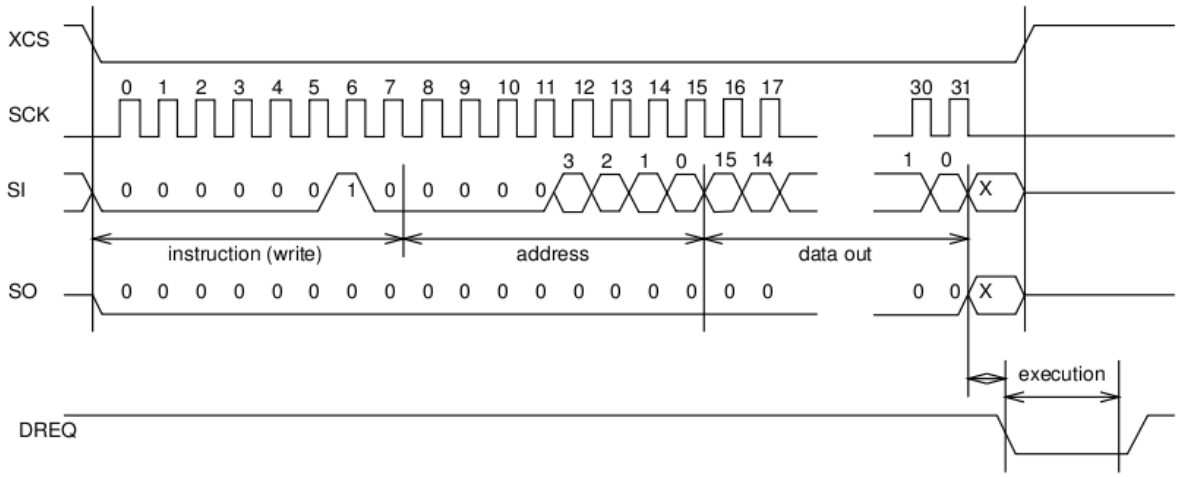
Adreça	Nom	Descripció
0x0	MODE	Configuració del mode d'operació
0x1	STATUS	Estat del descodificador
0x2	BASS	Control del Bass/Treble
0x3	CLOCKF	Freqüència del rellotge / multiplicador
0x4	DECODE_TIME	Temps de descodificació en segons
0x5	AUDATA	Informació del fitxer d'àudio descodficat

0x6	WRAM	RAM (escriptura/lectura)
0x7	WRAM_ADDR	Adreça per operar WRAM
0x8	HDAT0	Informació sobre el fitxer descodificat (de la capçalera del fitxer)
0x9	HDAT1	Informació sobre el fitxer descodificat (de la capçalera del fitxer)
0xa	AIADDR	Adreça on comença l'aplicació
0xb	VOL	Control del volum
0xc	AICTRL0	Registre pel control de l'aplicació
0xd	AICTRL1	Registre pel control de l'aplicació
0xe	AICTRL2	Registre pel control de l'aplicació
0xf	AICTRL3	Registre pel control de l'aplicació

No tots seran utilitzats, per tant només es descriuran aquells que es necessitin.

3.1.2. Escriptura dels registres

Necessitarem modificar alguns valors d'aquest registre, i per tant, haurem de crear una funció capaç de modificar-los. Aquesta funció, l'anomenarem *dec_write()*, i tindrà dos arguments: l'adreça del registre a modificar, i el contingut a escriure en aquest registre. Segons ens especifica el document del fabricant, l'escriptura ha de seguir el procediment mostrat en aquest diagrama de temps:



El nombre de bytes transferits serà un total de 32:

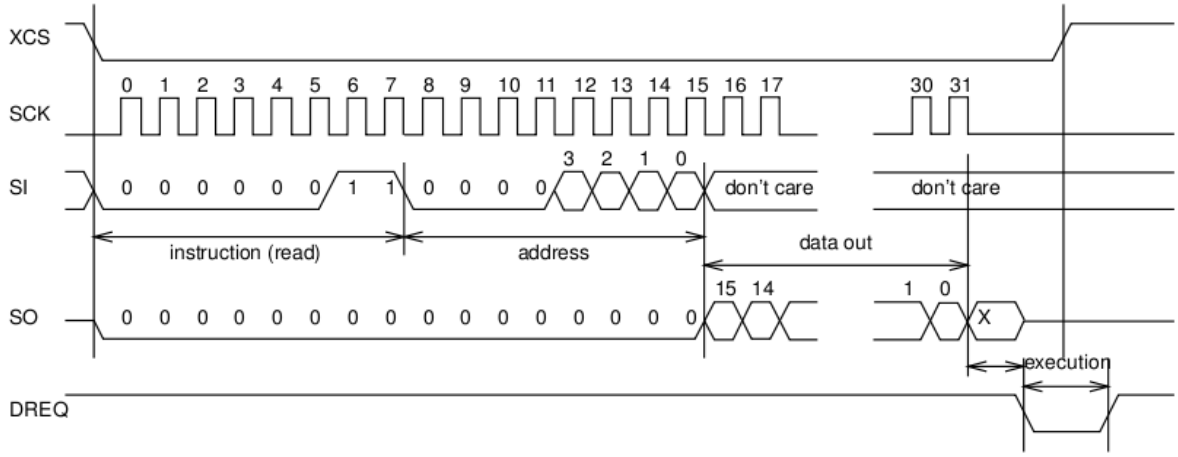
- 0-7 (8bits) - SI → Tipus d'instrucció (Escriptura o Lectura), en aquest cas Escriptura (0x2)
- 8-15 (8bits) - SI → Adreça del registre on es vol accedir
- 16-31 (16bits) - SI → Valor a escriure en el registre escollit

→ Fixeu-vos que en el moment en que comença l'enviament de la comanda, el port XCS (*Chip Select*), canvia a nivell baix, cosa que significa que les dades de la línia SPI són pel descodificador.

3.1.3. Lectura de registres

Així com hem creat una funció per l'escriptura, ens caldrà crear-ne una altra per la lectura de

registres: *dec_read()*. Només té un argument: l'adreça del registre que volem llegir. La funció retornarà els 16bits llegits. El diagrama de temps és el següent:



El funcionament és semblant al d'escriptura, però en aquest cas, es fa ús de la línia SO, on llegirem els 16bits del registre que hem demanat.

- 0-7 (8bits) - SI → Tipus d'instrucció (Escriptura o Lectura), en aquest cas Lectura (0x3)
- 8-15 (8bits) – SI → Adreça del registre on es vol accedir
- 16-23 (8 bits) – SO → Primers 8 bits del registre (Obtinguts com a resposta enviant qualsevol valor a través de SI).
- 24-31 (8 bits) – SO → Últims 8 bits del registre (Obtinguts com a resposta enviant qualsevol valor a través de SI).

3.1.4. Enviament de les dades d'àudio

Finalment ens queda l'enviament de dades del fitxer MP3. La funció encarregada de fer-ho serà *dec_datawrite()*. En aquesta funció entra en joc el pin DREQ. Per poder enviar dades d'àudio, necessitem que aquest pin estigui a nivell 1. Tot i això, serà una comprovació externa a aquesta funció. També es fa ús del port XDCS, per la sincronització del byte enviat. El procediment és el següent:

1. Posta del port XDCS a nivell 0
2. Enviament de 8bits corresponents al fitxer d'àudio.
3. Posta del port XDCS a nivell 1

Notes: Les dades, es recomana enviar-les en blocs de 32bits, per tant, la crida de 4 cops seguits aquesta funció.

3.2. Inicialització

Abans de poder fer qualsevol cosa, hem d'inicialitzar el dispositiu. Per fer-ho crearem la funció *dec_init()*. Els passos a seguir són molt concrets, i si algun fallés el descodificador es mantindria no operatiu. Aquest procés el realitzarem després d'haver inicialitzat el sistema SPI, just al començament de la funció principal *main()*.

1. Configuració dels ports XCS, XDCS, XRESET com a sortida, i DREQ com a entrada.
2. Reset de hardware. Realitzat posant el port XRESET a nivell baix durant uns pocs milisegons i tornant-lo a posar a nivell alt.
3. Posta del sistema SPI a baixa velocitat
4. Activació del sistema SPI natiu (Registre MODE, bit 11 a 1)

5. Graduació del registre CLOCKF. Si el nostre rellotge no és de 24.576Mhz (el valor per defecte) haurem de graduar el valor d'aquest registre. Aquest valor es pot calcular amb la formula 0x8000 + (FREQ/2000), on FREQ és la freqüència del rellotge utilitzat en Hz (En el nostre cas és de 12.28Mhz).
6. Posta del sistema SPI a la velocitat anterior.

*Algun d'aquests passos es poden repetir diversos cops fins observar que el registre ha estat modificat amb el valor que hem assignat. D'aquesta manera evitem que en fallar el primer intent la funció retorni error.

3.3. Control del volum

Per graduar el volum de l'àudio de sortida, el descodificador ens ofereix una manera molt simple, que fins i tot pot graduar el volum independentment per cadascun dels canals: dreta i esquerre. El volum consisteix en un valor de 16bits que pot anar de 0x00 (màxim) fins a 0xfefe (mínim). Però en realitat, aquest valor de16bits, són dos valors de 8bits:

Primer byte (Volum del canal esquerre)								Segon byte (Volum del canal dret)							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Total: 8bits, Màxim valor: 0xfe								Total: 8bits, Màxim valor: 0xfe							

Així doncs, cada canal té un valor del rang de 0 (0x00) a 254 (0xfe), en passos de 0.5dB.
La funció encarregada de canviar el volum, l'anomenarem *dec_setvol()*, i constarà de 2 arguments: el volum del canal esquerre i el del canal dret.

1. Escriptura al registre VOL del valor del canal esquerre (desplaçat 8 posicions a l'esquerra) sumat (OR) amb el valor del canal dret

3.4. Prova del mòdul

Un cop vistes totes les funcions necessàries, ens caldrà provar el nostre codi. Per fer-ho, reproduirem el primer fitxer MP3 que es trobi en el directori arrel de la targeta SD. El procediment detallat és el següent:

1. Inicialització dels diferents components/sistemes: SPI, targeta SD, sist. de fitxers FAT i per descomptat el descodificador.
2. Si tot això es duu a terme amb èxit, cridarem a la funció *fat_getentryinfo()* perquè ens desi les dades del primer fitxer MP3 localitzat a l'arrel.
3. Finalment, crearem un bucle que s'executi mentre la llargada de bytes llegida del fitxer no sigui 0 (és a dir, mentre no s'hagi acabat de llegir). Sota aquest bucle en crearem un altre de la llargada dels bytes llegits. Després, comprovarem si el pin DREQ es troba a nivell 1, i si és així, enviarem un bloc de 32bytes de dades. Repetirem aquests passos fins acabar la cançó.
4. Un cop acabada la cançó, enviarem 2048 bytes nuls (amb el valor 0x00) per indicar que la cançó s'ha acabat.

4. Bibliografia

VLSI, “VS1011e Datasheet”, http://www.vlsi.fi/datasheets/

VLSI, “VS10XX Application notes”, http://www.vlsi.fi/appnotes/

SCHILDT, Herbert. *Programación en C, Cuarta edición*. Madrid: McGraw Hill, 2001.

http://www.atmel.com/avr

http://www.avrfreaks.net

Agraïments: Qibo Zhang